Report from Lab Assignment #3
TDT4258 Energy Efficient Computer Systems

Emil Taylor Bye
Sigve Sebastian Farstad
Odd M. Trondrud

April 26, 2013

Abstract

This report presents a solution to assignment #3 of TDT4258 at the Norwegian University of Technology and Science during the spring of 2013. The assignment was to write a video game in C for Linux running on the Atmel STK1000 development board. Linux Device Drivers for the STK1000's buttons and LEDs were written in C. The solution presented in this report is a rhythm video game, implementing the .SM file format.

Contents

Part I
Introduction

This report presents a solution to assignment #3 of TDT4258 at the Norwegian University of Technology and Science during the spring of 2013. The objective of this lab assignment is to write a computer game titled "Scorched Land Defense" in C in a Linux-environment on the STK1000 development board.

The assignment's goal is to further students' skill in C programming in, introduce them to C-programming for Linux and further expose them to the inner workings of Linux and its kernel (including use of the Linux shell, compilation of the Linux kernel and writing kernel modules, hardware programming in and writing device drivers for Linux).

1 Scorched Land Defense

> In warfare, the policy of burning and destroying everything that might be of use to an invading army, especially the crops in the fields. [13]

Scorched Land Defense (SLD) is a warfare tactic. The assignment requests that a computer game with the given title be written.

We can interpret the quoted definition as describing a situation in which one entity (A) attacks a defending entity (B) which, rather than defending itself directly, attempts to destroy its own resources. A's objective thus becomes not simply the destruction of B, but also the prevention of B's scorched land defensive strategy while B's achieves victory by either successfully carrying out its defensive strategy or through eliminating the threat (i.e. the defeat of A).

Throughout the course of a "game" both entities perform actions with the intended goal of achieving victory. The end of a game is a state in which neither side cannot perform any actions or any of the victory conditions are met.

In video games, the player(s) typically controls some entity and is able to select what actions the entity should perform through some interface, typically a keyboard, gamepad, controller or screen.

1.1 Scorched Land Dance Dance Defense

In Scorched Land Dance Dance Defense (SLDDD) the entities have four distinct available actions throughout the course of a game. If performed within the specified time frame, the actions aid the entities in achieving their given victory conditions. The players' objective thus becomes to perform as many successful actions as possible throughout the course of a game.

The time window in which a given action will be successful if performed is presented to the player as an object moving across the screen. When the object reaches a specified point on the screen, the player must perform the action corresponding to its position. The closer the object is to the specified point at the time when the player performs the correct action, the more successful the action is. Successful actions result in the player receiving "points". More successful actions are worth more points than less successful actions. Some number of points is required to achieve a victory condition.

The player may choose which entity (or side) he or she is playing as. If more than two players play at the same time, they may imagine themselves both being the same entity working towards the same goal or separate entities fighting against each other.

2 DDR & StepMania

2.1 Dance Dance Revolution

Dance Dance Revolution (DDR) is a music video game series produced by Konami. The series' gameplay involves the player using his or her feet to step on portions of a platform corresponding to arrows that appear on the screen to the beat of a song. During gameplay arrows scroll across the screen (typically from the bottom of the screen and upwards), eventually passing over a set of stationary arrows before reaching the opposite end of the screen. The game's objective is for the player to hit the corresponding arrows on the *dance platform* when a scrolling arrow overlaps with the stationary arrows. The player is given a judgment based on the accuracy which with they hit each note. A sufficiently poor judgment will decrease the player's life bar. The player's life bar reaching zero results in a game over.

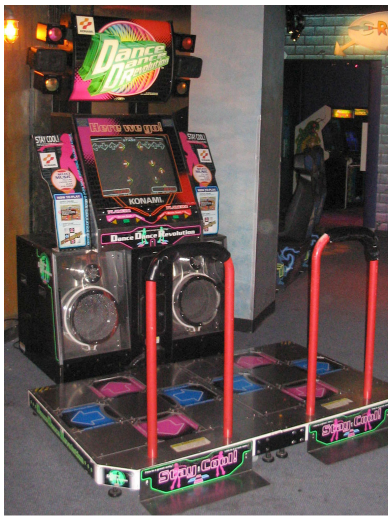
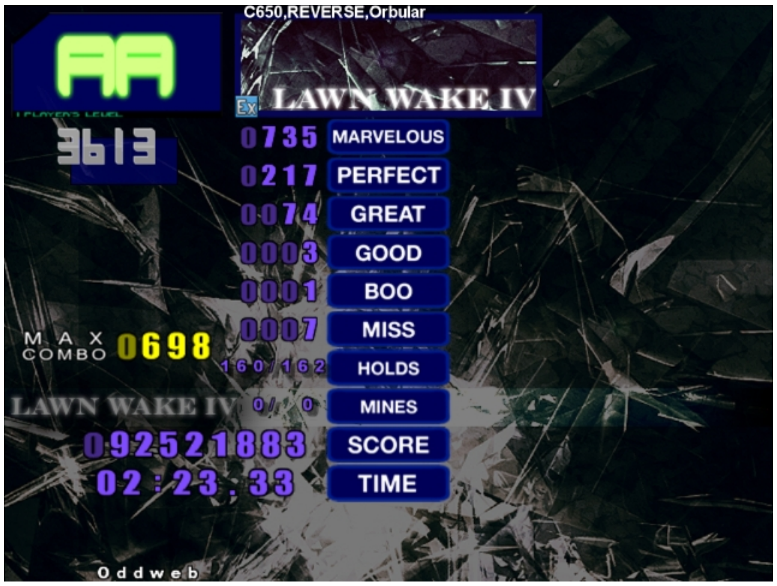


Figure 1: An original Dance Dance Revolution machine. Image courtesy of Poiuyt Man.

2.2 StepMania

StepMania (SM) is a cross-platform open source rhythm video game which was developed as a Dance Dance Revolution simulator, released under the MIT license. It allows for custom songs (known as "Stepfiles") and further modification of gameplay elements (such as the timing of judgments). While originally developed as a DDR simulator, it is often played using a keyboard rather than a dance platform.



C650,REVERSE,Orbular
AA
LAWN WAKE IV
3613

```
0735  MARVELOUS
0217  PERFECT
0074  GREAT
0003  GOOD
0001  BOO
0007  MISS
160/162  HOLDS
0/0  MINES
092521883  SCORE
02:23.33  TIME
```

MAX COMBO 0698
LAWN WAKE IV
Oddweb

Figure 2: A screenshot of a slightly modified StepMania 3.9 score screen. The song is Lawn Wake IV by The Flashbulb. Image courtesy of Oddweb.

Scoring in StepMania is based on judgments: each “step” (moving arrows) the player hits results in a judgment based on how far away the arrow is from the stationary arrows when the player “hits” it. The closer the arrow is to the stationary arrows when it is hit, the better the judgment. A sufficiently poor judgment (or a complete miss) results in the player’s life bar depleting.

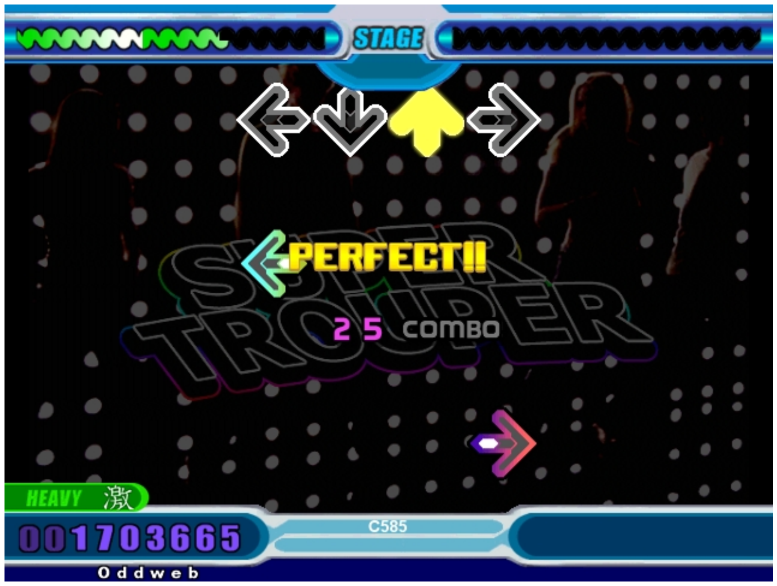


Figure 3: A screenshot of a slightly modified StepMania 3.9 in-song interface, with the player having recently scored a “perfect” (the second best judgment in unmodified StepMania 3.9). The song is Super Trouper by A-Teens. Image courtesy of Oddweb.

2.3 Glossary

The following is a list of terms used in relation to DDR/StepMania.

- **Arrows**: The arrows that scroll across to the rhythm of the song.
- **Stationary arrows**: The row of stationary shapes (or targets) the arrows approach.
- **Dance platform/mat**: A type of controller pressure sensitive areas that correspond to the arrows. See Figure 1.
- **Song**: A “song” in StepMania includes the actual song/sound track and its steps.
- **Dance pattern**: Dance pattern: the set of arrows for a given song at a given difficulty.
- **Step**: All the arrows at a given time value (as in one step: one rearrangement of the feet on the dance platform). All arrows in a step have to be pressed within the step’s time window for it to register as a hit.
- **Note**: Alternate name for “step”.
- **Double**: A type of step containing two arrows.
- **Triples (trips)**: A step containing three arrows.
- **Quadruples (quads)**: A step containing four arrows.
- **Judgment**: A “score” given for each step hit.

Part II
Description and Methodology

This part describes the video game and details its development. Procedure, setup and configuration, tools and program details are also covered.

3 Functional Description

How does the vidya work? HOW DO I PLAY VIDYA???? The vidya is controlled using the eight buttons on the STK1000. What buttons do what in the menus?

3.1 The gameplay of Scorched Land Dance Dance Defence

3.1.1 The role of the LEDs

The LEDs are used to indicate which buttons that can be pressed in each state. For example: during gameplay `LED7`, `LED5`, `LED3` and `LED2` are lit, corresponding to the buttons used to play the game.

3.1.2 Splash screen

The *splash screen* is the first screen displayed to the player. It states the name of the game and sports some flame-like effect thing. Pressing `SW3` advances the game to the *song selection menu*. Check out figure 4.

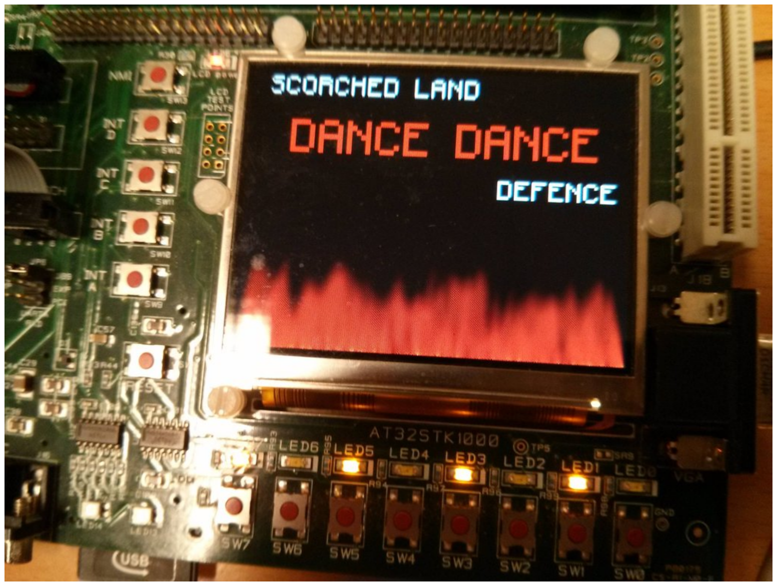


Figure 4: SLDD's splash screen.

3.1.3 Song selection menu

This is the menu from which the player selects which song he or she wishes to play. Pressing `SW3` selects the highlighted song, while pressing `SW2` traverses the list one song upwards and `SW1` traverses the list one song downwards.

In addition to `LED1`, `LED2` and `LED3` being lit when this screen is active, three icons are visible in the bottom right hand corner of the screen. These indicate the consequence of pushing the corresponding buttons.

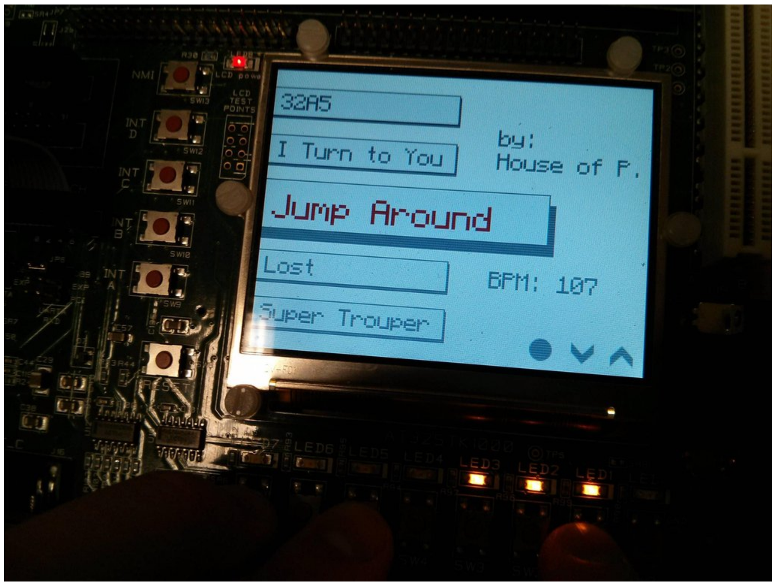


Figure 5: The song selection menu of SLDDD. The three icons in the bottom right hand corner means twice the affordance.

3.1.4 In-song gameplay

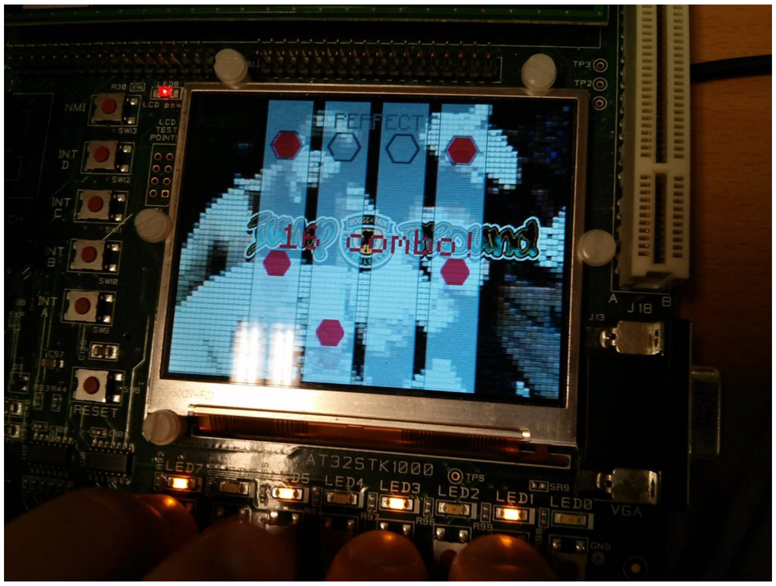


Figure 6: The in-song interface of SLDD. The song being played here is “Jump Around”.

pretty cool! The hexagons scroll upwards and

3.1.5 Score screen

Pressing `SW0` returns to the *song selection screen.*

we could have a picture of the in-game screen with circles around the different stuff and then a legend explaining what the different stuff is. Like the stationary arrows and the incoming arrows and the life bar and stuff

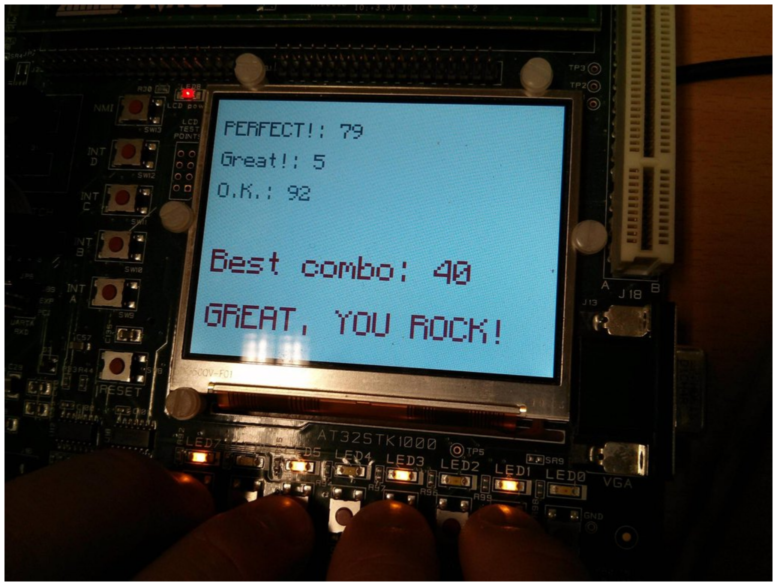


Figure 7: SLDDD's score screen. The song played was "Jump Around".

explain how judgments work and list the timings we have implemented
explain how the life bar works

4 Solution Components

This section describes the solutions' various components.

4.1 Linux

The assignment required that the Linux 2.6 kernel be compiled for the AVR32 STK1000 development board. Specifically, Linux 2.6.16.11-avr32-20060626 was used.

Linux was compiled and flashed onto a `16GB` SD Card, by virtue of following the distributed guidelines and a great deal of blood, sweat and tears[1].

4.2 Linux Device Drivers for the STK1000

Something about LDDs? Linux Device Drivers are pretty cool. They live in kernel space. Kernel space is like user space in the same way that the actions of children are like the actions of adults.

Init and Exit functions and cleanup and such

4.2.1 Loading drivers

4.2.2 LED driver

A custom driver was written for the STK1000 to facilitate the turning off and on of the LEDs. On initialization the driver obtains a major number dynamically and requests some region before enabling the relevant I/O pins by writing `0xFF` to the PIO Enable Register (`PER`) and to the Output Enable Register (`OER`).

Upon exit, the driver turns the LEDs off by writing `0xFF` to the Clear Output Data Register (`CODR`) and `0x00` to the PIO Enable Register (`PER`) to disable the I/O pins. It also releases the region it requested during initialization.

[1]there were so many tears.

The driver interacts with the LEDs by first turning all the LEDs off (by writing `0xFF` to the `CODR`) and then enabling the desired LEDs by writing the given value to the Set Output Data Register (`SODR`).

For a more detailed explanation of the set-up of the LEDs on the STK1000, see [6].

4.2.3 Button driver

A custom driver was written in order to allow the software to read the buttons' state.

The button driver allows the video game to read the state of the buttons, letting them function as a source of input for the players. Its initialization is similar to the LED driver's, except that it writes `0xFF` to the Pull-up Enable Register (`PUER`) rather than the `OER`. Its exit function is similarly different.

The driver reads the button's state by returning the value found in the Pin-Data Status Register (`PDSR`).

For a more detailed explanation of the set-up of the buttons on the STK1000, see [6].

4.2.4 Sound driver

/dev/dsp! OPEN SOUND SYSTEM -¿ http://en.wikipedia.org/wiki/Open_Sound_System

4.2.5 Display driver

The STK1000's has a screen. It has a resolution of `320×240` and sports 32 bits per pixel (24 bit color depth, the last 8 bits are unused) [13]. However, only 24 of these are enabled by default [4].

Data is displayed on the screen by writing a bitmap buffer directly to `/dev/fb0`.

4.3 Game Engine – WORKING TITLE

this is the game engine we developed. it has parts that are pretty good. like sound, it can play sound and display things on the screen.

4.3.1 Sound

how does the engine handle sound? we just pipe that to /dev/dsa right? pretty much I guess.

4.3.2 Graphics

them graphics are pretty good but how do we handle them???? everything is a bitmap, right?

4.4 "Fontenizer": the Font Engine

A font rendering engine (`game/include/font.h` and `game/src/font.c`) was developed in order to easily render some given text on the screen. It does this by iterating through a given string[2] and locating each character's corresponding glyph in a given font bitmap. The glyphs are strung together and passed to the screen buffer.

4.5 The `.SM` file format

The `.SM` was created by the developers of StepMania. It is at the heart of every "Song" in StepMania as StepMania reads these files to obtain the location of the actual audio file as well as the steps.

yo imma just rewrite http://www.stepmania.com/wiki/The_.SM_file_format

5 Configuration of the STK1000

5.1 Jumpers

The jumpers on the STK1000 were set to the positions specified on page 61 of the lab compendium [13].

[2]Actually a C string.

5.2 GPIO Connections

Two flat cables of the same type used in assignment 1 [6] and 2 [7] were used to connect the buttons (`J25`) to `GPIO 0-GPIO7` (`J1`) and the LEDs (`J15`) to `GPIO 8-15` (`J2`). In the previous exercises the LEDs were connected to `GPIO 16-31` but this was not possible in this assignment due to requirement that the jumper `SW6` is set to `MACB0/DMAC` [13] (page 21).

5.3 Audio

The audio was configured as in [7].

6 Development of the Program

drivers, game engine, .sm file format something something let's get back to this part later.

yo we made them drivers

then we made them font engine

and implemented the .sm format

then we made some graphics

We did some prototype testing and came to the conclusion that having two players control each trying to use four buttons on the board simultaneously did not result in an enjoyable experience. Our testers preferred being the sole user of the STK1000's buttons. A possible explanation for this is that all of our testers were above the age of 20 and did not posess the child-like hand size required to operate four neighbouring STK1000 buttons at the same time.

and implemented transparency

then we played dat music

and got the steps to synch up to it

6.1 Features left unimplemented

here are some features we did not have time to implement

preview of the selected song in the main menu

other step types (doubles, mines, holds) Note: SLDDD treats each individual arrow as a step.

multithreading the entire vidya is single threaded, meaning the playback of music runs in the same thread as the rest of the game. Because threading is for pussies.

automatic adjustment of note scroll speed we have to add songs manually and tailor the notes' scroll speed this is the opposite of agile cloud :(

life bar

7 Programming Environment

7.1 Allegro

`Allegro 4.2` was used to emulate the STK1000 screen in order to run and test the engine code without direct access to the STK1000. For more information about Allegro, see `https://www.allegro.cc/`

7.2 Make

See [6] for a brief description of Make.

7.3 Other tools

- `Vim` was employed as the authors' text editor of choice.
- `git` was used for version control during the development of the solution.

- The project is hosted in a private GitHub repository.
- The report was written with LaTeX.
- `oggdec` from `vorbis-tools v1.4.0` was used to convert `.ogg` files to raw audio files.
  - `http://www.rarewares.org/ogg-tools.php`
- `gparted` was used to format the SD Card before installing Linux on it.
- The `u-boot` bootloader was used to flash Linux onto the SD Card.
- `Texture Font Generator (ssc)` – a texture font generator bundled with Stepmania `5.0 beta 1a` – was used to generate font textures.

Part III
Results and Tests

8 Energy Efficiency

the solution doesn't make use of interrupts. the buttons are polled each frame. some sort of interrupt-based scheme could surely be implemented.

update() is called more often than render().

cpu usage vs redrawing on the screen (is it better to redraw large areas or just the specific areas affected between frames?)

9 Testing

9.1 Basic functionality tests

9.1.1 LED & button driver test

Description

This test is designed to assertain whether the LED and button drivers function as intended.

Prerequisites

- a driver for the STK1000's LEDs.
- a driver for the STK1000's buttons.
- a working installation of Linux running on the STK1000.

Procedure

1. Load the LED & button drivers using `insmod` and `mknod`.
2. enter `cat /dev/buttons > /dev/leds` into the console.

Expected results

When `SWn` is held down `LEDn` should light up.
$n \in [0, 7]$

9.2 Energy consumption

what about it? Right we should probably test it.

9.3 Game engine test suite

all those tests we made
all those tests we made
runnin' through my board
runnin' through my board
runnin' through my board
(runnin' through my board)
they were not enough

9.3.1 Evaluating the efficiency of functions

Select functions print their runtime/TTC (time to completion) to the console, allowing an evaluation to be made regarding their efficiency. This facilitates the evaluation of changes made to functions: if a change in the function results in an increase in its runtime, the change did not contribute to increased efficiency.

This particular technique is a kind of profiling[3] dubbed "ghetto profiling" by one of the authors.

9.4 Gameplay

we had people test the gameplay. those who did not enjoy themselves were eliminated.

9.4.1 Prototyping

rough test of gameplay two people attempted to control the STK1000's buttons at the same time, resting one finger on each button. After the conclusion of the test the participants reported that

9.4.2 External reviews

we let some peeps play the finished game. they rated it somewhere between 0 and 100. the test involved having them play the game.

10 Discussion

hey yo what up what is there to discuss? maybe we should have conducted more gameplay tests underway?

Part IV
Evaluation of Assignment

The inclusion of Scorched Land Defense as the target video game seems rather pointless as everyone was allowed to make *any* video game they wanted to. If the goal is supposed to be the creation of some specific video game or program, it should be defined in less vague terms than Scorched Land Defense was.

We were unable to flash the SD Card on the lab computers. While attempting to compile the Linux kernel we also encountered way more problems than the distributed guidelines and exercise lectures had us believe it'd be.

The "defense" in "Scorched Land Defence" is spelled inconsistently in the provided material. In the compendium it is spelled with a 's' on two occasions (in section 4) and 'c' on six occasions (in the preface and throughout section 1). In the Lab Introduction Lecture and the Exercise 3 Lecture it is spelled with a 'c'. Unsure if the two occurrances of 's' are typos or not, we have written it as "defense" roughly one out of three times throughout our work.

[3]http://en.wikipedia.org/wiki/Profiling_(computer_programming)

Part V
Conclusion

The finished video game is pretty smooth and will surely achieve a perfect score of 100/100 on Metacritic within the first twenty four hours of its release.

The authors feel they have gotten pretty good at cranking out reports and that it can be difficult to maintain energy efficiency as the number one priority when developing a video game.

Here is a video demonstration of the finished game: VIDEO LINK HERE????

References

[1] Atmel. Avr assembler user guide. `http://www.atmel.com/Images/doc1022.pdf`.

[2] Atmel. Avr32 architecture document. `http://www.atmel.com/images/doc32000.pdf`.

[3] Atmel. Avr32 at32ap7000 preliminary. `http://www.atmel.com/Images/doc32003.pdf`.

[4] Atmel. Avr32416: Avr32 ap7 linux lcd panel customization. `http://www.atmel.com/Images/doc32105.pdf`.

[5] Atmel. Power jumpers on ap7000. `http://support.atmel.no/knowledgebase/avr32studiohelp/com.atmel.avr32.tool.stk1000/html/jumper_settings.html#Power_jumpers`.

[6] Emil Taylor Bye, Sigve Sebastian Farstad, and Odd M. Trondrud. Report from lab assignment #1, tdt4258 energy efficient computer systems.

[7] Emil Taylor Bye, Sigve Sebastian Farstad, and Odd M. Trondrud. Report from lab assignment #2, tdt4258 energy efficient computer systems.

[8] Caltek/BST. Bst bs1901w manual. `http://www.docstoc.com/docs/67828383/BS1901W`.

[9] Jonathan Corbet, Allesandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers*. O'Reilly, third edition, 2005.

[10] University of Toronto Faculty of Applied Engineering. Lab reports. `http://www.engineering.utoronto.ca/Directory/students/ecp/handbook/documents/lab.htm`.

[11] Stefano Nichele. Introduction, tdt4258 energy efficient computer systems. `http://www.idi.ntnu.no/emner/tdt4258/_media/intro.pdf`.

[12] Stefano Nichele. Tutorial lecture for exercise 3. `http://www.idi.ntnu.no/emner/tdt4258/_media/ex3.pdf`.

[13] Department of Computer and NTNU Information Science. Lab assignments in tdt4258 energy efficient computer systems. `http://www.idi.ntnu.no/emner/tdt4258/_media/kompendium.pdf`.

All internet resources were checked on April 26, 2013.