SIMS

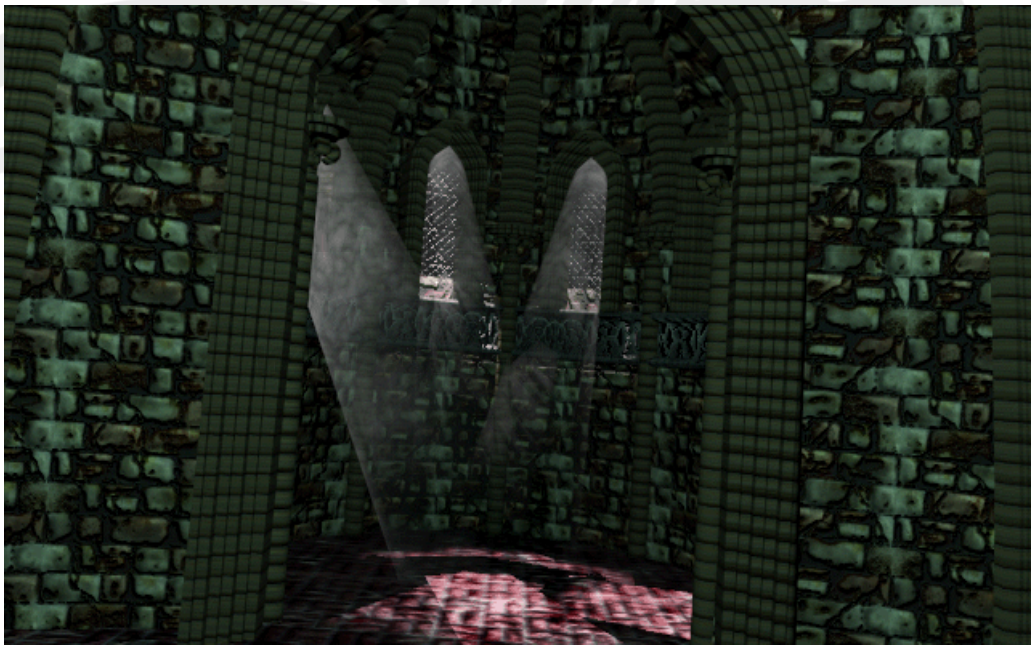
Sprite **I**lluminating **M**ask**s**

Written By Eric M. Scharf
Engineering Contributions by Floria Ross

The Vampire the Masquerade game requires 3D-rendered, 2D-sprite-driven characters to animate through 2D light sources within static 3D-rendered 2D shots. SIMS (Sprite Illuminating Masks) offer one such solution, the conceptual process for which is detailed below.

Set (An Environment Containing Multiple 3D-Rendered 2D-Displayed Shots):

Example – The **Annex Shot** from within the Cathedral Set.



Lighting Setup:

The example shot features light being cast through the portals in the front, to the left, and to the right of an annex. For each light source in a shot, there must be a corresponding shape or "sprite illuminating mask" that designates from where each light source is originating and to where each light source is projecting. These masks are created by rendering 3D data of each light source as physical volumes.

Each light volume will have a gray scale range. This range corresponds with a generic, reusable texture-map that is applied to each piece of 3D light geometry within Autodesk 3D Studio. Each gray scale will be initially limited to 5 grays (from white to black, brightest to darkest). This limitation may be overcome depending on how the available shot palette is arranged (with a greater number of muted or neutral colors potentially allowing for a wider gray scale range).

PC Sprites:

The PC (Player Character) sprites will command a static set of 32 colors within each 256-color, set-specific palette. Those 32 colors are dedicated to hair, skin tone, clothing, and how each of those character-specific details interact with each light source within a given shot.

NPC Sprites / Objects:

NPCs and props (interactive environment objects) will maintain 64 unique colors per set. NPCs and props may also borrow from the 32 colors of the PC. These 64 colors are also used for interpolation of sprites moving in and out of areas that differ in light exposure (e.g., from shadows to a harsh spot light).

Palette Breakdown:

The entire 256-color per-set palette is divided into two 128-color palettes (the "common" or shared palette and the background palette).

Common Palette Composition:

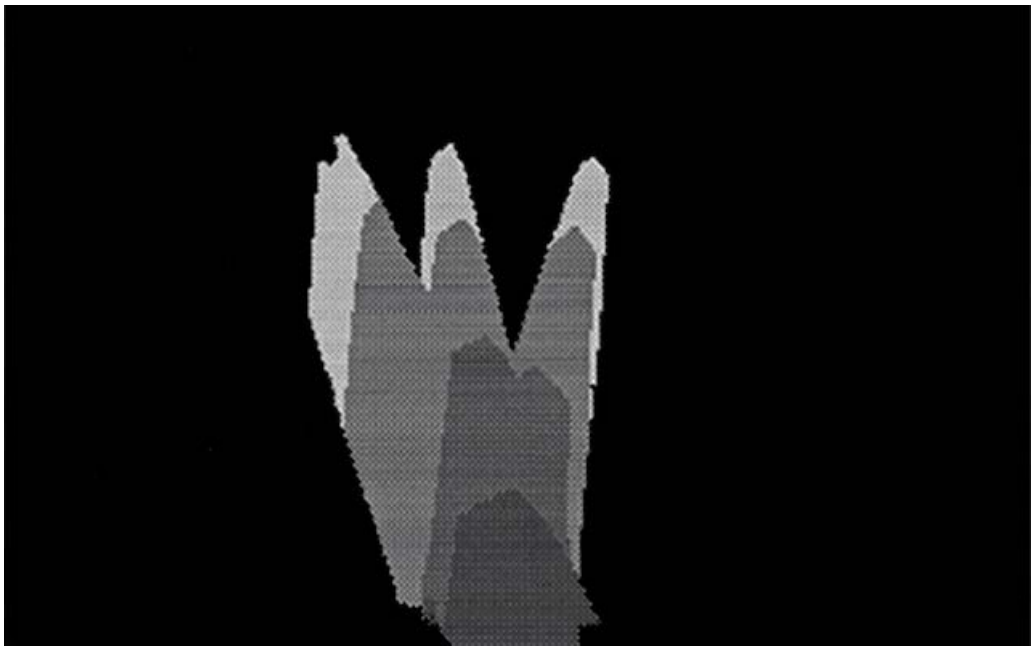
32 colors for the PC (which are shared with the NPCs)
32 colors for the GUI (Graphic User Interface)
64 colors for the maximum number of (3) NPCs per set

Background Palette Composition:

128 colors specific to a particular set

Production Pipeline:

A rendered shot (640x368) in PCX format will be delivered by the art team to engineering. This rendered image will be accompanied by a secondary rendered PCX (640x368) which displays the origin and projected direction of each light source within the given shot. The light sources are shown as a five level "flux" gradient.



Engineering Design of Varying Light in Sets and Shots:

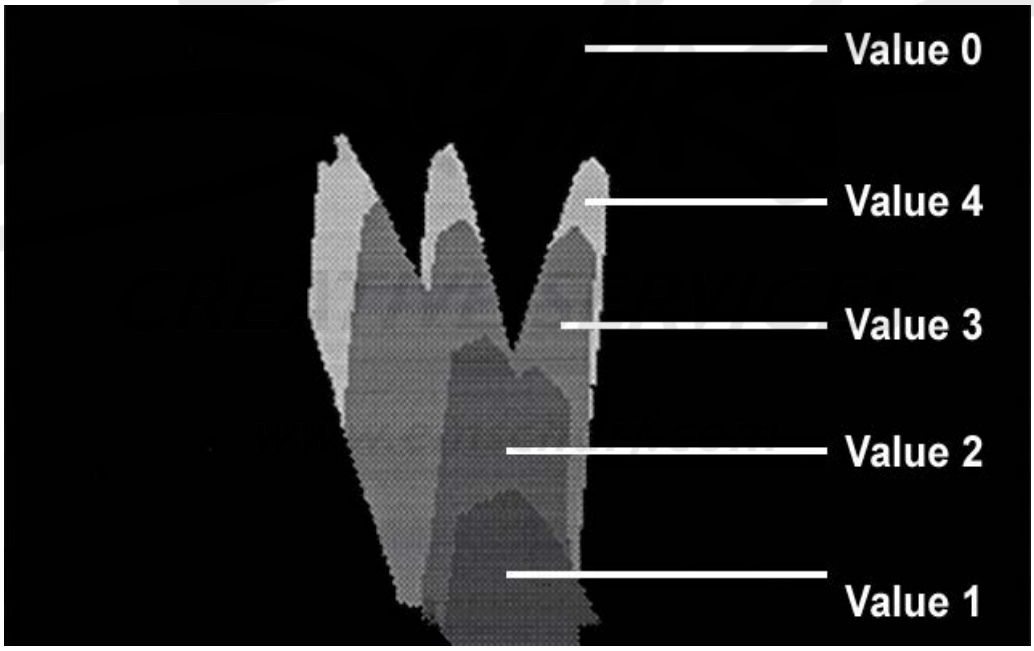
The secondary PCX generated by the art team will contain a series of 0-4 values which correspond to shades of greys.

Black = 0, Dark Grey = 1, Medium Grey = 2, Lite Grey = 3, and White = 4.

The palette consists of a series of ramps with each ramp containing 5 colors. Each sprite and prop are assumed to be rendered in the first color of a color ramp. The correct palette index is calculated by taking the shot's light source bitmap value at a given x, y coordinate position and using that position as an offset with the sprite / prop pixel value. The process will compare each and every pixel, because of an understandable focus on speed, efficient computation, and the reasonable minimum detail required to make the sprite / prop look good.

Example Light Source Bitmap Values:

Value 0 implies that no palette adjustment is necessary
Value 1 implies that the palette index for this pixel is increased by 1
Value 2 implies that the palette index for this pixel is increased by 2
Value 3 implies that the palette index for this pixel is increased by 3
Value 4 implies that the palette index for this pixel is increased by 4



Sample Section of Palette of Background:

Index: 100 101 102 103 104 | 105 106 107 108 109 | 110 111 112 113 114

Please NOTE: 100, 104, 110 are base colors.

If the light source bitmap pixel is 2 and the base color is 105, the actual index for the pixel will be 107.

As the data in the light source bitmaps is based on offsets from the rendered sprite / prop, the original bitmap of the sprite / prop has to be preserved at all times, and the actual bitmap that is rendered is generated in real-time. This is the same procedure the "z-depth object detection" operates, thus, combining the z-depth and light detection will be advantageous.

Operational Flow:

"n" is defined as the number of pixels that can be grouped together to achieve maximum effect with least effort ratio.

For each nth pixel within the sprite / prop:

Offset = value of the nth pixel of the light source bitmap for each following n pixels expressed.

New sprite = original pixel value + offset.

Increment sprite count by n.

Sample Test Data and Output

Sprite	Light Source	n	Result
80, 123, 128, 200, 207, 209, 100, 110, 54, 50	0, 0, 2, 3, 3, 4, 1, 2, 3, 2	2	80, 123, 130, 202, 210, 212, 101, 111, 57, 53

Enhancements to Basic Method:

The light source file can be converted into a data file that is exclusive to actual data used and none of the data that is skipped.

The light source, from the above example, would look like: 0, 2, 3, 1, 3

The above example will require additional calculations towards locating the x, y position in the light source data, which aligns with desired efficiency and a reduced memory footprint.