CHAPTER 3
DISPLAY POWER MANAGEMENT (LCD)

Thin-Film Transistor (TFT) LCD displays are most prevalent type of displays for mobile devices today due to their best visual quality in most of the environment lighting conditions. Most of the LCD display power is consumed by the display’s backlight. This backlight-approach is inefficient as most of the energy is wasted when displaying a darker image. To reduce this inefficiency, we reduce the backlight level of the LCD screen (conserving energy), and compensate for this reduction with content enhancement methods (brighten the content) in order to maintain (to the best of our ability) the intended quality of the image being displayed. We enhance the content brightness using tone mapping techniques, which are efficient when compared to linear brightness enhancement techniques. In this chapter, first we briefly introduce LCD display technology in Section 3.1 and tone mapping technique in Section 3.2, followed by the specific tone mapping function that we use in Section 3.3 and then we describe how we build the system using this function.

3.1 LCD Display Technology

TFT LCD displays have two major components: the LCD panel and a light source, called the backlight, at the back of the LCD panel that illuminates the panel [33, 34]. The panel filters the backlight based on the values of the pixels in the display
buffer — the brighter the pixel, the more light from the backlight is allowed through. Traditional LCD displays used Cold Cathode Fluorescent Lamps (CCFL) for the backlight while modern displays use LED arrays instead.

There are three kinds of TFT LCD panels: transmissive, reflective and transflective [35]. The transmissive displays (Figure 3.1) use the backlight to illuminate the pixels and offer a wide colour gamut and high contrast. Transmissive displays provide the best visual quality under a broad range of lighting conditions; from completely dark to office lighting for example. However, they tend to be unsuitable for extremely bright environments where the ambient light overpowers the backlight.

Reflective LCDs (Figure 3.2) are unique because they operate without the use of a backlight. Instead, they rely solely on ambient light. This allows the device to consume far less power than transmissive display. Reflective displays provide best performance in bright outdoor areas, but require a frontlight to improve the viewing experience in dimly lit environments. A transflective LCD uses both transmissive
and reflective methods. It uses a backlight, similar to transmissive displays, but also adds a reflective mirror. Transflective LCDs are a compromise that allows good performance under any lighting condition.

In general, all the three types need a light source, either backlight or frontlight to illuminate the display. Almost all the power in an LCD display is consumed by the backlight or frontlight, with the filter itself requiring comparatively low power. Hence, reducing the backlight or frontlight power consumption is beneficial to all these three types of displays. In the following sections we primarily focus on reducing the power consumption of the backlight. Our methods are equally applicable for displays with frontlights.

Though other display types such as, Organic Light-Emitting Diode (OLED) display and its variants such as, Active-matrix OLED (AMOLED), Electrowetting displays are popping up in mobile world, LCDs are still prevalent today due to their best visual quality in most of the environment lighting conditions. Hence, our solution in
this chapter is optimised for TFT LCD displays. For OLED and Electrowetting display techniques our solution can be adapted by leveraging on the physical characteristics of these displays. The next chapter focuses on tone mapping techniques for OLED displays.

3.2 Tone Mapping Technique & Its Advantages

As stated earlier, a transmissive LCD display consists of a backlight that shines through a filter (TFT array substrate). This filter is controlled by the pixel values that are to be displayed and modulated by the backlight. Our solution takes advantage of the fact that the perceived brightness is due to the backlight being modulated by the filter, which in turn is controlled by the frame content to be displayed. A dark frame appears dark because the filter does not allow much of the light to shine through. However, the same effect can be achieved by using a dimmer backlight, and controlling the filter to allow more light to shine through. That is, the backlight can be dimmed by boosting the frame content to achieve the same level of perceived brightness (PB). PB is the final brightness output and is what the user finally sees on the LCD screen.

To increase the brightness of the image, the pixel values need to be increased for all pixels in the frame [32–34]. This can be done in two ways; linearly and non-linearly. There are two limitations with using a linear approach. First, saturation of pixel values will result in poor content quality (the image will look overly bright in some areas). Second, doing pixel-by-pixel transformations for every frame is compu-
tationally expensive on mobile devices and could result in less energy savings (due to high CPU costs) and lower frame rates [36] — making it impractical for games that demand high frame rates.

Hence, we instead used a non-linear Tone Mapping technique to increase the brightness of the image. In general, tone mapping is a technique used in image processing and computer graphics to map one set of colours to another. One of the common use of tone mapping is to compress the images with high dynamic range (HDR) to a lower dynamic range (LDR) so that they can be displayed or printed on devices which have limited dynamic range. Dynamic range is defined as the range of light intensities present in a scene, which is generally high in real world images. More details about tone mapping is available elsewhere [49,50,97,98]. In this research work, we use tone mapping to change the brightness levels of an image while preserving the image contrast.

Tone mapping operators are usually classified as either global (spatially uniform) or local (spatially varying). Global operators apply a single luminance transform function to every pixel in the image while local operators apply non-linear transform functions to selected pixels. As stated above, these linear global operators are simple and computationally efficient, but have difficulty effectively preserving local contrast in most HDR images. Local operators solve this problem by using a non-linear spatially varying mapping — two identical input luminance may be mapped to different output values based on properties of their local neighbourhood. As shown in Section 3.3, a local non-linear operator achieves much better quality than a linear global operator.
3.3 Using the Gamma Function for Tone Mapping

A generic tone mapping function for minimising backlight to save energy for our system is defined as follows.

**Tone Mapping Function.** Let $I$ be the Perceived display Brightness of the image, $\alpha$ be the set of pixels of the image, $BL$ be the Backlight Level and $D_{\text{max}}$ be the maximum tolerable Image Distortion. The objective of tone mapping function is to find a mapping operator for a given $I$ that maps $\alpha \rightarrow \beta$ such that, $BL$ is minimised with the distortion $\leq D_{\text{max}}$.

To lower computational overheads on mobile devices, we use a standard Tone Mapping operator called the *Inverse Gamma Function* ($\gamma$). This function is also called *Gamma Compensation* or *Gamma Correction* and is widely available in all graphics implementations (OpenGL, X11, game rendering engines, etc.) and we simply refer to it as the **Gamma** function. Gamma’s non-linearity slows down saturation heavily; thus allowing increased energy savings with minimum quality loss.

The Gamma function is already implemented in many commercial 3D games, including Quake III, and we leverage this function to change the brightness of the displayed image. In 3D games, some of the rendered objects are supposed to represent illumination objects such as, lights. The illumination effect is achieved by applying Gamma. In addition, to make game objects appear photo-realistic, Gamma is always applied during the final rendering phase [99]. Since Gamma is already being applied by 3D games in the final rendering phase, we can perform our image brightness changes, using Gamma, without incurring any additional computational cost. In
addition, Gamma is also available as a hardware-assisted function (reducing the computational cost even further) and is already a part of many modern mobile GPUs such as the Nvidia Tegra 2 [100].

The effect of changing the brightness of an image using both a linear approach and the non-linear Gamma-based approach is shown in Figure 3.3. The three images shown are the original image, the linearly brightened image and the Gamma brightened image. The linearly brightened and Gamma brightened images save the same amount of power. We also provide the histograms and colour saturation values for each image. Colour saturation is defined as the percentage of pixels where any one of the pixel’s sub components (Red, Green or Blue in RGB colour space) have been clipped (greater than 255 in 8-bit representation) due to transformations.

Figure 3.3. The Effect of Gamma and Linear Transformations. The Amount of Power Saved is the Same for Both Approaches
We observe that both the linear approach and Gamma shift the image histogram to the right (indicating increased brightness). However, Gamma retains the shape of the original histogram while the linear approach exhibits high distortion (due to high pixel colour saturation). We observed that increasing the Gamma value increases the mid-tones of an image by a large amount with a comparatively smaller increase for the extreme black and white pixels. This non-linear property makes it hard to determine, for any given image, how much we can dim the backlight to achieve the same overall perceived brightness. We explain in Section 3.4.2 how we used high precision light sensors to obtain a good relationship between Gamma and perceived brightness.

Changing the Gamma of an image also changes its global contrast as shown in Figure 3.4. We define global contrast as the standard deviation among all the pixel values in the image. A low contrast results in the image appearing “washed out” as all the pixels look similar. In particular, applying a higher Gamma value results in a higher contrast loss. Thus, we cannot brighten the image too much if we want to preserve the image quality.

In addition, we also observed that the rate at which an image loses contrast varies according to the current brightness and colour of the image. Bright or light coloured images lose contrast faster than dark images. This is shown in Figure 3.5 where the change in contrast for ten images (image brightness level 1 is the darkest image and image brightness level 10 is the brightest image) is shown. We applied a constant gamma value of 3 to all ten images shown in the Figure. Hence, even for the same Gamma value, different images will exhibit different levels of quality (contrast) loss.
To account for this variable contrast loss, we dynamically change the amount of Gamma correction applied according to the brightness level of the current game environment. If the game environment is bright (outdoors, full lighting, etc.), we apply a smaller gamma boost compared to dark game environment. We show in Section 3.4.5 how we used a user study to obtain the maximum Gamma that can be applied to different brightness levels while still retaining the quality (contrast) at acceptable levels.

### 3.4 System Design

As described in Section 3.3, we use the Gamma function to boost the image brightness. By using Gamma, which is already used in the frame rendering pipeline, we do not add any significant computational overhead. However, the non-linear nature of Gamma (affects the contrast of images in non-linear was) prevents us from just apply-
ing Gamma uniformly to all images. This naive approach would result in significant and varying user perceived quality loss.

In the rest of this section, we describe how to operationalise this approach by first studying the relationship between the backlight intensity, image brightness, and the power consumed. We then conduct an experiment to calculate the maximum possible reduction in backlight that can be achieved, by using different values of Gamma, that still preserves perceived image quality. We then describe how to calculate the amount of compensation (in terms of backlight reduction) needed when changing Gamma for a particular image. Finally we describe our complete run time adaptive system that dynamically adjusts Gamma and the backlight levels to achieve significant power savings while preserving user perceived image quality.
3.4.1 Backlight Power Measurement

First, we measure the amount of power consumed by various backlight levels by measuring the average current drawn by a 15 inch Lenovo W500 laptop with different backlight brightness levels (with negligible CPU/network load throughout the experiment) using the measurement system shown in Figure 3.14. We then repeat the experiment on the HTC Magic & HTC Hero smartphones, using the Power Tutor [101] software to measure the the backlight power values on each smartphone.

Our measurements, as described in Figure 3.6, shows a linear relationship between the backlight level and the percentage of power consumed by the backlight. It is thus possible to save significant amounts of energy by reducing the backlight level of LCD screens.

![Figure 3.6. Power vs Backlight level](image)
3.4.2 Gamma to Backlight Relationship

We now describe our methodology to determine the mapping between the backlight reduction and the Gamma-induced associated image compensation. As Gamma is a non-linear and complex function, we studied the relationship between Gamma, image brightness, and backlight levels through a series of experiments. These experiments were all conducted in a dark room using precision light sensors from Phidgets [102] which can measure human perceptible light levels ranging from 1 lux (deep twilight) all the way to 1000 lux (Overcast day). The aim of the experiment was to determine the appropriate amount of Gamma that was needed to compensate for backlight reduction under different conditions.

We determined this as follows; We first set the Gamma and backlight intensity to their default values (1 Gamma and 255 (full) backlight) and measured the perceived brightness (PB) of the screen using a light sensor placed close to the screen and pointing at it. Then, we increased the gamma by $\Delta$ and found the appropriate backlight level (BL) such that the PB was constant and similar to the base level. Equation 3.4.1 shows this functional relationship.

\[
BL = f(\gamma) : \text{PB is constant} \quad (3.4.1)
\]

For this experiment we used sample images from the Quake III game. The value of the Gamma parameter ranged from 1 to 10, where 1 indicates default frame content and higher values result in brighter content. The backlight levels ranged from 0 to 255, where 0 indicates the backlight was off and 255 was the brightest backlight level.
3.4.2.1 Analysis of Gamma to Backlight Relationship

Figure 3.7 shows the results of this experiment for various Gamma levels. We fitted an exponential curve on the plotted data and tested its fitness quality using $R^2$ regression analysis. $R^2$, called the coefficient of determination is a commonly used statistic to check the goodness of fit. $R^2$ ranges from 0 to 1 with a value close to 1 indicating a good fit. The curve in Figure 3.7 has an $R^2$ value of 0.9276, indicating an excellent fit.

From this curve, we obtained Equation 3.4.2 (generated from the functional relationship between BL and gamma shown in Equation 3.4.1) and used it in our algorithms to map from gamma values ($\gamma$) to the backlight level (BL).

$$BL = 209.33 \sqrt{1/\gamma}$$  \hspace{1cm} (3.4.2)
The equation shows the non-linear nature of Gamma as lower values of Gamma change the brightness levels of the image significantly more than higher values of Gamma. As our goal is to increase the image brightness (and thus decrease backlight levels), we use lower values of Gamma in our algorithm. In particular, small values of Gamma result in a big difference in the content brightness.

3.4.3 Measuring Image Quality

A key question in our approach is to understand the impact on image quality that our use of Gamma with backlight reduction introduces. From Figure 3.3, we have some evidence that Gamma does not impose as significant quality loss. But how do we measure this objectively?
To answer this question, we use a variety of simple and widely accepted [35] objective quality metrics used by the image processing community. We use *Mean Square Error* (MSE) and *Peak Signal to Noise Ratio* (PSNR) as our two quality metrics. MSE and PSNR compare the modified image with the original image to give the deviation in terms of the distortion and quality gain respectively.

Let $x$ be the pixel intensity array of the original image and $y$ be the pixel intensity array of the modified image. The MSE between these two images is, the mean value of $(x_i - y_i)^2$ for all pixels in $x$ and $y$. This is shown in Equation 3.4.3. Higher MSE means higher distortion in the modified image.

$$MSE(x,y) = \frac{1}{N} \sum_{i=0}^{N} (x_i - y_i)^2$$  \hspace{1cm} (3.4.3)

PSNR is derived based on MSE as shown in Equation 3.4.4. The $MAX$ value indicates the dynamic range of a pixel. For example, images with 8-bits per pixel will have $2^8 - 1 = 255$ as the dynamic range. In our computations we have normalised all the pixel values to the range 0 to 1 and hence, the dynamic range is 1 ($MAX = 1$). Higher PSNR means the quality of the modified images is comparable to the original image.

$$PSNR(dB) = 20 \log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right)$$  \hspace{1cm} (3.4.4)

Though MSE and PSNR are widely used for their simplicity, they are merely objective measurements and they ignore the attributes of *Human Visual Perception* (HVS). To account for HVS, we also use a more complex metric, called *Structural
SIMilarity Index (SSIM), which accounts for human perception and is gaining increasingly popularity among the image processing community.

The principle philosophy underlying the SSIM approach is that HVS is highly adapted to extracting structural information from visual scenes. In particular, images are highly structured and these structures are important cues about the perceived quality of an image. To account for this structural property, SSIM breaks the image into patches and compares the quality of each patch with other patches using three attributes of quality; namely luminance, contrast and structural similarity.

Let $x$ be the pixel array of the patch from the original image and $y$ be the pixel array of the patch from the modified image. The SSIM Index between these two patches, $SSIM(x,y)$ is defined in Equation 3.4.5. The index ranges from -1 to +1, with -1 representing high distortion and +1 representing low distortion (high quality). The SSIM index will be +1 when a patch is compared with itself.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$ \hspace{1cm} (3.4.5)

where,

- $\mu_x$ the average of $x$;
- $\mu_y$ the average of $y$;
- $\sigma_x^2$ the variance of $x$;
- $\sigma_y^2$ the variance of $y$;
- $\sigma_{xy}$ the covariance of $x$ and $y$;
\[
c_1 = (k_1L)^2, \quad c_2 = (k_2L)^2 \text{ two variables to stabilise the division with weak denominator;}
\]

- \( L \) the dynamic range of the pixel-values (typically, this is \( 2^{\text{no.ofbits per pixel}} - 1 \));
- \( k_1 = 0.01 \) and \( k_2 = 0.03 \) by default. These values are obtained through experiments \[103]\.

In this thesis we are interested in comparing two complete images. Hence we use the Mean SSIM (MSSIM) value, shown in Equation 3.4.6, that averages the SSIM values of all the patch comparisons. A more detailed explanation of the SSIM, including its complex mathematical derivation and improvements over existing approaches is available elsewhere \[103]\.

\[
MSSIM(x, y) = \frac{1}{N} \sum_{j=1}^{N} SSIM(x_j, y_j) \tag{3.4.6}
\]

where \( N \) is the number of patches or windows.

### 3.4.3.1 Using the Image Quality Metrics

Figure 3.8 shows an image after applying a Gamma level of two to it (Figure 3.8b) and after decreasing the backlight to compensate (Figure 3.8c). We observe that the histogram of the final backlight-compensated image is closer to the original image compared to the image that only had Gamma applied. In addition, the final image has a MSSIM score of +0.94 which is very close to a perfect +1 and better than the
middle image’s score of +0.79. The PNSR value is also in the 20 dB to 25 dB range which has been reported to be of high quality on mobile devices [104,105].

Overall, the images tell us that it is possible to achieve power savings with little quality loss and that we need to adjust the backlight not only to save power but also to improve the image quality.

3.4.4 Computing Image Brightness

As described above, our approach changes the brightness of an image using Gamma. However, first we have to measure the current brightness of the image to understand how much Gamma to apply. A naive approach would be to measure the brightness of each image pixel and then compute an average value. However, this is computationally intensive and unnecessary.

We instead estimate the image brightness through careful sampling. To do this, we select \( \approx 2000 \) samples pixels (1 out of every 20x20 pixels) from the image. For each pixel, we compute its brightness as a function of the pixel’s Red (R), Blue (B), and Green (G) colour values. This is a common calculation, shown in Equation 3.4.7, used in image processing. In general, brightness values range from 0 to 255, where lower values represent darker images and higher values represent brighter or lighter images.

\[
^{1}\text{PixelBrightness} = \sqrt{0.241xR^2 + 0.691xG^2 + 0.068xB^2} \tag{3.4.7}
\]

\(^1\text{(http://alienryderflex.com/hsp.html)}\)
We then compute a weighted average (favouring pixels towards the centre of the screen) of all the pixel brightness as the final image brightness value. This average value is then discretised into 14 levels to allow for efficient use in the rest of our computations. These 14 levels are not linear, and were selected to provide as much detail as possible for the common lighting values seen in our test games, while still covering the entire brightness range. We achieved this by using brightness levels 1 to 12 to cover the entire expected brightness ranges and reserved levels 0 and 13 to indicate values that are too dark and too bright respectively.

3.4.5 Human Calibration of Gamma Thresholds

3.4.5.1 Objective

To maximise the amount of energy conserved, we need to know the maximum amount of Gamma that can be applied to a particular image such that the resultant perceived image quality, after backlight compensation, is still acceptable to the game player. As Gamma is non-linear, the same amount of Gamma increase results in different levels of quality loss for different images. If the image is already too bright, increasing Gamma will result in a “washing-out” of the image and there will be a significant amount of loss in overall contrast. Compensation by dimming the backlight will not restore the lost contrast. We thus conducted a short user study to study the acceptable values of Gamma for each image brightness level.

3.4.5.2 Methodology and Setup

For this user study, we used four non-author Masters and six non-author PhD students from the same lab as the authors. Seven of the ten participants were Male
and they had an average age of 26 years. To run this study, we designed a gamma slider tool, which allowed the users to slide the gamma value to an acceptable level. We used Equation 3.4.2 to automatically adjust the backlight to appropriate levels when the participants changed the Gamma value using the tool 3.9. The participants were given the survey form, attached in Appendix A to fill up their most preferred Gamma values from the Gamma and Backlight adjusting tool for two quality thresholds described below.

We used the above-mentioned 14 discrete brightness levels, and selected 2 different images for each brightness level (28 images in total) from the Quake III game. The goal of the user study was to obtain an aggressive threshold for each image. The aggressive threshold is the highest Gamma level that might show quality loss but not up to the level where the images are washed-out.

We used the same experiment to also obtain the conservative threshold for each image. The conservative threshold is the highest gamma level which still results in images of comparable quality to the original image. This threshold is always lower than the aggressive threshold and preserves quality at the expense of power savings.
3.4.5.3 Results

With 10 users and 2 images per brightness level, we had a total of 20 points for each of the gamma thresholds. These points were averaged to yield the final threshold values for each image brightness level. These points are shown in Figure 3.10 for each brightness level. The image brightness levels of 0 to 13 correspond to the darkest to brightest images available in the game.

The figure shows a definite trend and that for different image brightness levels, users prefer different Gamma compensation values. In particular, as image brightness increases, the threshold values decrease. The trend lines gives $R^2 = 0.7986$ and $R^2 = 0.8911$, which indicate good quality fits for both the trend lines. From the trend we get Equations 3.4.8 and 3.4.9 for adjusting Gamma. Equation 3.4.8 corresponds to a set of aggressive values which gives the highest power saving with acceptable image and gameplay quality. Equation 3.4.9 corresponds to set of conservative values which gives the best image and gameplay quality with significant amounts of power saving.

\[
\text{Aggressive: } \gamma = -0.10 \times CBL + 4.03; \quad \text{(3.4.8)}
\]

\[
\text{where } CBL = \text{Content Brightness Level}
\]

\[
\text{Conservative: } \gamma = -0.07 \times CBL + 2.13; \quad \text{(3.4.9)}
\]

\[
\text{where } CBL = \text{Content Brightness Level}
\]
3.4.6 Objective Analysis of Gamma Thresholds

In addition to obtaining user-derived Gamma thresholds for various image brightness levels, we also analysed the correlation between our user study results and common objective metrics. We used global contrast change (loss) (GCL) as the metric to measure image quality, as it is one of most accepted quality metrics in the image processing industry [106], and quite suitable for measuring the quality loss due to Gamma correction. GCL uses the standard deviation among all the pixels in an image to compute the global contrast of an image. A change in this standard deviation indicates contrast error.

Our analysis of the conservative threshold (Equation 3.4.9) shows a minimal contrast change (less than 4 on average, on a scale of (-255 to +255)). Similarly our aggressive threshold (Equation 3.4.8) keeps the contrast change within 16 on average.

Figure 3.11 shows the effect on contrast, using the conservative thresholds, on images with different brightness levels (1 is the darkest, 10 is the brightest). This
figure, when compared to the earlier Figure 3.5 (also shown in Figure 3.11 as the dotted line), clearly demonstrates the importance of using different Gamma values for different image brightness levels. In particular, our adaptive thresholds do not suffer the same high contrast losses seen in the previous approach where a constant Gamma value was applied. We quantify the amount of power saved and the effect on user-perceived quality of both the aggressive and conservative thresholds in Sections 2.3 and 3.7.3.

3.4.7 Run-time Algorithm

Putting all the insights gained in the earlier parts of this section together, the actual run time portion of our final system, shown in Figure 3.12, can be described as follows:

- Every 250ms, our algorithm computes the brightness value of the current image (or frame). We run our algorithm every 250ms only as brightness levels in a
game tend to remain constant for this amount of time (players can’t move faster than this). We then compute an average brightness level using the brightness values of current image and the four previous images.

- If the average brightness value is the same as the previous average brightness value, the display settings are left unchanged and the algorithm finishes the run.

- If the brightness value is different, a new gamma value is calculated using either the conservative (Equation 3.4.8) or aggressive (Equation 3.4.8) thresholds according to the game player’s preferences (save as much power as possible without quality loss or save even more power with some quality loss). The backlight level is then adjusted to match the Gamma value using Equation 3.4.2.

We use an average brightness value (of the current image and the previous four images) to determine if we need to change the Gamma and backlight levels. This was intentionally done to ensure that increases and decreases of brightness happens gradually, usually one level at a time. This reduces the probability of big changes in backlight values which cause highly noticeable flickering effects.

3.5 Implementation

3.5.1 Selection of Games

As discussed in previous chapter, do not try to implement a single solutions for all type of games. As we exploit the map specific and genre specific knowledge in our algorithms, each approach is optimised to work with a specific type of games and game maps. Hence, we have selected 3 different games to implement our algorithms.
We have implemented our algorithms for network power management in Quake III [42] to represent FPS games, and the Ryzom [107] to represent MMOG games. For display power management, we used Quake III [42] and its Android port Quake III called Kwaak3 [108] to represent FPS games and Planeshift [109] to represent MMOG games.

We chose these games as, the code for these game have been open sourced and these are popular games: Quake III is a commercial multi-player game that has sold millions of copies. Its engine is still used by many many new popular games such as Urban Terror [110] (nominated for Mod DB’s Mod of the Year Award). Planeshift has been in operation since 2001 and it was opensourced from the beginning. RyZom has been in operation since 2004 and recently in 2010 July it was opensourced. Both Planeshift and Ryzom has a large loyal player-base.
3.5.2 Changing Backlight Level and Gamma

We have used the xgamma function provided by linux systems to implement our tone mapping technique for laptop displays. It allows us to set gamma in a range from 0 - 10, default gamma level being 1. It actually performs gamma correction (inverse gamma). Values greater than default 1 will increase brightness and below 1 will decrease brightness.

As there is no xgamma support in Android phones, we used OpenGL’s alpha blending technique to tone map (brighten) the content.

To adjust the backlight level we used `setpci` command, which is a standard utility for querying and configuring PCI devices. It allows us to set the backlight level to a value from 0 to 255, where, 0 is darkest (no backlight) and 255 is brightest (maximum backlight).

In Android phones, we directly write the required brightness level of the backlight to the kernel maintained file (`/sys/class/leds/lcd-backlight/brightness`) for current system brightness as shown in the following command snippet.

```
{
  echo BLevel > /sys/class/leds/lcd-backlight/brightness
  where, BLevel is the backlight value ranges from 0 to 255
```

3.6 Evaluation Methodology

In this chapter, we describe the hardware and software components as well as the process used to evaluate our system. Our test scenarios for display power saving were
designed to show the following: What is the potential power savings achievable by
our algorithm when quality of the game visuals are same as the original? We show
the results for this experiment in Section 2.3.

Due to the complexity in modelling human visual perception, we could not evalu-
ate the gameplay quality analytically. Instead, we performed an extensive user study
to evaluate the quality of modified versions of the games that use our system. The
user study methodology and setup are described below and the results are described
in Section 3.7.3.

3.6.1 Power Measurement Testbed Setup

Figure 3.13 shows the testbed used for our experiments. The game server is im-
plemented and run on a Dell Precision T7500 computer which comes with Intel(R)
Xeon(R) E5520 2.26GHz/8MB L3 Cache/12 GB RAM. We run the server in Ubuntu
10.10 platform. The game clients are connected through wired network and differ-
ent wireless networks to the game server. We used a variety of Lenovo Thinkpad
laptops (Model T61/T60/W500) and Android phones (HTC Magic/Dream/Nexus
One/HTC Desire HD) as game clients. To obtain accurate power measurements, a
high-speed multifunction Data Acquisition Equipment (DAQ) USB-6251 and a Signal
Conditioning Equipment (SC-2345) from National Instruments (NI).

To measure the current drawn by the system, we used a high-speed multifunction
Data Acquisition Equipment (DAQ) USB-6251 and a Signal Conditioning Equip-
ment (SC-2345) made by National Instrument. They are capable of getting up to
1.25 million samples of signal per second which creates highly accurate results [111].
It takes signal inputs from the Signal Conditioning Equipment and sends the cur-
Figure 3.13. Testbed

Figure 3.14. Setup for Laptop Power Measurement (Overall). [Note: Lenovo Thinkpad W500 Laptop Adapter outputs 20V DC]
rent and voltage measurements to the computer through an USB cable. We got
the graphical display results of our measurement via the software Laboratory Virtual
Instrumentation Engineering Workbench (LabVIEW for short, also a product by Na-
tional Instrument). In our experiment, we are taking in 1000 sample readings per
second for high accuracy.

3.6.2 Power Measurement - Methodology

We performed our power measurements on a Lenovo W500 laptop and two dif-
ferent Android smartphones (HTC Magic & HTC Hero). The measurements were
also done with two different games, Quake III and Planeshift. Different variants
of Quake III and Planeshift, that implemented different versions of our system, were
played throughout the measurements and the power consumed by each of the variants
was recorded.

We used five different versions of the game: static-conservative, dynamic-conservative
(the conservative thresholds shown in Equation 3.4.9), static-aggressive, dynamic-
aggressive  (the aggressive thresholds shown in Equation 3.4.8) and the default
(no power-saving) version. Static-conservative used the lowest Gamma value among
all the dynamic-conservative threshold values and static-aggressive used the highest
Gamma value among all the dynamic-aggressive threshold values. Thus the static
cases represent and test the extreme cases of our system.

We used the difference between the original version and each of the power opti-
mised versions to calculate the power saved by each version. To ensure that our power
measurements were repeatable, reliable, and could be usually compared between the
different versions, we used the same pre-recorded gameplay sessions with each game version.

We measured the power consumed on the laptop by first removing the battery and then intercepting the power intake. The current consumed (in Amps) was measured using a National Instruments Data Acquisition (DAQ) device, as shown in Figure 3.14, for a period of over one minute with the voltage set at 20V (Lenovo Thinkpad W500 Laptop Adapter outputs 20V DC). The total energy consumed was then calculated using the voltage and current values. For the smartphones, we used the Power Tutor application [101] that measured the power consumed by the mobile display over a period of time in Joules.

3.6.3 User Study - Methodology, Participants, and Setup

The user study was performed using Quake III and three different game maps that spanned the entire brightness levels. The maps used were: a darker map (q3tourney4), a brighter map (simpsons-q3) and a map with the combination of darker and brighter areas (reqbath).

We recruited a total of 60 undergraduate students from Singapore Management University (SMU) for our study. Participation was open to all students at SMU and we solicited participation through university emails. Our participants were a mix of students from technical and non-technical majors. We used Lenovo Thinkpad laptops (Model T61/T60/W500) with external mice for the user study to avoid input modality issues (playing Quake III on a smartphone is quite hard).
Each user study took about 60 minutes to complete. Before testing our system, each participant completed a short (2-3 minutes) demographics survey to determine their familiarity with networked games. The questions were:

1. Frequency of computer/mobile gaming;
2. Experience with computer/mobile games in general;
3. Experience with FPS (First Person Shooting) game;
4. Experience with Quake III game.

Table 3.1 summarises the participant demographics.

The study protocol was as follows: each participant was first given a short training session to teach them how to play the default version (no power savings) of Quake III on a non user-study map. After the training session, each participant was asked to play all four power saving variants of the game on each of the three maps. The order of maps and power saving variants were randomised for every participant. We also included the default non-power saving variants as one of the random experiments to test if the participants were able to differentiate the highest quality non-power saving baseline from the other 4 variants.

Thus, in total, each participant played five Quake III games of three minutes each for each of the three maps. To avoid player confusion, we made each participant finish all five experiments for a particular map before moving to a completely new map (although we did randomise the order in which they played the maps).

For each map, the participants were asked to play the baseline non-power saving variant before playing each of the other five versions. This was to allow them to
accurately calibrate the quality of each variant to the best possible quality for that map. As mentioned above, for each map, the five variants were presented in random order to avoid learning and ordering effects.

After testing each variant, each participant was provided a simple questionnaire with six questions (shown below) that they had to rate using a 5-point Likert scale marked with the adjectives ”Strongly-Agree”, ”Somewhat-Agree”, ”Neutral”, ”Somewhat-Disagree” and ”Strongly-Disagree”. The questions were:

1. The game WAS as playable as the baseline.

2. The colours of objects WERE differentiable and clear;

3. All the objects in the game WERE clearly viewable;

4. The darker and brighter areas in the game WERE distinguishable and clear;

5. The shadows or light-rays WERE clearly viewable;

6. The cross-hair pointer of your weapon WAS visible and clear in all areas of the game.

The complete survey form with questionnaires we have used for the user this study is attached in Appendix B.

3.7 Evaluation Results

3.7.1 Baseline Measurements

To provide a baseline for our system, we first measured the regular power consumption of both the laptop and the smartphones under different scenarios. These


Table 3.1. Demographics Statistics for the User Study

<table>
<thead>
<tr>
<th>Total Number</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>Male (34), Female(26)</td>
</tr>
<tr>
<td>Proficiency level in computer/mobile games</td>
<td>Novice (16), Average (37), Expert (6), Never played computer/mobile games(1)</td>
</tr>
<tr>
<td>The frequency of playing computer/mobile games</td>
<td>Almost every day (22), Few times in a week (18), Few times in a month(17), I never play games (3)</td>
</tr>
<tr>
<td>Played any FPS game before</td>
<td>Yes (43), No (11), Not sure about the game type(6)</td>
</tr>
<tr>
<td>Played Quake game before</td>
<td>Yes (8), No (33), Have not heard about Quake game(19)</td>
</tr>
</tbody>
</table>

baseline measurements are shown in Table 3.2 for the laptop and Table 3.3 for the HTC Magic (the HTC Hero was similar).

For the laptop, the power consumption of the LCD screen varied from 12.6% (with the CPU and network busy) to 17.7% (with the CPU and network idle). Note that the LCD power consumption is relatively small as the W500 laptop uses an Intel Core2 Duo T9600 CPU that draws significantly more power than a smartphone processor. In addition, background activities, and other components such as the RAM and HDD also consume significant amounts of energy.

For the HTC Magic, we observe that the LCD Display, as expected, consumes a far greater percentage of the total system power. In particular, the display consumes between 45% (when the CPU and network are busy) to 96% (when the CPU and network are idle) of the entire system power consumption.
Table 3.2. Baseline Power Consumption of the Laptop

<table>
<thead>
<tr>
<th>State</th>
<th>Energy consumed over 1 minute (Joules)</th>
<th>Total Energy consumed by LCD Display (Joules)</th>
<th>% Energy consumed by LCD Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Network</td>
<td>Display</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>OFF</td>
<td>BLANK</td>
<td>1936.62</td>
</tr>
<tr>
<td>MIN</td>
<td>OFF</td>
<td>FULL</td>
<td>2353.30</td>
</tr>
<tr>
<td>MIN</td>
<td>FULL</td>
<td>BLANK</td>
<td>2167.69</td>
</tr>
<tr>
<td>MIN</td>
<td>FULL</td>
<td>FULL</td>
<td>2541.52</td>
</tr>
<tr>
<td>FULL</td>
<td>OFF</td>
<td>BLANK</td>
<td>2741.95</td>
</tr>
<tr>
<td>FULL</td>
<td>OFF</td>
<td>FULL</td>
<td>3177.27</td>
</tr>
<tr>
<td>FULL</td>
<td>FULL</td>
<td>BLANK</td>
<td>2889.86</td>
</tr>
<tr>
<td>FULL</td>
<td>FULL</td>
<td>FULL</td>
<td>3307.32</td>
</tr>
</tbody>
</table>

Table 3.3. Baseline Power Consumption of the HTC Magic

<table>
<thead>
<tr>
<th>State</th>
<th>Energy over 1 minute (Joules)</th>
<th>Total Energy Consumed ( Joules)</th>
<th>% Energy Consumed By LCD Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Network (3G &amp; WiFi)</td>
<td>Display</td>
<td>CPU</td>
</tr>
<tr>
<td>MIN</td>
<td>OFF</td>
<td>FULL</td>
<td>1.5</td>
</tr>
<tr>
<td>MIN</td>
<td>FULL</td>
<td>FULL</td>
<td>1.5</td>
</tr>
<tr>
<td>FULL</td>
<td>OFF</td>
<td>FULL</td>
<td>12</td>
</tr>
<tr>
<td>FULL</td>
<td>FULL</td>
<td>FULL</td>
<td>15</td>
</tr>
</tbody>
</table>
Table 3.4. Power-Savings Measurements

<table>
<thead>
<tr>
<th>Game Variant</th>
<th>Quake III (%)</th>
<th>Planeshift (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic-aggr</td>
<td>68.26</td>
<td>67.07</td>
</tr>
<tr>
<td>Static-aggr</td>
<td>69.79</td>
<td>68.40</td>
</tr>
<tr>
<td>Dynamic-cons</td>
<td>48.95</td>
<td>47.89</td>
</tr>
<tr>
<td>Static-cons</td>
<td>20.04</td>
<td>21.73</td>
</tr>
</tbody>
</table>

3.7.2 Measured Analytical Results

In this section, we provide the power saving achieved by our system. Table 3.4 shows the results for Quake III and Planeshift. These values are the power savings achieved on the laptop. We verified, using the mobile version of Quake III called kwaak3 [108], that the power savings for Quake III were similar on the mobile phone. We were unable to run Planeshift on the Android smartphones as there was no useful mobile port of the game.

Overall, the power savings for both games were similar and allow us to draw some broader conclusions. In particular, the static-conservative variant saves the least amount of display power ($\approx 21\%$) but it also has the lowest quality loss. However, the dynamic-conservative variant is able to save $\approx 47\%$ of the display power while achieving similar quality (as shown by the user study in Section 3.7.3).

If the user is willing to tolerate some amount of quality loss, the dynamic-aggressive variant saves the same amount of display power ($\approx 68\%$) as the static-aggressive variant, but with significantly better perceived quality (Section 3.7.3).
3.7.3 User Study Results

In the previous section, we showed that we can save between 21% to 68% of the display power. However, how does this affect the perceived user quality? In this section, we describe our user study results that investigates the impact of our system (and all its variants) on user perceived quality.

The results of the user study, shown in Figure 3.15, can be summarised as follows:

- For all 6 questions, for both static and dynamic settings, the users are able to consistently differentiate the high quality (low Gamma) and lower quality (high Gamma) variants.

- As expected, the user rating for static-conservative is very close to the default non-power saving variant. However, dynamic-conservative has comparable user ratings even though it achieves significantly more power savings (47% versus 21%) than the static-conservative variant.

- The user rating of dynamic-aggressive is noticeably better than static-aggressive even though both save about the same amount of power (68%).

- The results for all 3 maps are consistent.

To check for outliers, we calculated the standard deviation among all our samples and found that they were at acceptable levels. The maximum standard deviation was 1.14 and the average standard deviation was 1.07.

3.7.4 Overall Result: System Works Very Well

We now combine the results from both the analytical power measurements (Section 2.3) and the user study (Section 3.7.3) to understand the trade-offs involved in
All values are average scores across all 3 maps. Pair-wise two-tailed t-test results were as follows: There were no significant differences between the Default and Static-cons and Dynamic-cons values. All the values for Static-aggr (for all 3 maps) were significant (at 5%) against the Default values. For Dynamic-aggr, all values for maps 2 and 3 (light map and light/dark map) were significant at 5%. Q4 and Q5 for map 1 (dark map) were significant at 5% with Q1, Q2, Q3, and Q6 being non-significant.

**Figure 3.15.** Results of the User Study for all 3 Maps. All Versions of the Game were Deemed Playable by the Participants.

saving LCD display power for mobile games. The first trade-off is that when high quality is required, it is clear that the dynamic-conservative scheme is the best variant as it saves significant amounts of display power (47%) while achieving comparable perceived quality to the more conservative static-conservative variant and the default full-quality non-power saving variant.

However, when energy efficiency is more important and the user is willing to trade-off quality for energy, then both the dynamic and static algorithms with high gamma settings can save about 68% of the display power. In terms of quality, the dynamic-aggressive variant achieves consistently better user ratings than static-aggressive and should thus be the variant that is used when higher power savings is desired. In
particular, even though users noticed quality degradation with the dynamic-aggressive variant, they still consistently rated the game as being highly playable on all 3 maps.

In summary, our adaptive (dynamic) approach, using either conservative or aggressive settings, saves substantial amount of display energy (47% to 68%) and offers excellent quality versus energy trade-offs.

3.7.5 Summary

The results show that by using non-linear tone mapping function we can achieve much higher energy saving than linearly increasing the pixel values described in the previous studies [32] [33] [34] [35]. The methods described in previous works can save up to a maximum of 40% energy. As we have used the pre-existing gamma function in contrast to new tone mapping functions described by Iranli et. al [36], our method do not incur any additional computational overhead. Hence, it is can save highest possible amount of energy (up to 68%) in mobile devices for a given quality constraint.