INTELLIGENT LIGHTING FOR GAME ENVIRONMENTS

Magy Seif El-Nasr
School of Information Science and Technology
Pennsylvania State University
magy@ist.psu.edu

ABSTRACT

Lighting design is an important topic of game development. There are many functions that lighting assumes in game environments, including directing attention, establishing good action visibility, evoking emotions, setting atmosphere, and providing depth. Current lighting design techniques rely on static manually designed lighting, where designers set up the positions, angles, and colors for each light in a level. Game environments are dynamic and unpredictable; physical and narrative scene content, including character locations, tension, and narrative goals, change unpredictably in real time due to user interaction. Thus, current static techniques often do not adequately adapt to serve desired aesthetic and communicative functions or perceptual effects. Recently, *Doom 3* incorporated dynamic real-time lighting and demonstrated many advantages of using real-time dynamic lighting in games, including heightening the emotional engagement and enhancing the overall interactive experience. However, the technique is scripted and tightly coupled to game content. In this article, we present ELE (Expressive Lighting Engine), an intelligent lighting system that automatically sets and adjusts scene lighting in real time to achieve aesthetic and communicative functions, including evoking emotions, directing visual focus, and providing visibility and depth. ELE operates as a separate system that interacts with game/graphics engines through a standard interface. In this article, we will discuss ELE and its interface with *Unreal Tournament 2003*. We will also present results showing ELE in action. These results show:

- Utility of real-time adaptive lighting in providing visual focus, setting atmosphere, evoking emotions, and establishing visibility during interaction in interactive environments
- Acceleration in the development process due to the introduction of an automatic system for lighting that can be overridden by designers at a high level, thus eliminating the time-consuming process of setting individual light parameters for each level and scene.

INTRODUCTION

Visual composition, including light placement, angles, colors, camera angles, field of view, movement, and environment textures, have an important impact on how game environments are perceived by players. Film, theater, animation, and game designers have recognized the role that visual composition plays and its impact on scene communication and perception [Alton 1995]; [Arijon 1976]; [Birn 2000]; [Bjorke 2004]; [Block 2001]; [Bordwell and Thompson 2001]; [Brown 1996]; [Carson 2000]; [Maattaa, 2002]. At GDC (*Game Developers Conference*), Will Wright identified several functions that visual composition assumes in game environments, including directing a player's focus to important elements in a game by balancing saturation, brightness, and hue of objects in a level [Wright 2004]. This is important to identify and acknowledge because it is a design element that affects game play and emotional engagement [Carson 2000; Maattaa 2002]. In this article, we will focus on one aspect of visual composition, specifically lighting.

Designers have identified several visual design functions or perceptual goals for lighting design. These goals include establishing visibility for important elements in the scene, directing viewer's attention to important scene elements (visual focus), establishing depth, evoking moods, and setting atmosphere, as well as providing information, such as the time of day and environment setting [Alton 1995]; [Birn 2000]; [Bjorke 2004]; [Block 2001]; [Bordwell and Thompson 2001]; [Brazel 1997]; [Brown 1996]; [Calahan 1996]; [Gillette 1998]. Lighting designers must satisfy these goals while maintaining visual continuity, which is important to maintain suspension of disbelief [Brown 1996]; [Calahan 1996].

These are important goals that affect game play at different levels. For example, setting the right mood and atmosphere is important to create the emotional involvement needed for a game [Calahan 1996]; [Carson 2000]. In addition, we have conducted several experiments where naïve users (i.e., users who hadn't played first-person shooter games before) were asked to play *Unreal Tournament* while wearing an eye tracker. The results show that due to poor lighting design (in this case, lighting that did not support good visual direction), subjects were unable to spot enemies fast enough, and therefore they died very quickly. This caused frustration and, in most cases, forced players to quit the game early.

Recognizing the importance of lighting, game designers often devote much time and a complete lighting team (in some cases) to configure lighting for a game. A lighting team typically spends days lighting a level by statically defining light positions, angles, and colors to achieve the desired visual design goals, including setting atmosphere and providing necessary visibility. This procedure is problematic, because lighting design depends on design parameters, such as character locations, object locations, and dramatic tension, which change unpredictably during interaction. Due to this unpredictability, static lighting design can result in experiences with several problems, including distracting colors, unlit characters, and insufficiently lit scenes or levels. For instance, in *Prince of Persia 2* (demo), at one instance in the game the participant is expected to jump on a specific ledge; however, due to poor lighting the ledge is not sufficiently visible, leading many users to experience frustration and disengagement (this has been reported by several participants who played the demo). In *Resident Evil 3*, there is a specific corridor in one of the levels that was lit with saturated colors to signify danger. The user is expected to pass through this corridor twice, once casually walking through and the other time being chased by a monster. The chosen colors worked very well for the latter condition, but were confusing in the first. These problems arise because static lighting is inflexible and context insensitive.

To alleviate these problems, we have developed a dynamic real-time lighting system called ELE (Expressive Lighting Engine). Given the current situation, ELE automatically selects the best lighting configuration that satisfies the desired perceptual goals, including establishing necessary visibility for game action, creating depth, directing player's attention to important scene elements, and evoking moods. This alleviates the problems faced by static lighting designs, and also provides a method for accelerating the lighting design process by allowing designers to direct the lighting system at a high level by adjusting high-level lighting constraints and goals (as will be discussed later) instead of requiring them to set low-level details of lighting parameters: positions, angles, and colors, for each level and situation in a game.

ELE has been interfaced with *Unreal Tournament 2003 (UT2003)* and *Wildtangent*. ELE is designed as a separate system that interacts with the game/graphics engine through a standard interface. This is important to decrease the coupling between the game engine and the lighting system. It also presents a more generic system that can be integrated with many games, and thus is not tightly coupled to game content.

This article first discusses the current methods used by the game industry. Then it discusses ELE, outlining its architecture and interface with *UT2003* and *Wildtangent*. Following this discussion, we will discuss some results showing ELE in action. Since ELE is composed of several subsystems, we will first show results illustrating these subsystems in isolation; then we will show results illustrating the utility of ELE in game environments. We will follow with a discussion outlining the main contributions of ELE and benefits of dynamic real-time lighting. Note that much work still remains in this area, and thus in conclusion, we will present limitations of the current implementation and future research directions.

Current Lighting Design Techniques in Games

The game industry has developed several approaches to lighting design. These approaches borrow directly from techniques used by animators [Birn 2000]; [Calahan 1996]. We identify two main methods for lighting used in games: ambient lighting used in the *Sims* and *Sim City*, and manual static lighting used in first-person shooter, adventure, and horror games, e.g., *Half Life*, *Unreal Tournament*, and *Silent Hill*. It should be noted that to portray shadows and add aesthetic quality to the experience, designers often pre-render or paint shadows in texture maps or player sprites. Also note that few games have augmented a statically lit environment with a real-time dynamic lighting approach, e.g., *Blade of Darkness* and *Doom 3*. While these examples provide a great example of the use of dynamic lighting in games, they still suffer several limitations. In this section, we will outline these methods and discuss their limitations.

Ambient Lighting Design

Ambient lighting is a method where objects in a scene are given constant luminance values [Moller and Haines 1999]. In general, ambient lighting presents a fast and simple lighting model. It ensures that all objects are visible. This type of lighting has been used in interactive entertainment productions such as the *Sims* and *Sim City*. Although the technique supplies the desired look and feel for games, such as the *Sims* or *Sim City*, it cannot be generalized for use in first person shooters, horror, or action games, where realistic and dramatic lighting is often more appropriate.

Manual Static Lighting Design

The majority of game lighting designers use static lighting designs, where they carefully plan and predefine the lighting in a scene [Carson 2000]; [Maattaa 2002]. They manually set lighting properties, including position, angles, and color, for each individual light in a scene. They then use a global illumination technique (e.g., radiosity algorithm) to render the level resulting in a light map that is then applied to the level. For aesthetic and atmospheric quality, some designers also use a combination of shadows and colors painted into texture maps.

While a manual static lighting design approach provides a successful method for linear media, e.g., film and theater, it does not provide an adequate solution for interactive environments. Unlike film and theater, interactive 3D environments are unpredictable due to user interaction. Design parameters, including character/object locations and their importance, change unpredictably at runtime. For example, character relationships change depending on users' behaviors and actions, and physical positions of characters change relative to users' actions. Therefore, designers cannot script lighting behaviors to accommodate all situations and perform desired perceptual effects. Hence, current manual designs often result in undesirable effects.

Most games, including *Max Payne*, *Resident Evil*, and *Prince of Persia*, use strategically placed lights that create a realistic feel by effectively motivating the lighting angles and positions by the major practical sources that exist in the scene, e.g. torches or windows. This kind of lighting, however, seldomly portrays the correct visual focus for the scene. In fast paced games, directing viewers' eyes (especially for inexperienced users) to the focus is crucial for engagement. We will discuss some qualitative experimental results that validate this claim in the results section. Additionally, in many adventure games, lighting can be employed to visually guide the player through a process or a task by directing his attention to important areas in the scene.

Dynamic Lighting

Dynamic lighting is still not evident in many games. *Doom 3* is one of the very first games to extensively use dynamic real-time lighting to enhance emotional involvement and immersion. In *Doom 3*, designers interplayed among several parameters, including position, orientation, saturation, lightness, and color warmth. An example of a lighting pattern that designers used in *Doom 3* was the sudden shift in lightness values from approximately 80% to 0%, and the increase in the affinity of color warmth in certain scenes when specific monsters appear. These changes increase visual tension and heighten emotional engagement.

However, these changes were mostly scripted by the designer and must be redesigned for each game. This leads to a design that suffers several limitations. First, scripting dynamic lighting patterns is a difficult problem, because designers must account for changes in character, object, and user positions and orientations, as well as lighting effects and colors used in previous frames. This usually generates a greater search space than what is currently accounted for. To manage this space, *Doom 3* designers limited the dynamic lighting patterns to a small set, which resulted in many repetitive effects. This problem is analogous to the well-documented sound design problem where limiting the sound effects and their transition leads to more repetitive patterns [Fay et al. 2004]. In addition, due to the large search space, scripting such dynamic lighting patterns is a daunting and time consuming task.

Intelligent Lighting—An Automatic System for Configuring and Dynamically Modulating Game Lighting

As an alternative approach, we have developed an automatic lighting design system, called ELE (expressive Lighting Engine), that configures and continuously modulates the lighting in a scene during interaction to enhance players' experiences. For that purpose, ELE is designed based on cinematic theories. ELE automatically, in real time, selects and modulates the lighting configuration, including the number of lights, their positions, colors, and angles, satisfying several visual design (or perceptual) goals:

- Directing visual attention to important objects
- Establishing depth
- Accommodating necessary visibility for the characters and action
- Evoking moods and setting atmosphere
- Paralleling dramatic tension of the experience over time
- Maintaining visual continuity

These visual design goals conflict with one another. For example, choosing a lighting configuration that establishes a particular atmosphere or mood may hinder visibility. Cinematic lighting designers formulate a lighting configuration by setting and varying priorities of the visual design goals over time [Brown 1996]; [Gillette 1998]; [Millerson 1991].

To accomplish this task, ELE uses constraint non-linear optimization to select the best lighting configuration that achieves the most important and desired visual design goals given their priorities and the current situation. Using cinematic rules discussed in [Brown 1996]; [Gillette 1998]; [Millerson 1991], we formulated an evaluation function for each visual design goal that evaluates the degree of visual design goal satisfaction given a lighting configuration, *p*. The overall cost function that ELE minimizes can be summarized in the following form (note: this equation is used here as an abstract representation to clarify ELE's overall function; later, we will discuss instantiations of this function used by ELE's algorithms):

$$\begin{aligned}&\lambda_v V(p) + \lambda_d D(p) + \lambda_m M(p) + \lambda_a A(p,o) + \lambda_t \lfloor T(p) - I_t \rfloor + \\ &\lambda_c C(p, p_-) + \lambda_l \left[\min_i \left\| k - l_i \right\| \right] + \sum_c \lambda_i \left[CI(p,c) - II_{c,s} \right] + K(p, p_-),\end{aligned} \qquad (2.1)$$

where p is a lighting configuration. Each term in this equation deals with a specific visual design goal, identified earlier. In the next paragraphs, we will define the variables and functions in the Equation 2.1 as well as the significance of each term and its utility.

λ_v is cost or importance for establishing visibility given the current situation; V(p) is a function that evaluates visibility of the scene given p. Visibility is important for players; lack of visibility can cause frustration and disengagement, as described earlier .

λ_d is cost or importance of establishing depth given the current situation; D(p) is a function that evaluates depth within the scene given p. Depth is an important perceptual element that cinematic lighting designers always strive to achieve [Viera 1993].

λ_m is cost or importance for establishing modeling given the current situation; M(p) is a function that evaluates modeling within the scene given p. Modeling is a term used by lighting designers to show character depth [Millerson 1991]. Modeling is often established by setting more lights at specific angles relative to the character [Rangaswamy, Sudeep. 2000]. This is important to separate characters from the background and also to provide depth [Alton 1995]; [Birn 2000]; [Viera 1993].

λ_A is cost or importance for establishing visual attention given the current situation; A(p, o) is a function that evaluates the desired attention given the desired focus object o and the lighting configuration p. Lighting designers typically use lighting to draw the audience's attention to a particular focus point in a scene [Kidd 2001]; [Lowell 1992].

λ_t is the cost or importance of establishing tension given the current situation; T(p) is a function that evaluates tension given p. I_t is the ideal tension value at the given time t. Tension is a fuzzy concept that is not well-defined in cinematic theory. Block, however, describes several techniques that designers use to achieve tension [Block 2001]. He describes techniques based on affinity of saturation, warmth, or brightness, and contrast of saturation, warmth, or brightness within one scene or between scenes over time.

λ_c is cost or importance for establishing visual continuity given the current situation; C(p) is a function that evaluates visual continuity of the scene given p and the previous setup p. This term is important to maintain suspension of disbelief during the experience.

k is the key light angle with respect to the camera; $\min_i \left| k - l_i \right|$ selects the best light source l_i for motivating a key light angle given the camera viewing angle. This establishes realistic lighting motivation for the scene by confirming to the existing practical sources [Birn 2000]; [Gillette 1998]. This is especially important for achieving realistic setting.

λ_i is the cost or importance of establishing a particular character mood or intention using lighting; CI(p, c) is a function that evaluates character intention for character c given the lighting configuration p; II_s is the ideal intention for character c given situation s. Figure 2.4 shows an example use of lighting to signify character intention and relationship. II_s is defined symbolically using cinematic terms: e.g., mysterious, evil, sinister, etc. [Campbell 1999]. This is then converted into numeric values for angles or colors as will be discussed later.

The last term in the equation defines specific color constraints established by a designer in terms of saturation, lightness, and warmth of particular areas and their change over time in the scene. This is important to allow designer control over color choices and styles.

As can be seen, selecting a lighting configuration using this equation involves establishing evaluation functions that evaluate a setup in the three different spaces: layout, angles, and colors. Since this requires search in a rather large search space, we divided the optimization problem into three relatively independent sub-problems: allocation, angle selection, and color selection.

ELE is then composed of three subsystems, as shown in Figure 2.1:

- A lighting allocation subsystem selects the best number of lights and placements
- An angle subsystem selects the best angles for each light
- A color subsystem selects the best color for each light

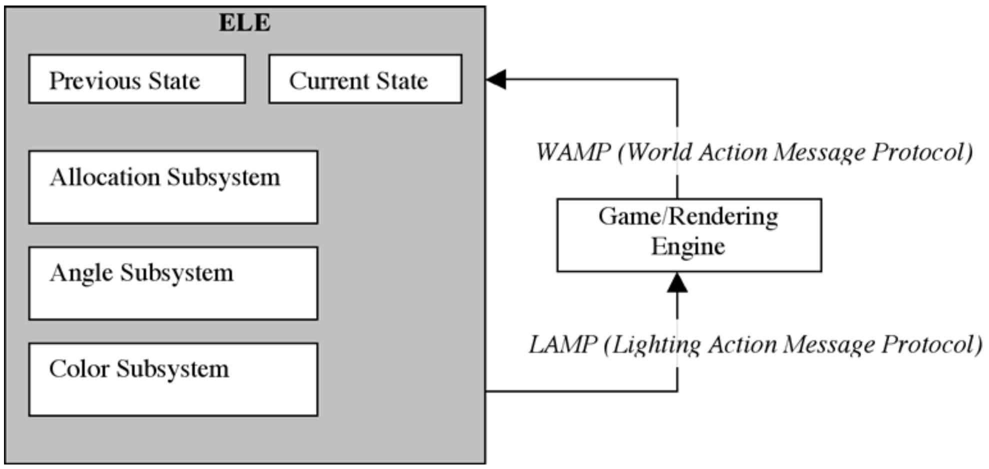


FIGURE 2.1 ELE's architecture.

Each of these subsystems uses the equation defined (or a subset of it) to guide its own optimization process. The resulting lighting configuration is then communicated to the game/graphics engine.

Before discussing these subsystems in detail, we will first discuss the interface defining the communication between ELE and the game/graphics engine. We have designed ELE to enable easy plug-in into game and graphics engines. This was achieved by setting a well-documented and defined interface by which ELE communicates a lighting configuration to a game engine, called LAMP (*Lighting Action Message Protocol*)[1] and another protocol by which the game engine supplies information to ELE, called WAMP (*World Action Message Protocol*). This architectural composition is similar in design to the architecture described in [Young et al. 2003].

[1] The word *protocol* is used here to refer to a defined message interface used by two or more processes communicating within the same computer or between computers over the network. This should not be confused with network protocols, such TCP/IP.

LAMP (Lighting Action Message Protocol)

This is an XML-based message protocol identifying several parameters:

- Light-area identifier
- Type (key, fill, or backlight)
- Type of instrument(spotlight, directional light, or point light)
- Color in RGB color space
- Attenuation
- Orientation including facing and up vectors
- Range
- Masking parameters
- Position in 3D space
- Penumbra and Umbra angles (depends on the light instrument used)

This message protocol is sent to the game engine in XML format. The message is sent every *x* milliseconds, where *x* varies depending on the application and pace.

WAMP (World Action Message Protocol)

Several game parameters are needed to provide good lighting that enhances and supports the experience. These parameters are:

- Focus object/character/area
- Local light sources in the level
- Characters' Information
    - Number of characters
    - Name of each character
    - Location of each character
    - Orientation of each character with respect to the camera
    - Character actions (e.g., walking, running, etc.)
    - For each character: desired intention to be portrayed at a given moment within the scene
- Object locations
- Stage configuration: height, width, and length of the visible room
- Practical light sources within the room
- Dramatic tension level of the moment within the experience

Some of these parameters are only sent during zone or room initialization, i.e., when the camera or player moves into a new room, others are sent regularly every *x* milliseconds, where *x* varies depending on the application and pace. For example, the stage configuration is sent only when the camera or player moves to a new room, while the locations of characters are communicated every *x* milliseconds. Thus, we identify two different types of message protocols that we group under WAMP:

1. Stage configuration and local light sources in the scene are sent every time the player or camera enters or exits a new room or zone
2. Character locations, names, orientations, object locations and orientations, dramatic tension level, visual focus, and character intention portrayal are sent every *x* ms. (in XML format)

Some of these parameters are also computed by ELE automatically, if none was supplied by the engine. ELE uses a default method for computing these parameters. For example, to determine where to direct viewers' attention, ELE uses the number of characters in the frame and the dramatic importance of their actions computed using simple rules, e.g., walking characters are more dramatically important than idle characters.

State (Current and Previous)

ELE also keeps internally a copy of the previous and current states, which includes the lighting configuration computed and communicated world information. This information is used by the evaluation functions discussed earlier. Perhaps the most obvious example is evaluating visual continuity, which requires both previous and current lighting settings.

Dynamic Light Allocation Subsystem

Dynamic lights are a scarce resource and need to be allocated and managed efficiently to comply with the requirements of lighting design discussed while achieving real-time rendering speed. Rendering time is proportional to the number of lights used. Thus, to achieve real-time rendering, most rendering engines limit the number of dynamic lights in a scene, e.g., *Wildtangent* (a publicly available rendering engine) restricts the number of dynamic lights to eight. On the other hand, lighting designers use many lights to gain finer control of the different areas in the scene and provide modeling and depth. Lighting designers at Pixar, for example, use 8 lights or more to light one character and 32 lights or more to light a complete scene [Rangaswamy, Sudeep 2000].

This problem is not new to game design; game designers often have to accommodate different methods for handling dynamic resource allocation, including CPU power, number of lights, and audio effects. Game designers often use scripted rules to shift resources, e.g., dropping background music or sound to emphasize a line of dialogue or an explosion. Accommodating lights in an interactive scene dynamically, however, is a harder problem, due to the number of constraints, including the necessity for visual continuity, visual focus, and action visibility.

To tackle this problem, ELE uses optimization to balance constraints and goals. ELE allocates lights to zones,[2] since allocating lights to a complete level is a waste of resources. For each zone

[2] This is different from the game industry's definition of a *zone*. Here we define zone to be a particular room within the level that is visible given the camera position and angle.

enter or exit event, ELE reallocates the lights. Selecting best light allocation for a zone requires addressing several perceptual goals, including visual focus, visibility, character modeling, depth, and visual continuity.

To direct the viewer's attention to the scene's visual focus and to provide character modeling, ELE divides the zone into several areas, *A*, depending on the maximum number of lights that can be used, the number of characters in the scene, and the visual focus. ELE creates these areas to cover the background, foreground, and characters within the visible region. Using a theatrical lighting allocation technique, ELE divides the stage into a number of overlapping areas (called acting areas), where the overlap region is set to a constant *o* [Gillette 1998]. ELE then uses a greedy algorithm to create areas for characters (called character areas), such that all characters are assigned to an area, as follows:

```
Step 1: For each character $c$ create a new area and assign c to it.
Step 2: Repeat.
    For each area $a$
        if $\exists\, a'$ s.t. $\|a-a'\| < \varepsilon$, and both are focus areas (or non-focus)
        then merge $a, a'$
    Each area $a$ is lit within a cylinder cyl($a$) with center, radius, and height given by:
```

$$\text{center}(\text{cyl}(a)) = \text{center}(\text{bbox}(a)), \tag{2.2}$$

$$r(\text{cyl}(a)) = \|\text{bbox}(a)\|_\infty + \max_{\text{character } c \in a} \|c\|_\infty + s, \tag{2.3}$$

$$h(\text{cyl}(a)) = \max_{\text{object } x \in \text{bbox}(a)} h(x), \tag{2.4}$$

where bbox(*a*) is the bounding box of all characters in area *a*; h(*y*) is the height of some object *y*; and *s* ("slop") is a constant. The notation $\|y\|_\infty$ is used to denote the maximum dimension of object *y*.

After creating these areas, ELE then allocates lights to each area depending on the level of visibility, modeling, and depth needed. To achieve this, ELE sets a maximum limit on the number of lights that can be assigned to each area. Non-character areas are assigned a maximum of one light. Visible character areas, however, are assigned a maximum of five lights, since character areas may require finer control to establish depth and modeling.

Spotlights are used for character and acting areas. On the other hand, the type of light used for the background area depends on existing practical sources. For example, point lights are used to simulate torchlight or candlelight, while spotlights or directional lights are used to simulate the effects of sunlight projected from a window or a door.

Equation 2.1 summarized the overall cost function for selecting a lighting configuration. In particular, the allocation subsystem is concerned with selecting an allocation or layout that establishes good visibility, depth, and modeling, while maintaining visual continuity. Therefore, the cost function used for this particular subsystem is:

$$\lambda_v V(p_t) + \lambda_d D(p_t) + \lambda_m M(p_t) + \lambda_c C(p_t, p_{t-1}). \tag{2.5}$$

Where *p* here is the light allocation rather than the entire light setup as in Equation 2.1, visibility, V(*p*), of a light allocation is defined as the percentage of visible areas that are assigned lights by *p*, or

$$V(p) = \frac{\left|\{a \in A \,|\, p^{-1}(a) \neq \varnothing\}\right|}{|A|}. \tag{2.6}$$

Modeling, in terms of allocation, is defined as the average number of lights assigned to character areas, or

$$M(p) = \frac{\sum_{a \in A_{character}} \left|p^{-1}(a)\right|}{\left|A_{character}\right|}. \tag{2.7}$$

The depth, D(*p*), of a light allocation *p* is the difference between the number of lights assigned to the background and foreground areas, or

$$D(p) = \sum_{a \in A_{background}} \left|p^{-1}(a)\right| - \sum_{a \notin A_{background}} \left|p^{-1}(a)\right|. \tag{2.8}$$

The visual continuity, C(p_t,p_{t-1}), of a light allocation *p* is defined as the difference between the configuration being evaluated and the one used in the previous frame:

$$C(p_t, p_{t-1}) = \frac{1}{|L|} \sum_{a \in A} \left| \left|p_t^{-1}(a)\right| - \left|p_{t-1}^{-1}(a)\right| \right| \tag{2.9}$$

Hence, given the information conveyed in WAMP: the local light sources, the stage configuration and dimensions, the importance of various lighting goals (set by the designer), and the formulas described, ELE uses the cost function (Equation 2.5) to find the best lighting allocation. We formulated a greedy algorithm that allocates lights to each visible area in the scene, as follows:

1. Assign each area the maximum number of lights it can have
2. Remove one light that will incur the smallest loss, where loss is measured by the cost given Equation 2.5
3. Repeat Step 2 until the number of lights assigned is less than or equal to the maximum

Angle Subsystem

ELE selects an angle for each light in the scene. In this section, we discuss the method by which ELE selects angles for lights allocated to character areas including key, fill, and backlight angles. The same techniques are used to calculate angles for lights allocated to other areas, and thus will not be repeated.

Similar to the light allocation subsystem, the angle subsystem selects an angle for each key[3] light that best establishes the visual design goals desired. Specifically, it uses optimization to select an angle for each key light in the scene to ensure visual continuity, maintain the illusion of a practical source, provide appropriate character intention, and ensure that all characters are visible. By providing appropriate character intention, we mean the use of backlighting, under-lighting, or side-lighting to portray different character characteristics, such as mysterious, sinister, etc. [Campbell 1999].

ELE selects the best angle to minimize the following cost function (instantiated from the overall cost function (Equation 2.1)):

$$\lambda_v V(k,s) + \lambda_c C(k,k_-) + \lambda_l \left[\min_i \left| k - l_i \right|\right] + \sum_c \lambda_i \left[CI(p,c) - II_{c,s}\right], \qquad (2.10)$$

where λ_v is the cost for visibility (as stated above), $V(k, s)$ is a function that evaluates visibility in terms of lighting angle relative to the character given the camera orientation. λ_c is the cost for visual continuity, and C(k, k_-) is the difference between current key light k and previous key light angle k_- . As stated, the third term selects the best practical source l_i for motivating a key light angle given the camera viewing angle.

For each character c, λ_i is the cost of deviating from desired character intention, CI(p, c) is a function that returns a perceived intention given a character and an angle, and $II_{c,s}$ is an ideal intention value for a given character c and state s. These intention values are represented symbolically, e.g., mysterious or sinister. For this purpose, we defined several character characteristics and their corresponding lighting angles based on cinematic and theatrical conventions [Campbell 1999]. For example, backlit characters are perceived as mysterious, and under lit characters are perceived as sinister [Campbell 1999].

Based on Millerson's [Millerson 1991] documented rules, we formulated the following equation to evaluate the visibility and modeling of a given key light azimuth angle relative to the camera k, and subject to key light angle s (Figure 2.2):

can't see equation (2.11)

Millerson recommends an elevation angle between π/6 and π/3 (in radians) [Millerson 1991], unless an evaluation angle is already established given the characters' desired characteristic, e.g., under lighting a character to show him as sinister.

ELE uses a non-linear optimization system based on hill climbing to select an angle for each key light that minimizes the cost function as shown. ELE uses rules based on Millerson's [Millerson 1991] guidelines to select, fill, and backlight azimuth angles depending on the value of the key light angle. According to Millerson's guidelines [Millerson 1991], fill light azimuth and elevation angles are calculated to be the mirror image of the key light angle. We define backlight azimuth angle as:

$$b = (k + \pi) \bmod 2\pi. \qquad (2.12)$$

[3] The main source of light that establishes direction and shadows.

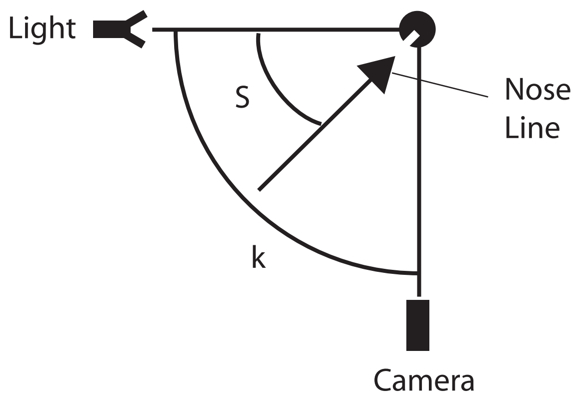


FIGURE 2.2 Angles between subject, camera, and light.

Color Subsystem

The color of lights in a scene shapes the feel of the entire image. Game designers often use contrast to create mood; examples can be seen in many games including *Silent Hill*, *Resident Evil*, the *Thief* series, and *Splinter Cell*. In these games, however, color parameters are set by a level designer, and do not continually change over time.

Our goal is to use ELE to automatically, in real time, adapt the lighting to the user's interaction, while allowing designers to write high-level rules setting constraints, including color constraints and desired lighting style. This is beneficial to designers, not only because it provides an easier, faster, and more design or art-oriented approach to lighting, but it is also a more appropriate design method, especially assuming non-linear or unpredictable environments, where narrative details, lighting conditions, and camera properties are not known and cannot be predicted at design time.

Designing such an automatic, adaptive color subsystem is difficult, since changing the color of one light may affect the entire image. Such a change affects not only visual tension, but visual continuity, visual focus, and visibility goals as well. So ELE manipulates colors using an optimization algorithm that searches for the best color that creates the desired effect.

ELE evaluates color using several parameters, including desired visual tension and its importance, importance of visual focus, the desired visual focus, importance of portraying depth, and importance of visual continuity. Using these parameters, ELE uses non-linear optimization to search through a nine-dimensional space of RGB values. It differentiates among focus colors, non-focus colors, and background areas to select a color for each individual light in the scene. Following Equation 2.1, ELE evaluates a color *c* by using a multi-objective cost function, where each objective evaluates the color assigned to each area type against the visual design goals, including establishing depth, conforming to color style and constraints, paralleling dramatic tension, adhering to desired hue, saturation, and lightness, and maintaining visual continuity. The equation for this subsystem is defined as follows:

$$\lambda_d D(c)+\lambda_a A(c,o)+\lambda_t \lfloor T(c)-I_t \rfloor+\lambda_c VC(c,c_-)+K(c,c_-) \tag{2.13}$$

where depth, $D(c)$, of a color vector **c** is defined as the color difference between colors lighting the background areas and those lighting other areas, formulated as follows:

can't see equation (2.14)

where B are the indices for background lights, NB are the indices for non-background lights, and E is the color difference defined.

The last term of the equation denotes a function for conforming to given artistic constraints in terms of warmth, saturation, and brightness for lights in the scene. This formula is defined as follows:

can't see equation (2.15)

where c is a vector of light colors for focus f, non-focus n, and background b, and areas at frame t. Color c_i is represented in RGB color space; S(c) denotes the saturation of color c; H(c) denotes the hue of color c; and L(c) denotes lightness of color c (in RGB color space). Color c_i represents the color vector for area i used in the previous state.

ELE uses CIEDE2000, a well-known formula for measuring color difference [Hill, Roger, and Vorhagen 1997]; [Luo, Cul, and Rigg 2000] to evaluate visual continuity, defined by function VC in Equation 2.13 and depth defined in Equation 2.14:

$$E=\sqrt{\left(\frac{\Delta L}{k_L S_L}\right)^2+\left(\frac{\Delta C}{k_C S_C}\right)^2+\left(\frac{\Delta H}{k_H S_H}\right)^2+\Delta R}, \tag{2.16}$$

where $\Delta R = R_T f(\Delta C \Delta H)$ and ΔL, ΔC, and ΔH are CIELAB metric lightness, chroma, and hue differences respectively; S_L, S_C, S_H are weighting functions for the lightness, chroma, and hue components; and k_L, k_C, k_H are parameters to be adjusted depending on model material information.

Following traditional lighting design theory, we used contrast to establish visual tension paralleling the dramatic tension in the scene [Block 2001]. Using the guidelines documented by filmmakers and designers, contrast is defined relative to the focus of the scene [Block 2001], which we define as the difference between colors lighting the focus area and a weighted sum of the colors lighting the other areas. Thus, T(c) is defined as follows:

can't see equation (2.17)

where ϕ is the color component (lightness, warmth, or saturation) over which we compute contrast (which largely depends on the style of the scene); c is a vector of light colors; c_i is a color for an area type i, where $i \in \{focus, non\text{-}focus, background\}$; and *focus* is the index of the visual focus area. Following this definition, A(c, o) is defined in terms of contrast, where c_{focus} represents o in this equation.

In order to evaluate contrast in terms of warmth, a function for measuring color warmth is required. Warmth is an elusive perceptual quality. There are no formulas defining color warmth in RGB color space. Based on the results collected by Katra and Wooten described in [Katra and Wooten 1995], we used a multiple, linear regression method to formulate color warmth in RGB color space, as follows:

can't see equation (2.18)

To search for the best color values for each light in the scene, we used gradient descent. Although gradient decent has major drawbacks, including occurrence of oscillations and being easily stuck in a local minimum, ELE uses gradient descent for several reasons. First, it provides a fast and simple solution. Second, a local minimum in this case is preferable because it provides a solution closer to the older one, thus ensuring visual continuity. Third, alternative methods rely on the existence of a second derivative, which is not necessarily true in this case.

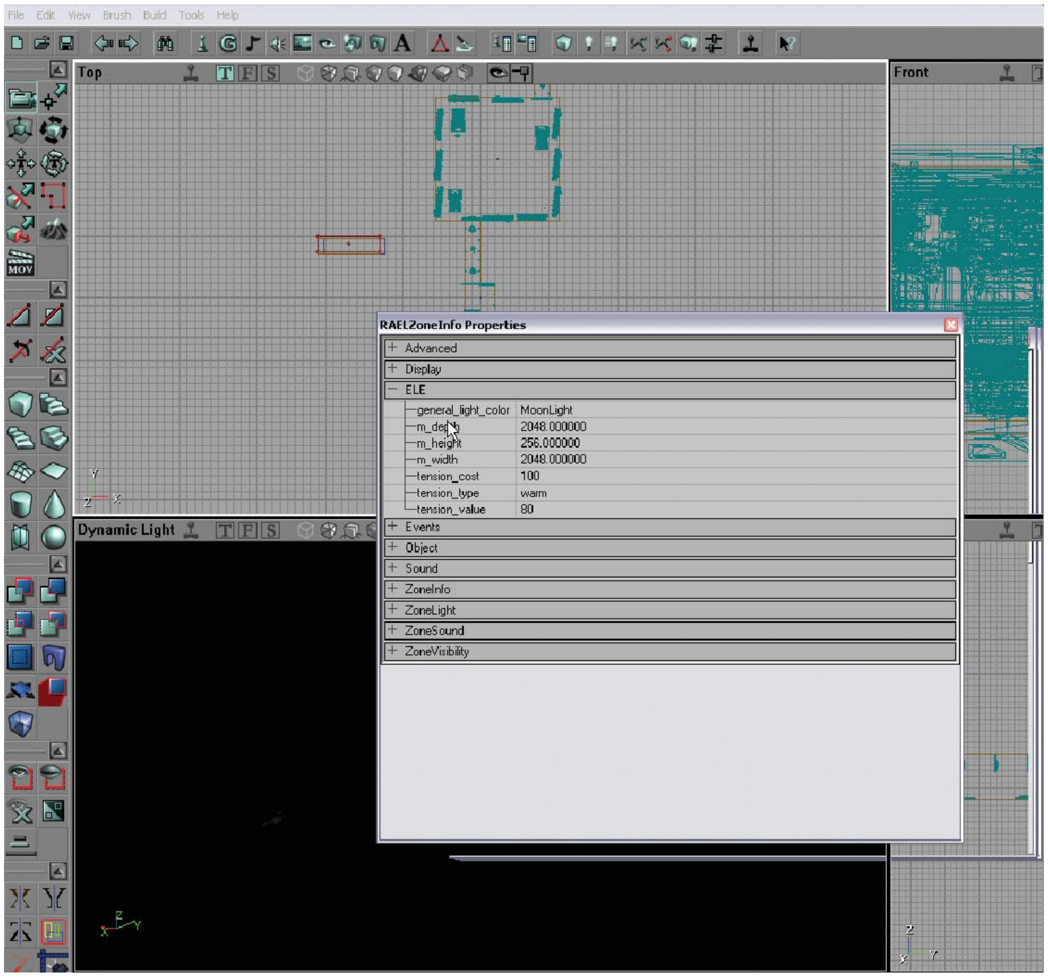
IMPLEMENTATION

ELE has been implemented in C# and Java. As mentioned earlier, it has been configured to use XML-based messages to communicate with rendering and game engines. This allows the architecture to be extensible and reusable. We have interfaced ELE with two rendering engines: *Unreal 2.0* (used by *Unreal Tournament 2003* (*UT2003*)) and *Wildtangent*.

We first integrated and used ELE in an interactive story called *Mirage* developed based on the Greek tragedy *Electra* [Seif El-Nasr 2004b]. A year later, we integrated ELE with a first-person shooter game created using the *Unreal 2.0* game engine. One concern was that the pace of first-person shooters was too fast for ELE, especially considering the optimization algorithms used. However, there were no performance problems with our current implementation of ELE within *UT2003*. ELE ran at a 30 frames/second rate, and thus was successfully ported to *Unreal* without any further optimization. In the remainder of this article, we will illustrate ELE using screenshots from *Mirage* and *Unreal*. Interested readers can download video demonstrations of ELE within *Mirage* and *Unreal* from [Seif El-Nasr 2004a].

Manipulating or Setting ELE's Parameters

As discussed above, ELE includes several weights that govern the lighting style and design goals' priorities. These weights are set for each zone in the game. Within *Unreal*, we integrated several variables to the *UnrealEd* interface to manipulate these parameters; Figure 2.3 shows some of them. Later versions of this interface contain all parameters and constraints, including color constraints. We have also designed a language that allows designers to set these weights and create rules to trigger weight changes depending on the story state. These changes can be absolute or relative to the current state held in ELE's memory. For instance, designers can create a rule that increases the visibility or mood goal priority when a character *x* draws a sword to attack the user.



general_light_color	MoonLight
m_depth	2048.000000
m_height	256.000000
m_width	2048.000000
tension_cost	100
tension_type	warm
tension_value	80

FIGURE 2.3 Specifying high-level parameters through UnrealED.

RESULTS

In this section, we will discuss results showing ELE in action. We will first show results illustrating the angle and color subsystems in isolation. In the following subsections, we will show some screenshots that illustrate the use of ELE in creating depth, influencing visual attention, and providing visual tension through manipulation of several lighting parameters, including colors, angles, and light layout, over time.

Angles and Colors Selection Subsystems in Isolation

As described in earlier sections, ELE uses several mathematical formulas to select the best angles and colors given the current situation (locations of characters, tension level of the game, etc.) and designers' set weights and constraints. In this section, we will show screenshots resulting from varying few weights and constraints and their effect on angle and color selections, in isolation. We have developed two standalone systems: ELE-angle and ELE-color, that allow designers to see, in real time, ELE's choices of angles and colors, respectively, as the effect of designers' manipulation of the lighting design parameters. These systems are also demonstrated through videos [Seif El-Nasr 2004a].To compare the resulting visual effects to the established cinematic techniques, we will show screenshots from movies showing similar effects.

Angle Subsystem

The angles of key, fill, and backlight as well as the contrast between fill and key light on a character's face affect visibility, emotional effect, modeling, and motivation. However, since these goals conflict with one another, designers often search for a compromise given design goals' priorities. ELE uses weights to establish these priorities. Figure 2.4 shows some screenshots where we varied the weights set to visibility and emotional effect (mood angle, which is represented in the equations shown using Character Intention function *CI*, and contrast). The figures show angles chosen for key light, fill light, and backlight. Since the emotional effect is a result of fill light intensity as well as light angles, the figure shows fill light intensity selection as a function of the set mood angle and contrast. It should be noted that the figure shows a small example of the possible set of variations produced by ELE using just a few lighting parameters and weights.

Figure 2.4(a) shows the angles and fill light intensity selected when the priority of visibility is set to high and all other goals are set to low. These angles are: key light = 45°, fill light = 315°, backlight = 150° (these are azimuth angles (horizontal) measured relative to the subject's nose line; vertical angles are set to 30° by default), and fill light lightness value = 100%. Figure 2.4(b) shows the angles and fill light intensity selected when the mood angle's priority is highest, and where the mood angle is set to a side light. Typically the mood angle can also be set by the designer using a rule-based system. ELE also uses default cinematic rules to set the mood angle depending on the story situation and the user model. The angles are: key light = 90°, fill light = 270°, backlight = 150°, and fill light lightness value = 100%. Figure 2.4(c) shows the angles and fill light intensity selected when mood angle's priority is highest and tension is high in the scene. Tension is a parameter that can be computed within the game depending on the interaction. For this particular demonstration, we set the brightness contrast to be high. The angles are: key light = 90°, fill light = 270°, backlight = 150°, and fill light lightness value = 0%. Figure 2.4(d) shows the angles and intensity of fill light selected when the priority of visibility is high and tension is also high. The angles are: key light = 45°, fill light = 315°, backlight = 150°, and fill light lightness value = 0%. The effects shown in Figure 2.4 can be seen in many movies and games, especially Film Noir movies.

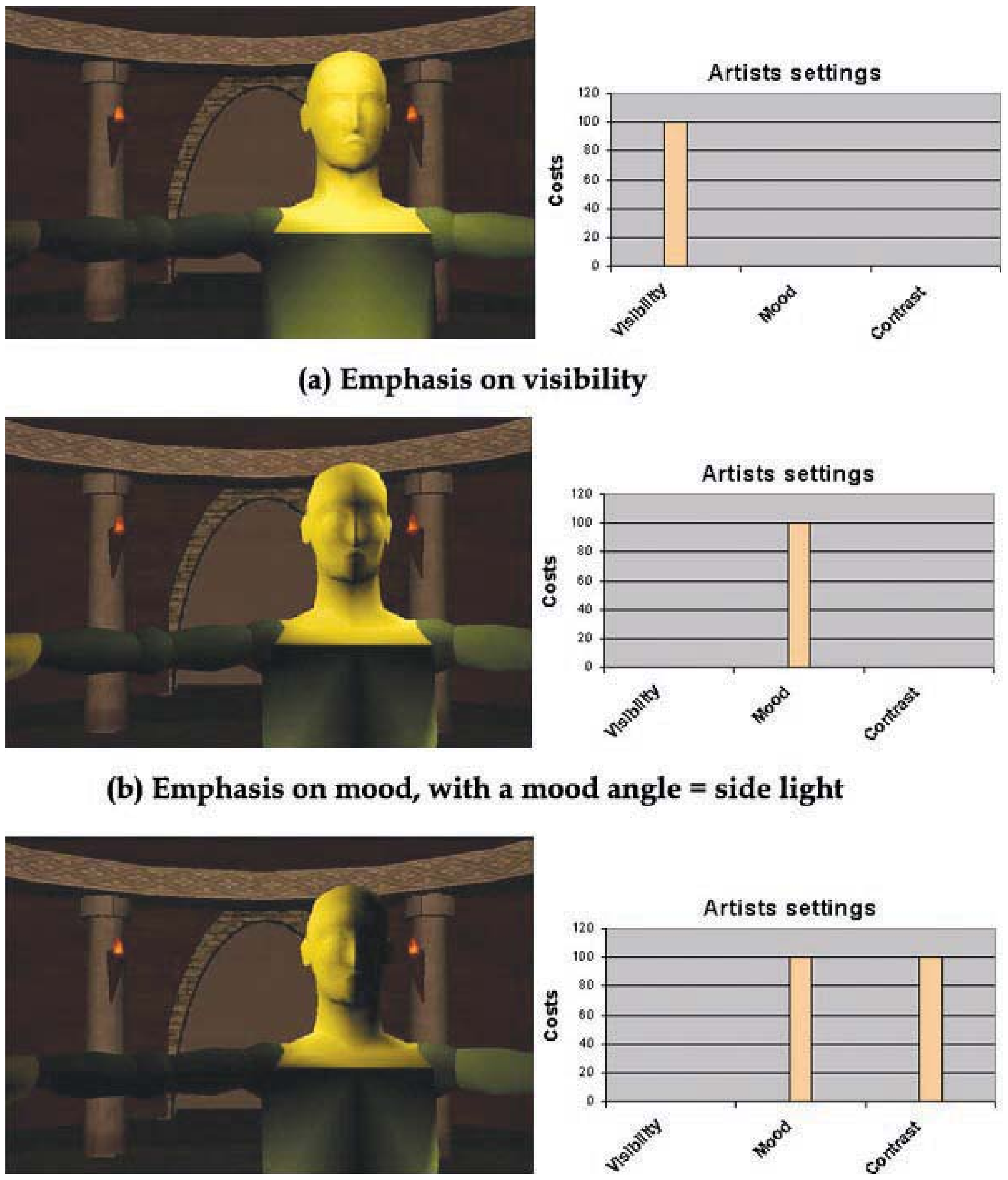

Artists settings
Costs
120
100
80
60
40
20
0
Visibility
Mood
Contrast
(a) Emphasis on visibility
(b) Emphasis on mood, with a mood angle = side light

FIGURE 2.4 Screenshots produced by the angle subsystem for different weights.

Color Subsystem

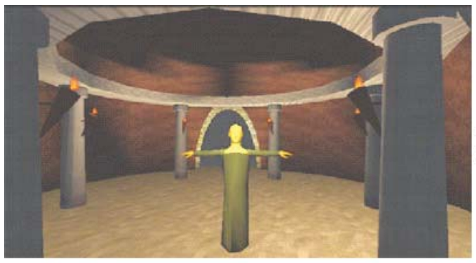
As described, designers use color and balance it to achieve several design goals, including increased emotional involvement, atmosphere, depth, visibility, and attention. To establish these functions, designers vary and balance color in several dimensions, including lightness of specific areas, contrast between focus and other areas in the scene, and warmth of specific areas in the scene. Figure 2.5 shows images produced by the color subsystem by varying the importance and value of lightness of non-focus areas, warmth of non-focus areas, and contrast between focus and non-focus areas, where the focus is the character in the scene. The images show the selection of RGB colors for the background area and the character's key light angle.

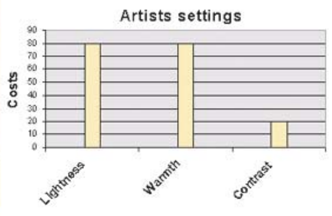
Figure 2.5(a) shows the selection of colors that reflect high lightness (80%) and warmth (80%) in the background area and low contrast between the focus (light on character) and non-focus areas. We validated the luminous value using Photoshop's histogram tool and an image analysis tool that we developed using formulas described in [Castleman 1996]. The warmth measure was validated by matching the colors produced to the graph of color warmth presented by [Katra and Wooten 1995]. Figure 2.5(b) shows the selection of colors that reflect high lightness (80%) and low warmth (5%) in the background area and high contrast between the focus (light on character) and non-focus areas, where contrast was set to be in terms of warmth and cool colors. Thus, as you can see, the color tone in the background area is blue and the color on the character's face is warm (red) because of the established high contrast. These values were also validated using the image analysis tool. Figure 2.5(c) shows the same setting as Figure 2.5(b), but the contrast was dropped leading to a cooler color on the subject's face. Figure 2.5(d) shows the selection of colors that reflect low lightness (20%) and high warmth (80%) in the background area and high contrast between the focus (light on character) and non-focus areas. In this particular image, contrast style was specified in terms of lightness. Thus, as you can see the lights on the character are much brighter than the ones used on the background.

To further validate the artistic and aesthetic effects demonstrated in Figures 2.4 and 2.5, we asked a director, a cinematographer, a lighting designer, and a gapher (a term used by film professionals to denote the person responsible for setting up the lights) to use the color and angle systems to vary the design parameters and critique the lighting effects produced. We asked them to focus their critiques on three different aspects:

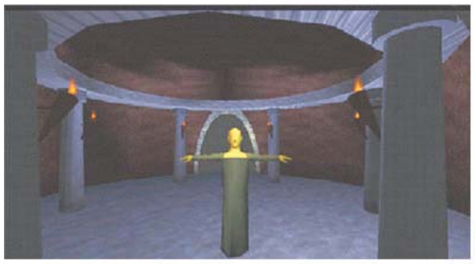
- The results of the lighting portrayed by adjusting the parameters
- The parameters themselves
- The similarity of the lighting techniques and images portrayed to ones produced by cinematic lighting

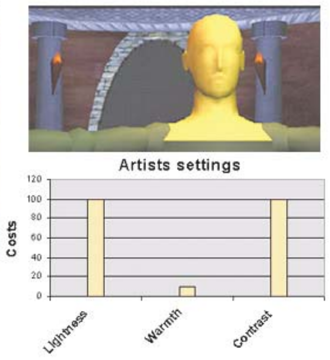
The critiques were very encouraging. Of course, film designers are accustomed to the use of cast shadows (e.g., shadow of the nose on the face) which are hard to reproduce using real-time rendering engines. Nevertheless, they all agreed that the lighting system produced images that were clearly similar to the ones produced in film; two of them specifically mentioned film noir movies and theatrical movies, e.g., *Moulin Rouge*. They all expressed interest in the parameters used, and thought they captured the lighting process very well. Two of them requested to use ELE to teach film lighting.



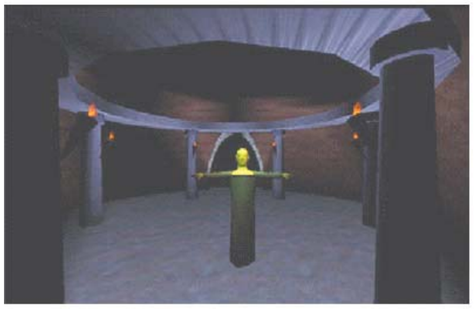


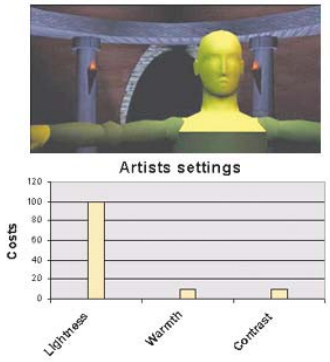
(a) Warm tones, low color contrast



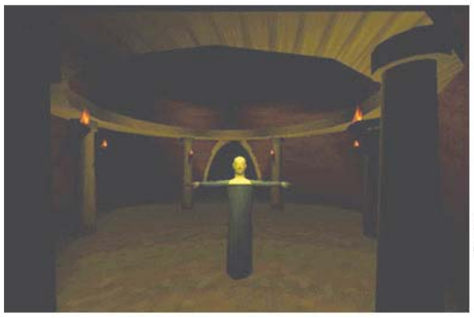


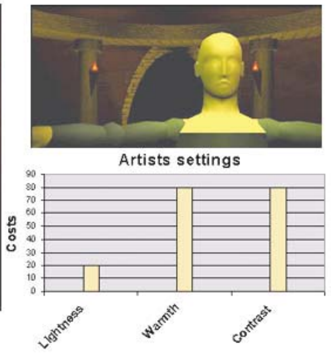
(b) Cool tones, high color contrast





(c) Cool tones, low color contrast





(d) Warm tones, high brightness contrast

FIGURE 2.5 Screenshots produced by the color subsystem under different weight configurations.

Portraying Visual Tension through Colors

Many game designers use sounds and visual patterns to emphasis dramatic moments in the game and escalate tension. For example, designers have used red flickering lights to signify danger (or health dropping beyond safe levels). In addition, some game designers increase color saturation each mission (*Devil May Cry*) to show an increase in tension. Since current techniques mostly use static methods, they are limited to predefined discrete changes in lighting, e.g., a change in saturation between missions where the tension value is predefined at the design phase, e.g., *Devil May Cry*. ELE provides a continuous technique for dynamically increasing visual tension while balancing other visual properties. To increase visual tension, ELE uses film convention discussed in [Block 2001]. Block describes techniques that use color contrast or affinity in terms of saturation, warmth, and brightness to represent visual tension.

As a testing case, we have configured ELE to increase tension every *x* milliseconds in a first person shooter game, producing results shown in Figure 2.6. We have configured ELE through the UnrealEdit interface (shown in Figure 2.3) to increase visual tension as a function of color saturation. Figure 2.6 shows screenshots of the *Unreal* game where ELE increases tension as a function of saturation. The images shown are at 2 seconds, 10 seconds, and 15 seconds into the game. Figure 2.7 shows the saturation values for each image. The figures show the pixel frequency (*y*-axis) for a particular saturation value (*x*-axis) showing the most frequent saturation values moving from ≈ 27 to ≈ 39 to ≈ 46. To produce these graphs, we passed the images through an image analysis tool (the same one used to verify and validate the resulting color, described above) that we developed based on color formulas described in [Castleman 1996] defining saturation and lightness. As shown, ELE varies the visual tension level portrayed while maintaining visual continuity and general color style of the scene, which is a torch-lit scene in this case. The technique is comparable to other techniques used by cinematographers and animators as discussed in [Alton 1995]; [Block 2001]; [Bucklan 1998].

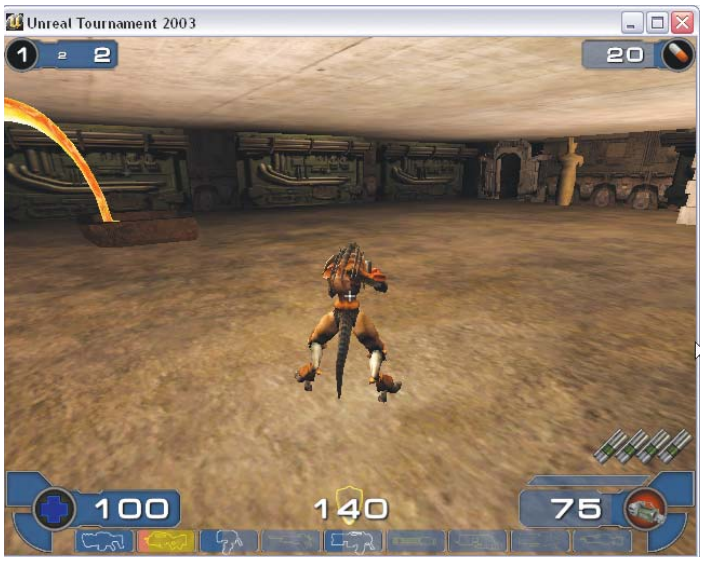


FIGURE 2.6A Visual tension in the scene.

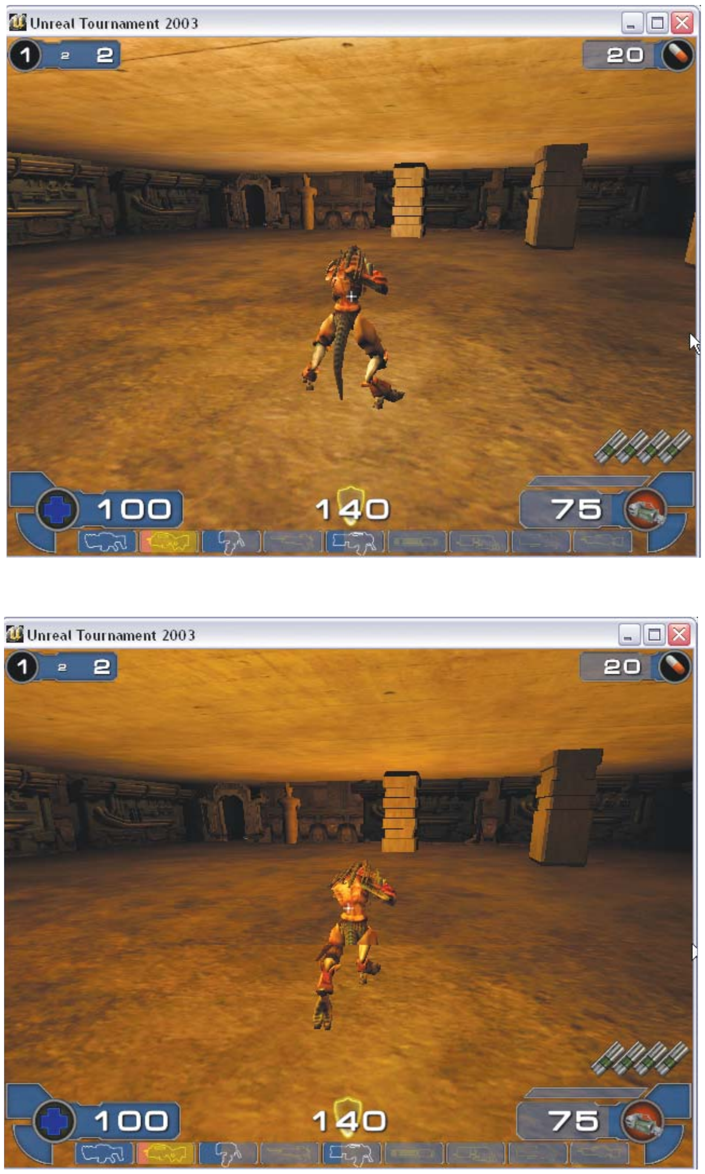


FIGURE 2.6B-C Visual tension in the scene.

While this technique may parallel the techniques used by cinematographers, it does not necessarily imply that it is an important technique for games. To validate this particular use and technique for games, we have compiled several screenshots from different games where it is evident that a similar technique was successfully used. For example, designers of *Devil May Cry 1* used color saturation to portray tension. Figure 2.8 shows saturation graphs for screenshots from

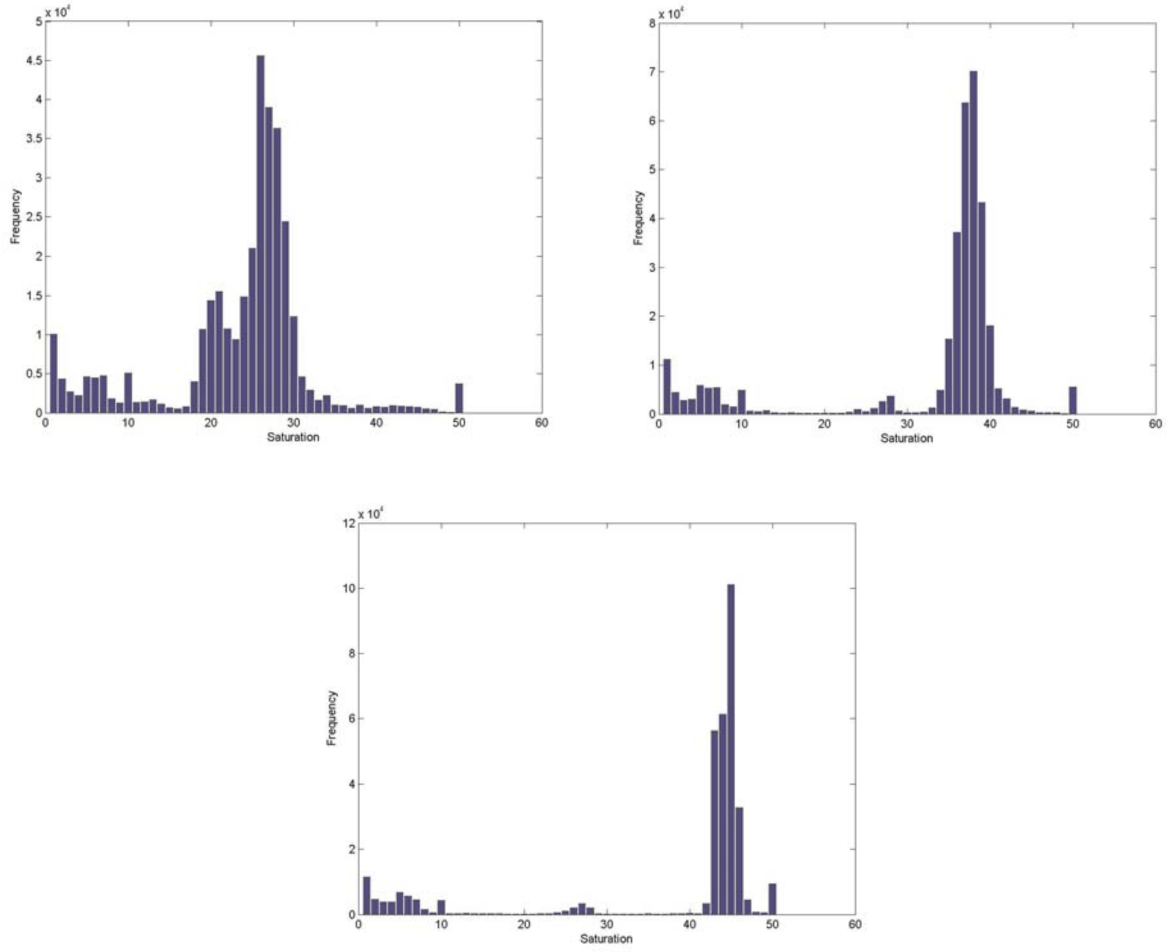


FIGURE 2.7A-C Saturation graphs for the images shown in Figure 2.6.

Mission 1 and *Mission 21*. *Mission 21* used very saturated warm colors to signify danger as shown from the graphs. *Doom 3* also uses many lighting techniques that interplay between high saturation, warmth, and low lightness at high tension moments in the game. We have also analyzed several screenshots of *Doom 3* in terms of saturation and lightness. Figures 2.9 and 2.10 show results of this analysis. The screenshots analyzed were taken at several high tension moments in the game, which use low lightness and high saturation as shown in the figures.

Currently, ELE supports an increase in tension through different methods, including increase in warmth, saturation (as shown in Figure 2.6), and decrease in lightness. Designers can dictate what method ELE uses to support tension, e.g., through saturation, warmth, or lightness. However, we hypothesize that there are more complex patterns for lighting that designers use to heighten

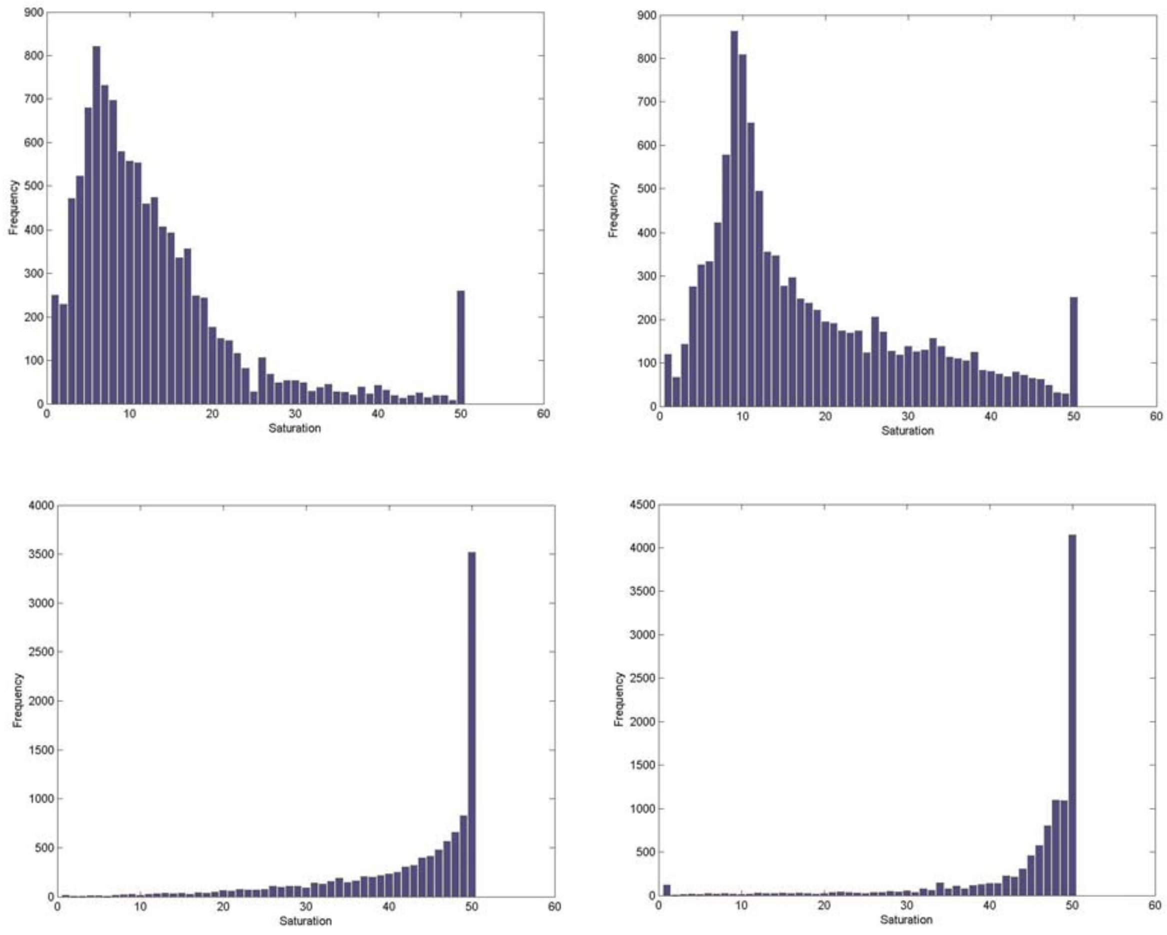


FIGURE 2.8 Saturation graphs for screenshots from *Devil May Cry Mission 1* (left 2 images) and *Mission 21* (right 2 images)

emotional involvement. *Doom 3*, for example, shows many lighting patterns that can be used to heighten immersion and emotional involvement. We are currently exploring a method for extracting cinematic and game lighting patterns and representing them mathematically within ELE. We are also in the process of creating an experimental design to measure the impact of color patterns on arousal using physiological measures, e.g., skin conductance, heart rate, and body heat. Many challenges still remain in the design of such an experiment, including isolating lighting effects and color from other visual content, normalizing physiological readings, and accounting for individual differences.

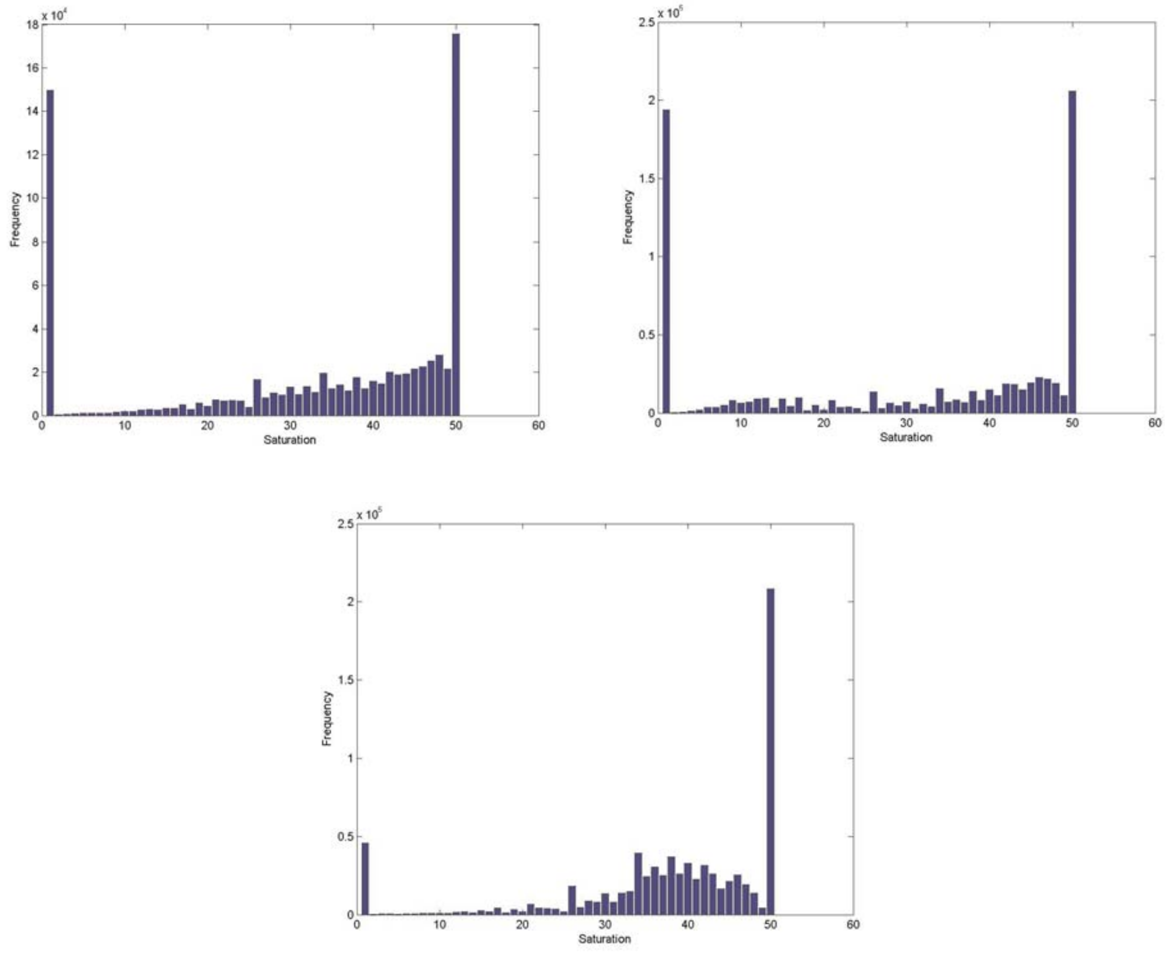


FIGURE 2.9 Saturation graphs for screenshots from *Doom 3*

Creating Depth

As described above, ELE can be adjusted to give importance to depth. In such a situation, ELE may sacrifice realistic, dramatic, and visibility goals for depth. Figure 2.11(b) shows a screenshot of a scene rendered using ELE, where the depth priority is emphasized. The figure also shows the effectiveness of ELE in portraying perceptual depth compared to other techniques such as camera fill. In the figure lit by ELE (shown on the right), depth is much more pronounced because ELE assigned different lights to the background and the foreground (allocation subsystem) and because the background lights were given lower intensity than those lighting the foreground (color subsystem). The camera fill approach renders a flatter image shown in Figure 2.11(a).

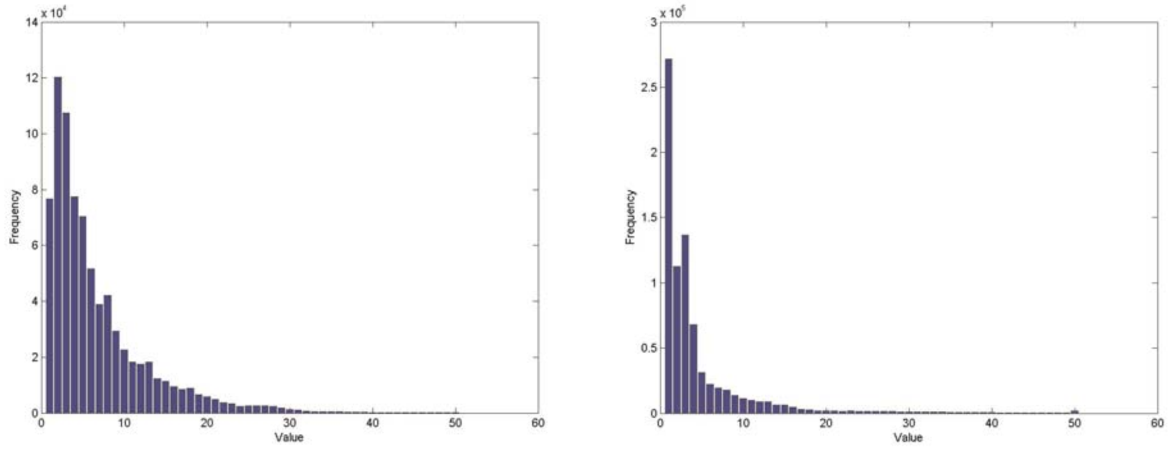


FIGURE 2.10 Lightness values for the screenshots from *Doom 3*

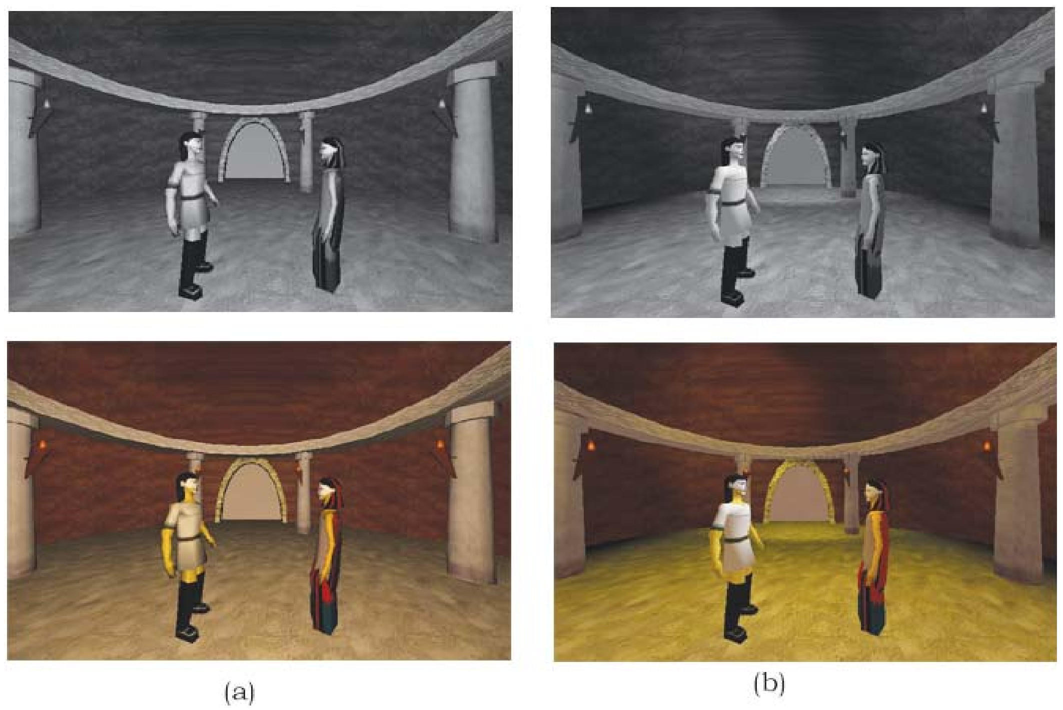


FIGURE 2.11 (a) Shows a screenshot from scene 8 from *Mirage* where camera fill is used. (b) Shows the same moment as (a) but with the lighting system where depth is given a high priority. Pictures on top portion of the image are black and white versions of the colored images shown.

Preliminary Results from Eye Tracking Experiments Showing Importance of Lighting for Establishing Visual Attention

We have set up a preliminary experiment to validate the use of lighting in directing attention. We invited 28 subjects to play *Unreal Tournament 2003*. We asked them to wear a head-mounted eye tracker and videotaped their interaction. Only seven of the subjects had played *Unreal Tournament* before. All subjects had played non first-person shooter video games before.

We asked them to play two games. Both games used the same game level and game type. The only difference between the two games was lighting, where one was manually designed by a graduate student in a game design course. This was done as part of an early assignment for the class, and was not originally part of this experiment. This is important to eliminate any bias that the student may have induced on the lighting of the level. The assignment was graded for best level and environment design. In the experiment we used this level lit by the student and the same level lit using ELE. ELE was configured with the following visual design goals priorities: visibility = high, visual focus = high, visual continuity = high, others were all low. Screenshots of some rooms of the level configured by ELE is shown in Figure 2.12.

The users were asked to interact with both games. They were not told about the difference in lighting between the levels in the beginning of the experience. However, they were debriefed about the use of lighting to grab attention at the end of the experience after they answered the survey. We interchanged the order of the levels between participants. Since in some cases, the participants quit early, we only analyzed the first two to three minutes of interaction.

Preliminary results of this experiment suggest that naïve users, using the manually configured level, were not able to decipher the environment quickly enough to respond to enemy attacks. They entered a room. Then, they started scanning the room for an enemy. By analyzing the videotaped interaction, we have identified that 100% of users missed the enemies the first three times they visually scanned the environment. Most users did not spot the enemy until the enemy started shooting. Naïve users were often shot and killed instantly. This caused frustration and early quitting. This problem was not observed when using ELE. On average, users using the level lit by ELE were able to spot the enemies within one scan through the environment.

These results were collected by:

- Manually analyzing the videotaped interaction where a white spot was superimposed on the game screen showing where the user is looking
- Analyzing user behavior while they play the game

Even though these methods are qualitative and imprecise, they still provide some good preliminary results that suggest the importance of lighting in directing attention.

Using this experiment, we can conclude that ELE was successful in directing attention. We can also conclude that depending on the experience of the lighting designer and the game design architecture, static methods can be difficult to use to dynamically direct attention. For example, a more experienced designer can use specific colored textures on characters to draw attention to them. A less experienced designer may have much trouble and may develop designs that suffer from several failure points, as discussed.

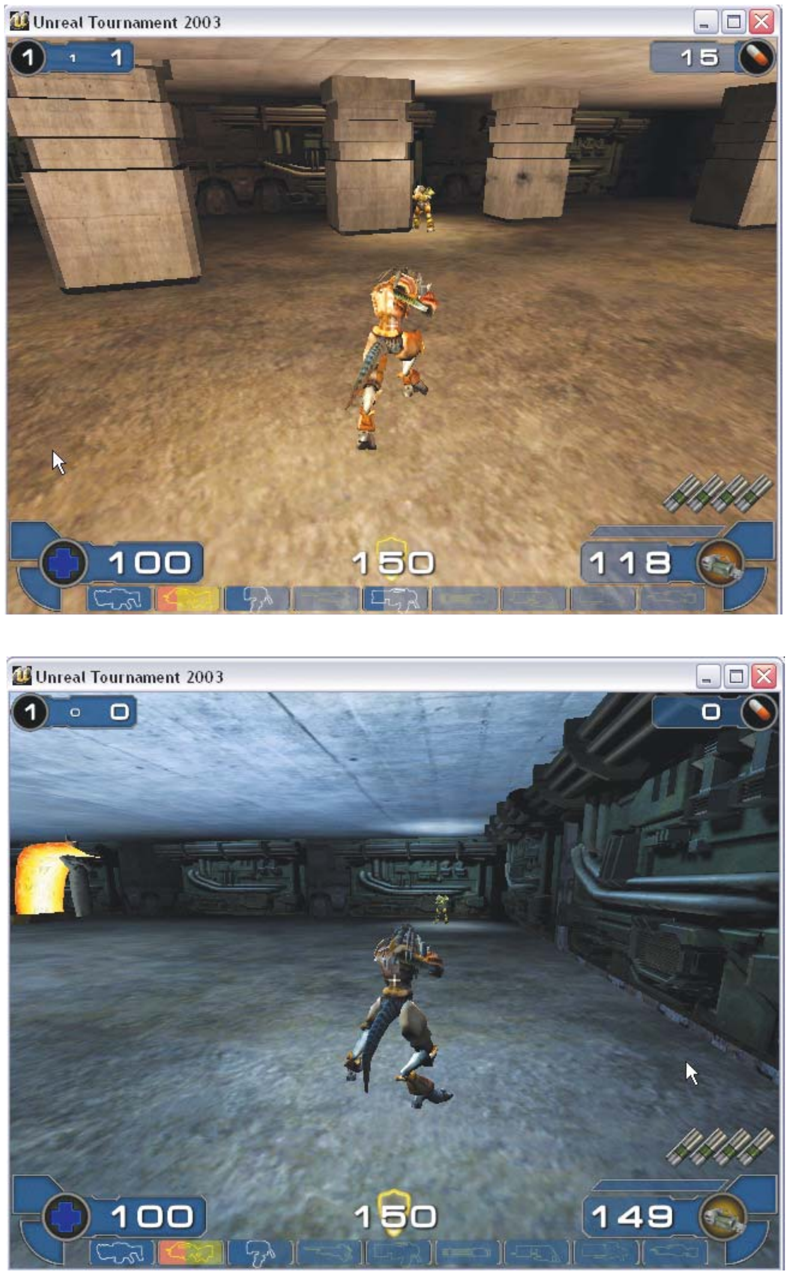


FIGURE 2.12A-B Visual focus.

Designers at Epic have acknowledged this problem. They reported at Unreal University that users of *Unreal Tournament 2003* weren't able to spot enemies easily. They corrected this problem in *Unreal Tournament 2004* by using halos around the characters. This fact further supports our hypothesis. This static solution solves the problem for games where objects and characters do not change importance or function. For example, a health pack is always considered important in a first-person shooter game, thus is always specifically colored to grab attention. These static techniques, however, fail in dynamic unpredictable environments where characters and objects change importance, goals, intensions, and sides.

Discussion

The previous section illustrated several screenshots showing ELE in action. In some cases, we validated ELE using screenshots from games, e.g., using saturation to portray visual tension. Although several advantages of using ELE over current techniques have already been discussed, in this section, we will identify and outline these benefits. It should be noted that depending on the game design architecture, some of these benefits may be more important than others.

Adaptability

Currently, lighting design techniques in games rely on a designer who statically and manually creates lighting designs for each level within a game. To create a robust design given the unpredictability of users' behaviors, designers will have to account for all possible contexts. The robustness of the design will then depend on (1) the designer and his experience, and (2) the game design architecture itself.

Since static techniques rely on a designer to account for all variations at design time, the result naturally depends on the level of experience of the designer. Figure 2.1(b) shows an example where the designer failed to anticipate a particular action or event, causing user aggravation because the enemy was not fully visible. Similar examples are evident in games such as *Prince of Persia*, *The Thing, Nocturne*, and *The Suffering*.

Currently games are either linear or non-linear with manageable predefined branching points (e.g., *Silent Hill* series). Thus, it is easy to define lights statically, because bad guys are always bad guys and important objects are always important. Dynamic lighting will become much more essential when we allow characters to change importance and intention relative to users' actions, and when objects are allowed to change importance in time. This leads to a more unpredictable environment, where static lighting techniques may be difficult to use. In addition, as games become more realistic, a more realistic simulation of time-of-day changes reflected in the lighting may be required. This will necessitate a dynamic simulation of lighting augmented with an intelligent system to balance the visual design goals for better game play.

In this article, we described an adaptive intelligent lighting system that at real time selects lighting layout, colors, and angles that satisfy the chosen style and given context, thus producing a context-sensitive lighting design. Therefore, one benefit of using ELE over current techniques is to create a more context-sensitive design that adapts to users' actions, thus alleviating the problems that surface in the games listed earlier. However, this is not the only benefit to adaptive real-time lighting. Adaptive lighting also plays an important role in enhancing the experience and heightening emotional engagement, as demonstrated in *Doom 3*. ELE extends this argument by providing an intelligent adaptive lighting system that is loosely coupled to game content and that can be used to produce similar effects as the ones used in *Doom 3*.

Abstraction of Lighting Design Patterns from the Actual Game

Many games use similar lighting patterns, as described above, e.g., an increase in color saturation and warmth to parallel game tension and heighten emotional involvement. These patterns are often re-done manually for each game and for each level within each game. This is a tedious and time consuming process considering the typical number of levels within a game. ELE provides a method of abstracting these patterns from actual game implementation, thus facilitating the design and promoting reuse.

Faster Development

Using ELE, designers can quickly set the lighting design in terms of style and aesthetic functions without worrying about low-level details of placements, angles, or color. This expedites the development process. Figure 2.7 shows few amendments that we added to the *UnrealEd* interface to manipulate general lighting parameters. Lighting the levels using this interface is as simple as setting these parameters. We have used ELE in two classes: an interactive narrative class and a game design class. Developing a game or an interactive narrative requires many days, weeks, months, or even years of content development, which is a process that includes story concept design, character design and modeling, character animations, creating and recording dialogue lines, and setting lights and camera angles. In both classes, students were asked to light the level first using current methods supplied by the engine used, which required them to set positions, layout, colors, and angles at design time. Students spent much time balancing the lighting parameters to achieve the desired effect. In subsequent assignments, they were asked to use ELE for lighting instead. This expedited their development process roughly by 5%. This is a rough estimate since the actual measure of time depends on the assignments and students' experience. While ELE had an impact on the development time, it is just one of many content development tasks that a developer will have to undertake; more tools are needed to accelerate the content development process.

Toward Modeling the Lighting Design Process

The lighting design process is a very complex and illusive process, which has yet to be fully modeled or documented. ELE provides a computational model of the lighting design process. The proposed model represents an important step toward modeling the lighting design process, which can be used by designers as a training tool or a guide for fully understanding the process. We are currently exploring other aspects of the lighting design process; ones that are not fully documented in cinematic or design theories.

Limitations

There are many limitations to the results presented. These limitations project themselves at different levels: limitations of the model, of the implementation, and of the engine used to show the results.

Limitation of the Model

ELE is the first attempt toward developing an automatic lighting design model. Lighting is a very complex design process that is most often undermined because it is very subtle and impacts viewers or participants mostly at a subconscious level. The model described in this article combines basic documented film conventions and represents them as mathematical formulas. There are many cinematic conventions that are not documented, e.g., complex lighting patterns. Also, games like *Doom 3* have introduced a different set of lighting patterns that have proved very effective in games. These patterns were not modeled in ELE. As mentioned earlier, ELE models tension using simple patterns of increasing contrast in terms of saturation, lightness, or warm and cool colors. But, there is no method for encoding complex lighting patterns as a function of time or narrative parameters.

In addition, ELE was developed and designed within a lab environment where we invited cinematographers, lighting designers, and directors to participate in the design process. We would like to integrate ELE within a game company where it can be used by game designers. This will provide us with several ways to improve the tool for use by game designers.

Lighting is one of the many visual parameters that affect perception. The visual design goals discussed here, such as depth, visibility, etc., also depend on camera, textures and character staging. These systems were not integrated within the current system, but still have an effect on the results presented. For example, the camera angle or field of view represented in the screenshots shown in the results section has a great impact on the images shown; we set these parameters to a default value in order to evaluate the lighting.

Limitation of the Implementation

The current implementation of ELE projected in the images shown does not account for shadows or inter-reflection of light. That is not a limitation of the model, but rather of the rendering algorithms used. The model itself does account for shadows and shadow placements by definition since it uses mathematical formulas based on cinematic theories. However, since the up-to-date rendering algorithms used did not account for cast shadows, inter-reflections, or refractions of light, the images evaluated and validated tend to be simplistic, as shown in Figures 2.8 and 2.10.

Future Directions

Many future research directions remain to be explored. As mentioned, ELE does not capture complex lighting patterns that heighten emotional involvement. We are currently working on extracting lighting color patterns used in movies and games and evaluating their effect on users' emotions using physiological measures. Depth perception is a very important element in 3D scene understanding. We hypothesize that it has a greater effect in games where users have to jump over ledges or spaced steps, e.g., *Legacy of Kain* and *Prince of Persia*. Since lighting plays an important role in depth perception, we aim to evaluate its impact within these tasks. The current rendering engines used to evaluate and validate ELE's performance do not incorporate cast shadows (self cast shadows) or inter-reflection of light. These are important properties to consider for lighting. For future research, we aim to interface ELE within an open source engine which we will extend to handle one- or two-level inter-reflections.

Conclusion

In this article, we have outlined a new intelligent lighting design system for game environments. The system, ELE (Expressive Lighting Engine), is grounded on many theories, including design theories from creative disciplines such as film and theater, and principles of vision from psychology, and perception. We presented some results that show the success of the proposed mathematical model in modeling and replicating some of the effects used by cinematographers and filmmakers within an interactive environment. We have also discussed the benefits of using ELE in games, particularly in increasing emotional engagement, enhancing depth perception and attention, and adapting the lighting to users' actions. *Doom 3* has already shown the success and utility of dynamic lighting in enhancing the interactive experience. With the exception of *Doom 3*, very little attention has been given to dynamic real-time lighting design. On the other hand, the game industry has devoted much attention to dynamic camera systems and dynamic audio. This article presents an important contribution toward recognizing the importance of dynamic real-time lighting design and provides an intelligent lighting system that adapts the lighting to the continually evolving context in interactive environments.

Acknowledgments

I would like to thank and acknowledge Dr. Frank Ritter and his students, especially Geoffery Morgan, at the Applied Cognitive Science Lab in the School for Information Sciences at Pennsylvania State University for setting up and helping with the eye tracking experiments. I would also like to thank my students, Joseph Zupko and Keith Miron, who helped integrate the lighting system with the *Unreal Tournament* engine.

References

[Alton, J. 1995]. *Painting with Light*. Berkeley: University of California Press.
[Arijon, D. 1976]. *Grammar of the Film Language*. New York: Focal Press.
[Birn, J. (Ed.). 2000]. *Digital Lighting & Rendering*. Indianapolis: New Riders.

[Bjorke, K. 2004]. “Cinematic Effects II: The Revenge.” Paper presented at the *Game Developers Conference*.

[Block, B. 2001]. *The Visual Story: Seeing the Structure of Film, TV, and New Media*. New York: Focal Press.

[Bordwell, D., and Thompson, K. 2001]. *Film Art: An Introduction* (6th ed.). New York: McGraw Hill.

[Brazel, R. 1997]. “Lighting Controls for Computer Cinematography.” *Journal of Graphics Tools,* 2(1), 1–20.

[Brown, B. 1996]. *Motion Picture and Video Lighting*. Boston: Focal Press.

[Bucklan, W. 1998]. *Film Studies*. Chicago: Hodder & Stoughton.

[Calahan, S. 1996]. “Storytelling through lighting: a computer graphics perspective.” Paper presented at the *Siggraph Course Notes*.

[Campbell, D. 1999]. *Technical Theatre for Non-technical People*: Allworth Press.

[Carson, D. 2000, March 01]. “Environmental Storytelling: Creating Immersive 3D Worlds Using Lessons Learned from the Theme Park Industry.” *Gamasutra*.

[Castleman, K. R. 1996]. *Digital Image Processing*. Englewood Cliffs: Prentice Hall.

[Fay, T., Grigg, C., Land, M., O’Donnell, M., Schmidt, B., and Whitmore, G. 2004]. “Panel: The State of Non-Linear Audio for Interactive Media.” Paper presented at the *Game Developers Conference*.

[Gillette, J. M. 1998]. *Designing with Light* (3rd. ed.). Mountain View, CA: Mayfield.

[Hill, B., Roger, T., and Vorhagen, F. W. 1997]. “Comparative Analysis of the Quantization of Color Spaces on the Basis of the CIELAB Color-Difference Formula.” *ACM Transactions on Graphics,* 16(2), 109–154.

[Katra, E., and Wooten, B. R. 1995]. *Perceived lightness/darkness and warmth/coolness in chromatic experience*. Unpublished manuscript.

[Kidd, W. (Writer), and J. J. Krakora (Director) 2001]. *Vermeer: Master of Light Video*. Washington: National Gallery of Art.

[Lowell, R. 1992]. *Matters of Light and Depth*. New York: Lowell-light Manufacturing, Inc.

[Luo, M. R., Cul, G., and Rigg, B. 2000]. *The Development of CIE 2000 Colour Difference Formula: CIEDE2000*, 2000, from *http://www.ifra.com/Website/ifra.nsf/html/ colorquality-club.html*.

[Maattaa, A. 2002]. GDC 2002: “Realistic Level Design for *Max Payne*.” *Gamasutra*.

[Millerson, G. 1991]. *The Technique of Lighting for Television and Film* (3rd ed.). Oxford: Focus Press.

[Moller, T., and Haines, E. 1999]. *Real-time Rendering*. MA: A K Peters, Ltd.

[Rangaswamy, Sudeep. 2000]. “Visual Storytelling through Lighting. ” *Game Developers Conference 2000*.

[Seif El-Nasr, M. 2004a]. *Demonstrations of ELE*, from *http://ist.psu.edu/SeifEl-Nasr/ele.html, http://ist.psu.edu/SeifEl-Nasr/ELEUnreal.html*.

[Seif El-Nasr, M. 2004b]. “A User-Centric Adaptive Story Architecture—Borrowing from Acting Theories.” Paper presented at the *International Conference on Advances in Computer Entertainment Technology* (ACE 2004), Singapore.

[Viera, D. 1993]. *Lighting for Film and Electronic Cinematography*. Belmont: Wadsworth Publishing Company.

[Wright, W. 2004]. "Triangulation: A Schizophrenic Approach to Game Design." *Game Developers Conference*. San Jose.

[Young, M., Riedl, M. O., Branly, M., Jhala, A., Martin, R. J., and Saretto, C. J. 2003]. "An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments." *Journal of Game Development,* 1(1).