ARCHANGEL STUDIO

## Light and colour in Rabbit

In this document I am going to explore the various ways we use light and colour as a design tool, and the reasons why these methods work. The observations and methods here are primarily aimed at the games engine we are using for Rabbit, called C4 engine. But the information could probably be found to be useful to people using any game engine. None of the concepts presented here should be considered definitive. They are merely methods developed through observation and practice

## Visual design philosophy

In the real world, light is an ever present and infinitely complicated ballet between the light photons and the material properties they interact with. And the sum of it that we see is so taken for granted that we never even think of it. As games worlds are nothing more than an interpretation of the worlds of reality and dream, the light itself takes on a more symbolic structure. And with all symbols, this is one of the reasons for their effectiveness.

Our over-all visual design philosophy can be summed up as this; *when designing a level, you do not attempt to replicate the real world. Rather you attempt to create what people will believe is a real world.*

## Darkness

Finding analogous comparisons between the real world and the game world in regards to light and colour, and their uses is not always practical when I design for games. Rather instead I look at another medium that is symbolic and interpretive in its use of light. Painting. In paintings true black in nearly never used. Dark is often used, but it is never quite the flat total black. And after all, what good is darkness, if you cannot see it? In games it is not good when you find an area that leaves your screen totally black. There are no visual clues for direction, so you are left to spin around wildly looking for something to navigate by.

This is sometimes used as a tool to disorientate the player, and can be seen used deliberately in games such as doom 3 and condemned 2. Usually this comes off as just annoying and has the effect of breaking the immersion with the world for the player.

So a lighting philosophy we will be using in our project development is;

*At no visual angle should there be a totally blank screen. There should always be a point or gradient of illumination perceptible.*

## Light

Lights can be used in a lot of interesting ways in games which can expand their effectiveness beyond just illuminating geometry. Lights placed correctly can provide visual cues for navigation. By carefully creating lit objects of note along the path the player can be herded along a linear path, or by creating a “light oasis” sufficiently different from the lighting on the linear path, say with a flickering or moving light source, you can lead the player to explore another area, while leaving it quite clear that this new area is a branch from the main path.

The practicalities of using light in games depend on the understanding of the different emitter types most commonly used in games, and how to use them in conjunction with each other to achieve the lighting you need in a scene.

## Light Source

All too often the actual source of the light is not given enough attention. The light source is the visual representation of where the light is coming from. A light bulb, a florescent sign, or even the sun.  Often a novice will light a level to reveal the geometry, and then light fixtures placed with little reference to the actual light being used. But on the other hand, light in games behaves quite differently to light in the real world, so positioning a light emitter in the same position as the model that is representing the light source will not create the illusion that light is being emitted from that model. This method fails for two reasons. If the light emitter is too near the light source model, the model will look over-bright, if it is an Omni directional light, that is. If it is a spotlight, the light source model will not be illuminated at all.

## Colour

If you look at the actual colour things are in the world around you, you will notice that there is very little to recommend it, at least the parts of it that humans colour. The colour from the sky, streetlights, and reflected colour from everything else tends to smooth out the jarring discontinuity of colour in the real world.

The use of colour is probably the most important element of our light solution. Firstly we dictate what colour our "darkness" is by creating an ambient light level with subtle but distinctive colour properties. We chose a dark green for our night time street, as it matches the skybox colours, compliments the orange colour of the streetlights, and makes the world appear dirtier than it actually is. Then you tint the lights to match the lighting colour scheme. We tint our main spotlights a light orange, our Omni lights a darker orange.  Then we place some small point lights up high on buildings with a blue-white tint. This emphasises the height of the buildings, but must be used sparingly. This creates a nice uniform colour palette where previously there was a jarring jumble of colour, and also goes a long way to suggest a certain atmosphere.

We have several colour palettes that will be uses in different places. For example the sewers will have a yellow tint to the ambient light level, with blue light from the bulbs. This will make the sewers feel cold, the yellow suggesting a particularly sewer-relevant type of dirt.

In essence, our colour philosophy can be described as this; *Start with a colour for darkness. Fill in with a complimenting colour for the main lighting scheme. Then place points of contrasting colour to accentuate certain properties of the level.*

## Motion

If nothing moved in a digital game, it would be a picture. Nearly all interactions in a games world require movement of some kind to happen. With today's technology there is no reason what light itself cannot play a larger part in this.

We can divide up movement in games as player movement, object and NPC movement, and light movement. Light movement is the only factor that will always be present in all three movement types.

- When the player moves, the field of view changes to show a new lighting vista.
- When an object or an NPC moves, the light on the object will change, casting new shadows. Occluding other lit objects.
- And of course when the light itself moves, it changes what the player sees on screen.

For light motion we can use two methods. One is to alter the brightness of the light, such as creating different rates of light flicker, and the other is to actually create a method of moving both the light source model and the light emitter itself. When moving a light you have the option of having the movement pre-scripted such as rotating emergency lights, lights attached to fans, or vehicles. You also have the option of creating dynamic movable lights, by making them swing when an object pushes them or impacts them, such as a light on a wire would. It is possible to attach lights to other objects that can move, such as doors.

Using these techniques correctly is all about balance. If you have too drastic an amount of light movement present it will be disorientating. Mild movement is enough for the larger more dominant lights. You can use more intense light movement to

highlight something important to game-play. It is best to make either the area that is affected by the intense light movement limited in size, or the light movement limited in duration, so it doesn't overly fatigue or irritate.

## Shadows

The shadows you create in a game level are every bit as important for design consideration as any other design factor. Shadow manipulation is really where the light/ geometry balance is felt most. Light plus geometry equals shadows. Games engine depending, there are various ways you can control shadows. One common function is being able to set an object to either cast shadows, or not. An advancement of this is to be able to dictate which lights the object will or will not use to calculate where a shadow will be placed.

## Light creating form

Using lights and shadows for structural and textural design in a level is an important function to keep in mind, and there are various ways to achieve this. The classic ones would be; placing a light under a staircase. This will projects shadows in a fan-like pattern up the wall, giving diagonal shapes to what could previously be a piece of level design overly dominated by ninety degree angles. Other implementations would be light shining through gratings and grids, or placing a cage over the light source model so it projects that cage pattern onto the surrounding geometry. Another method of using lights in a textural manner is to actually apply textures to lights, or use projectors.

Most engines these days have some form of texture loading function on at least some of their light types. The best functionality for this I have seen was in the Doom 3 engine. In there you could apply any type of texture to any light, even coloured and animated textures. Unreal engine 2 and other engines allow the use of projectors. These project a texture onto the level geometry, just as a real world projector can be used to project an image onto a surface. The image you project onto the game world's geometry is a grey scale image, where everything over 128 gets brightened, and everything below 128 gets darkened. All of these techniques allow you to further enhance your level by introducing low cost visual complexity, but care must be taken to ensure the result does not look overly contrived. Make sure there is a good excuse for the lighting.

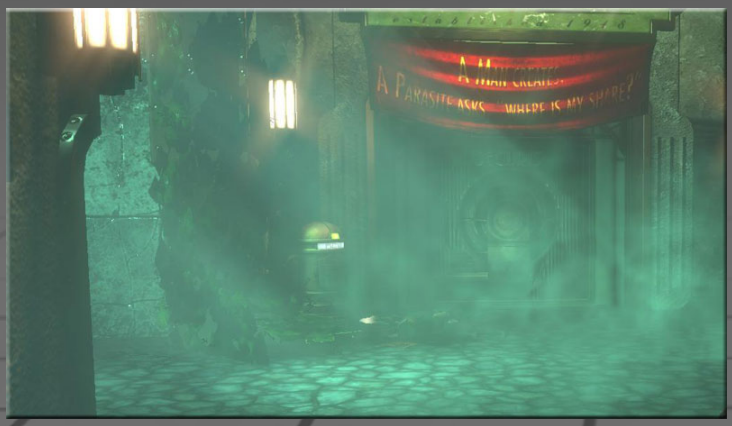
## Materials

The approximation material composition of the geometry is an important factor relating to lighting. Whether the geometry is made of smooth polished metal, rough stone, skin or ice is expressed by how it lights. With totally flat ambient lighting and no specular reflection every material would look more or less the same. With the advent of multi-texture material shaders, the ability to dictate the bumpiness and shininess of a material has made a massive difference to the level designer's ability to describe what things are made of. How you light these things will help emphasise their material properties. Close point lighting of shiny materials will give a greater sense of shiny reflection. Lighting from two directions with different lights will soften smaller indentations and smooth out the larger bumps and cracks in a rock type material.

## Light, geometry and negative space

This is more about a way of thinking than a definable practice. The actual game does not take place in the level geometry. It takes place in the spaces where the level geometry is not. The level geometry merely provides a visual backdrop and functional boundaries to the game environment. Remember when lighting a level that not only must the level geometry be lit and visible in the right places, but the spaces between the level geometry must also have the correct balance of light in it to affect on any dynamic objects which will pass through that space. Physics objects, NPC characters, and the player model itself.

## Volumetric lighting



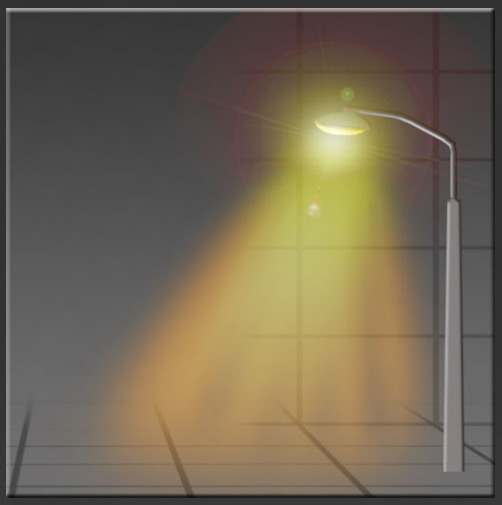
Following on from the negative space idea, you can start to consider ways in which light occupies space, and find ways in which to simulate it. As mentioned, light in the real world bounces off particles in the air. That is why you can see light beams.

Light beams can be simulated with either particle effects or billboard sprite textures.

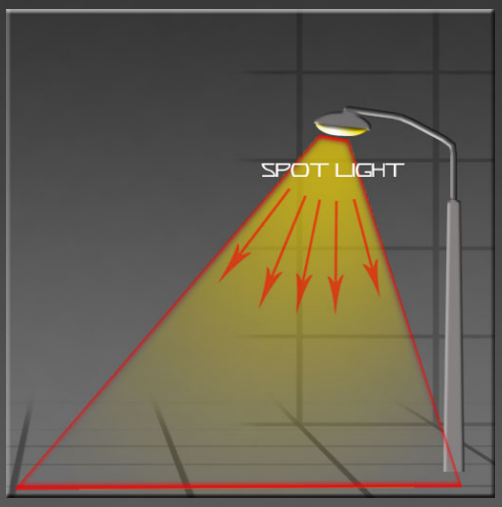
Light also illuminates dust, smoke, steam and fog. When implementing these atmospheric particle effects in a game level, it is best to treat them as part of the lighting solution, as they can be choreographed to serve or enhance similar functions.

## Practical lighting breakdown

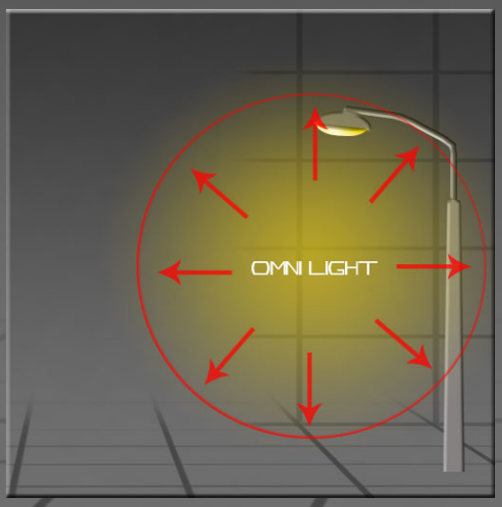
Below is a step by step breakdown of how we will set up a simple streetlight system. Depending on context, you approach lighting an area differently. So long as you are aware of the design elements you have control over, and a knowledge of the tools you have access to in your games engine, you will find a suitable solution.



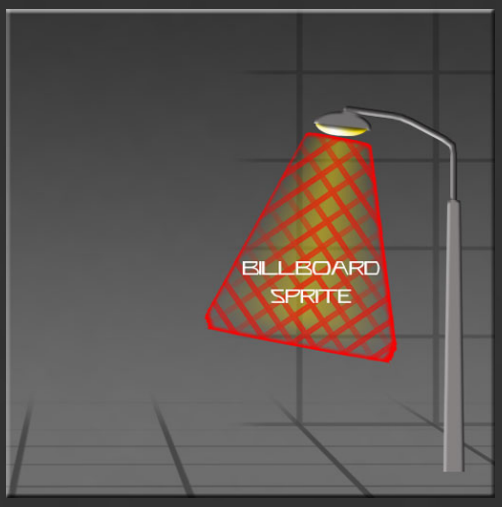
Here is a graphic demonstrating light in the real world. The light is emitted from the bulb, is being bounced around by the air particles thus illuminating a far more vast area than the direct beam. The light will then bounce off the ground, the lamp post, and everything else. The particles in the direct blast of the light form a visible beam of light, and there is also a flare.



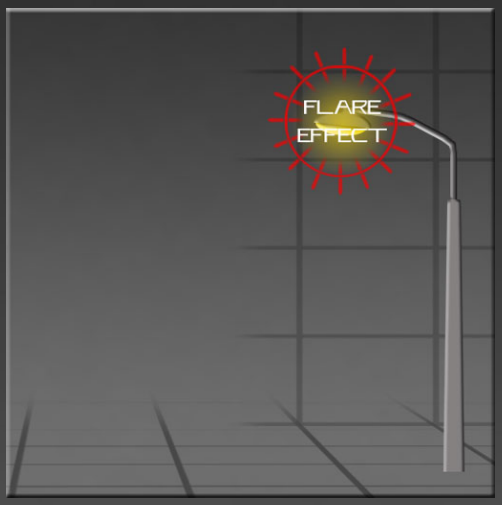

To approximate the effects you see of real light we can start with a spotlight. This generates illumination in a cone. We can then use a black and white texture as a projection that can create a fall-off pattern.
Still, this light only illuminates things in its cone. There is no light bouncing around providing illumination to either the lamp post of the surrounding environment, nor are there any visual light beams.
All you will see is the places geometry intersects the cone of light.




Here we are placing an Omni light beneath the lamp post. We put this in as a second light, with a large radius and a darker tint. This provides illumination to the lamp post and the surrounding geometry. Note the Omni light is about half way between the “source” of the light and the geometry it is illuminating. If it were placed too near the light it would overly illuminate the lamp post and the geometry directly above the lamp post.
Also, this light will try and splash the shadow of the lamp post over the building behind it, which would look all wrong, so you have to be sure that this light does not cast the lamp post’s shadow. Most games engines have the facility to set which lights generate shadows from which pieces of geometry.




In the real world, light hits all the little particles in the air. This gives the illusion of light fall-off, Rayleigh scattering and also the Tyndall effect. In digital 3d this is termed “volumetric lighting” there are two methods of faking volumetric lighting in games engines. One is with a particle effect. This can be computationally expensive. The other is with a billboard sprite effect. Essentially a 2d plane with a translucent Texture on it that is always rotated to face the player. We use a slight variation of this technique, called a quad effect.




Flares are a visual effect of light scattering within the mechanism of the eye. Not an actual property of light. some engines have various levels of sophistication in how they implement flares, but they usually function on a similar premise to how we fake volumetric lighting. With a billboard sprite and a translucent texture. This is placed in front of the light source model.

## Conclusion

Light and colour are important elements to game design. But games are vast areas with a lot of different specialist areas to take into account. The level of detail you put into one area of design must be backed up with equally thorough design in all other areas to maintain cohesion. During the design and prototyping faze most developers would work out some guidelines helping to define a production pipeline. How structured this would be would depend on the size of your team and the scale and complexity of the project. The larger the overall scale, the more a developer will need to get clear guidelines written down so the entire development team understand their role within the project. How you choose to communicate the guidelines to your team is up to you. As long as the rest of the team understand.

# Bibliography

Seif El-Nasr, M.  Niedenthal ,S. Knez, I. Almeida, P. Zupko,J. (2006) Dynamic Lighting for Tension in Games [Online] Available: **http://gamestudies.org/0701/articles/elnasr_niedenthal_knez_almeida_zupko**

[Accessed April. 24, 2008].

Freudenrich, C. (n.d.) How Light Works. [Online] Available: **http://science.howstuffworks.com/light.htm**

[Accessed April. 24, 2008].