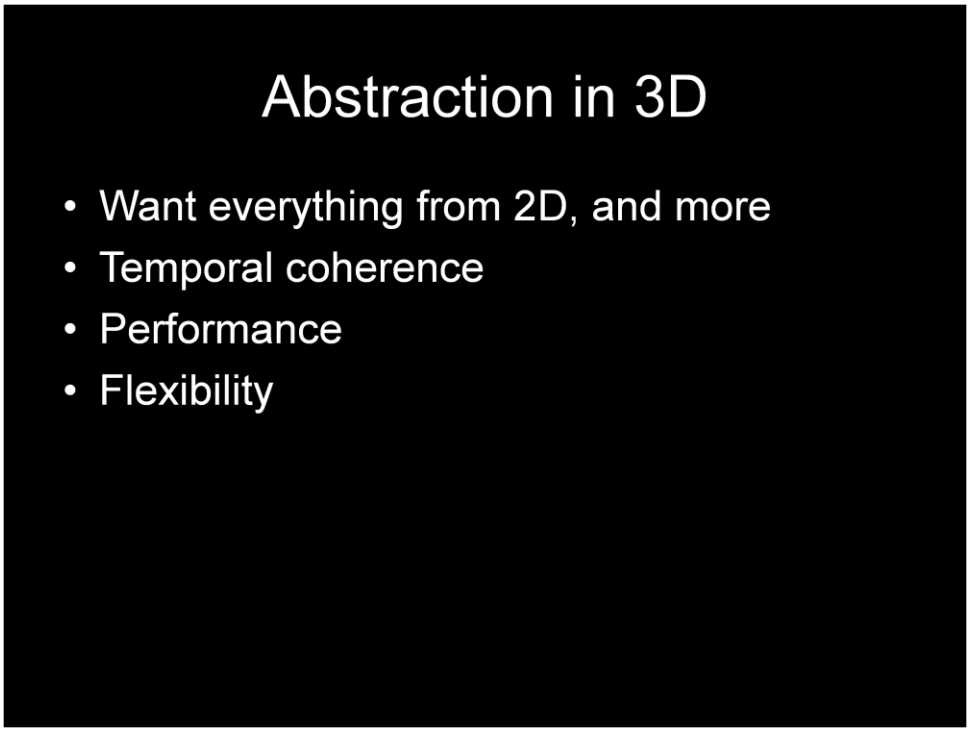
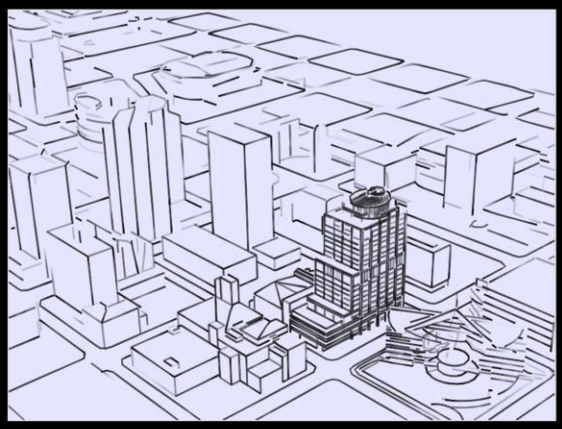

Part VIII:
Controlling Detail and Attention
Forrester Cole
Line Drawings from 3D Models
SIGGRAPH 2008



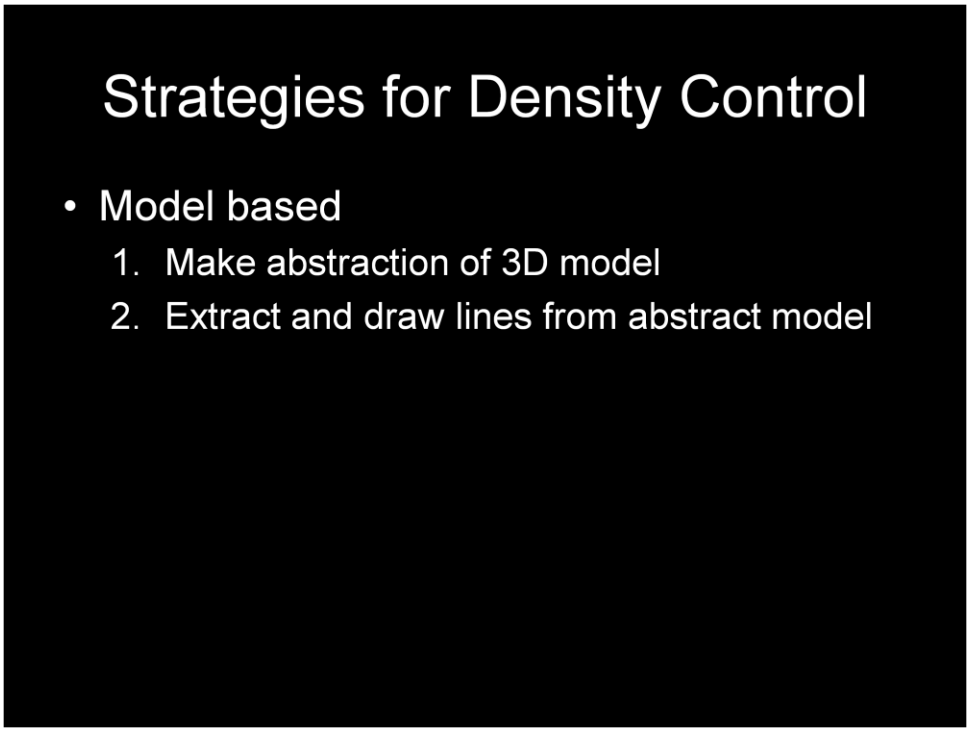
In three dimensions, we would like the same control over abstraction as in two dimensions, but there are several additional requirements. We'll focus on animated and interactive applications, since if we only care about still images, we can just use the techniques Doug described. For animation, we need temporal, or frame-to-frame coherence between consecutive frames. For interactive applications, we need both temporal coherence and good performance. A desirable, though not strictly necessary, quality is flexibility in the types of lines that the method can handle.

Line Density

- Line density important cue for abstraction
- Basic pipeline lacks control of line density



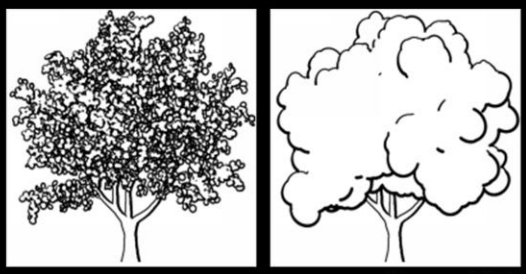
Line density is one of the most important cues for abstraction in line drawings. In the image shown, the central building is obviously emphasized while the surrounding buildings are not, and the only variation between the two is the line density. However, the basic pipeline described previously hasno natural control over screen space line density - lines are drawn wherever the extraction and visibility algorithm determine they should be.



The simplest way to deal with inordinate line density is to nip it in the bud by creating an abstraction of the model itself, such as by smoothing. The lines are then extracted from this smooth model, and rendered as normal.

Abstraction of Trees

- Represent leaves with disks
- Vary disk radius to control abstraction
- Good temporal coherence

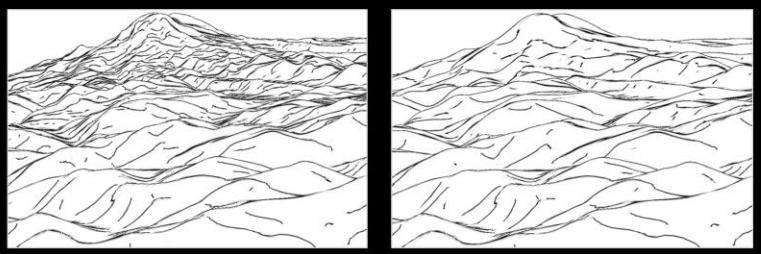


[Deussen 2000]

So, how do you make an abstraction of a model? One way is to create a specialized method tailored to a specific type of model, such as bushy trees. In this example, we represent each leaf with a disk, and vary the radius of these disks to control the level of abstraction. This method has the nice effect of not only controlling density, but also varying the shape of the lines themselves.

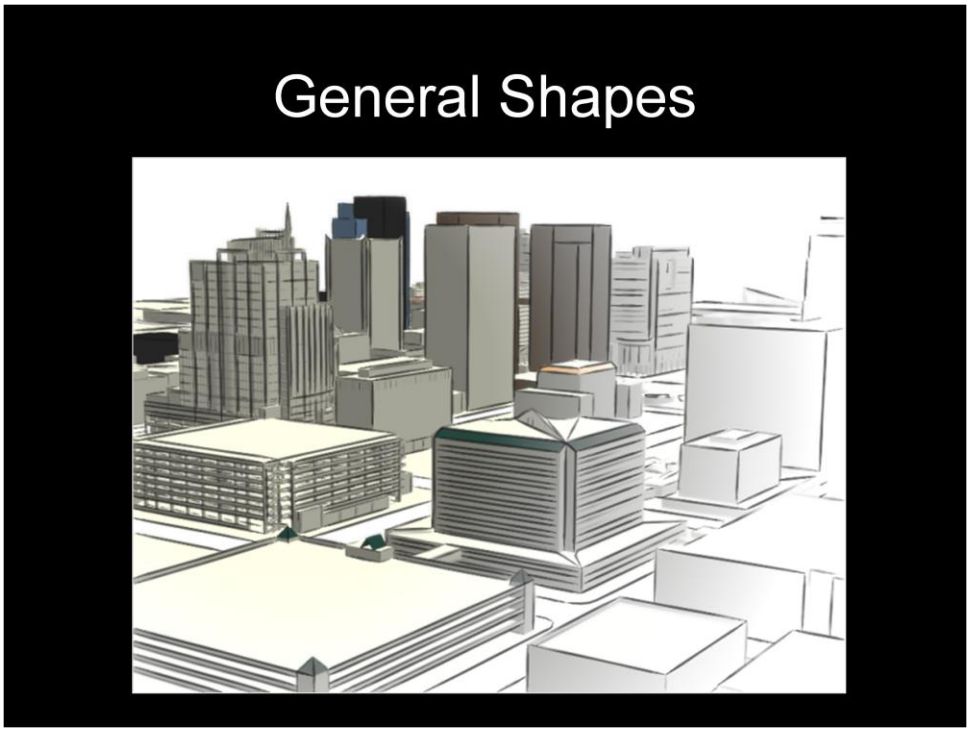
Multi-Resolution Meshes

- Create a set of filtered meshes
- Blend meshes based on viewing distance
- Extract lines from blended mesh
- Only works for smooth shapes

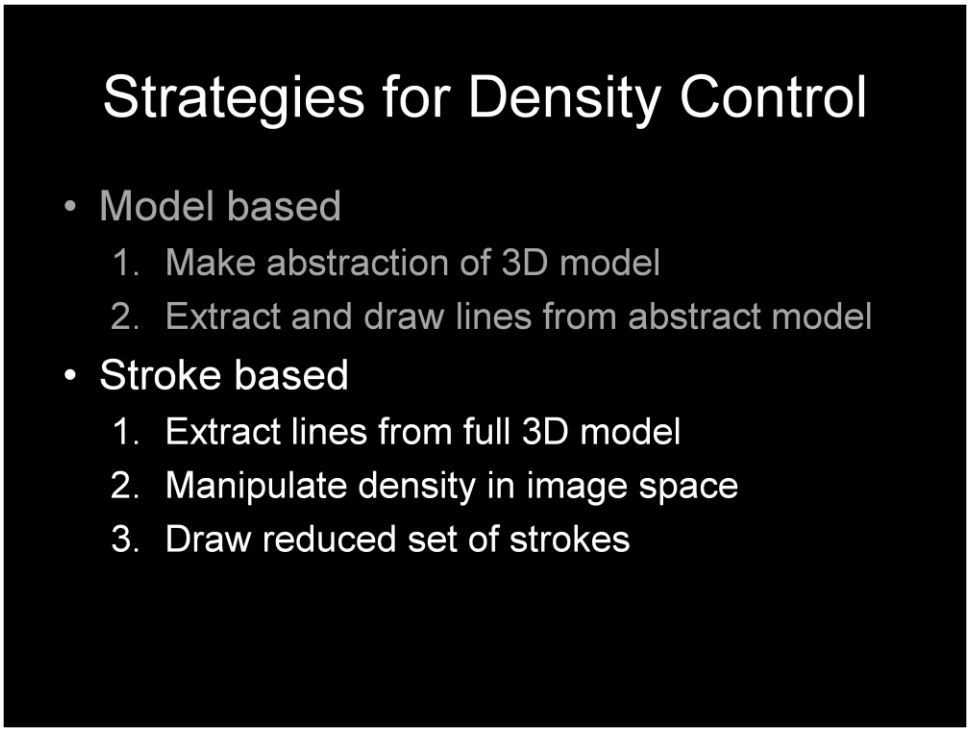


[Jeong 2006]

More generally, we can create a set of simplified meshes using some low-pass filtering method. We then interpolate between these meshes at runtime, giving precidence to the smooth mesh in areas where low density are desired, and the detail mesh for areas of high density. The major issue is that this method is restricted to smooth models that can be effectively filtered without losing their character.



If you have shapes that cannot be filtered easily, such as these buildings, or worse, have a set of lines built in to the model itself (such as the lines on the building in the foreground), then its difficult to use a model based strategy.



A stroke based strategy can be more general because it doesn't depend on changing the model. The density control happens in image space, after the lines are extracted and turned into 2D strokes. The source of the lines is not important, be it extracted silhouettes and suggestive contours or manually added decorative line, because they all get turned into 2D strokes.

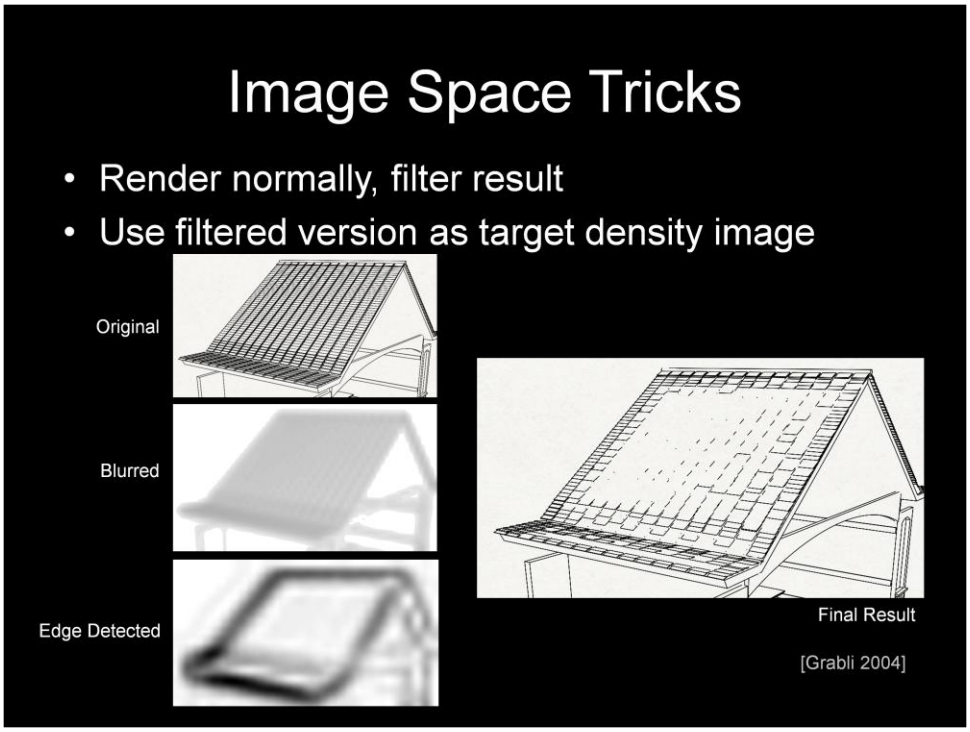
Stroke Based Control

- Compute line density as strokes are drawn
- When density reaches threshold, stop drawing
- Order of strokes is important

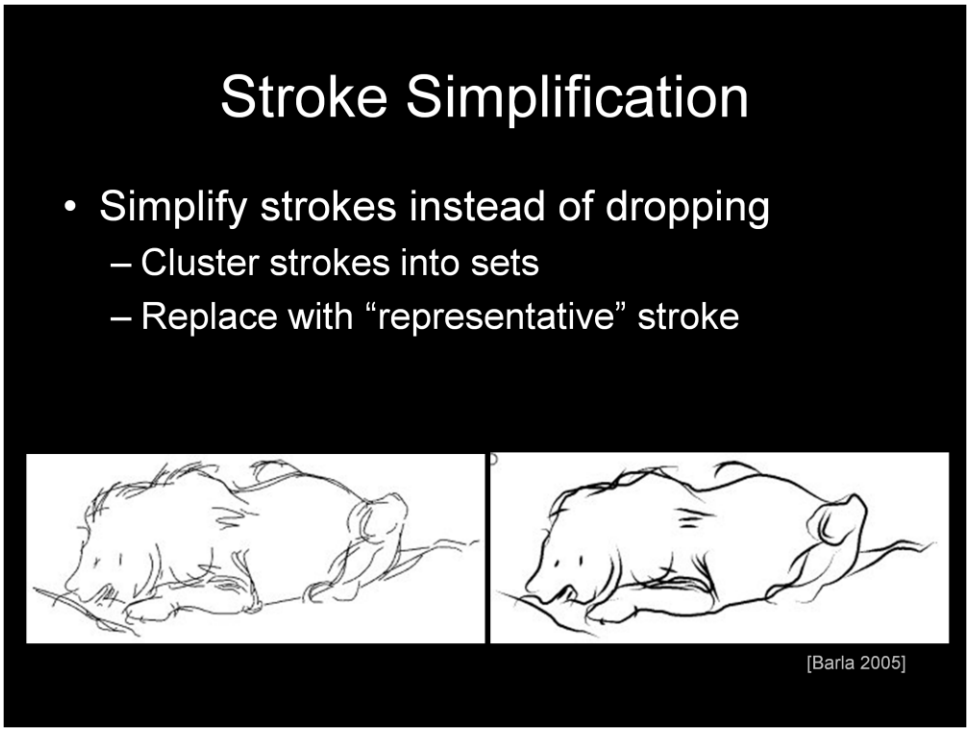
Original | Unordered | Ordered by Depth Importance

[Grabli 2004]

The simplest way to control line density in image space is to keep track of the density as each new stroke is drawn. The density function is usually defined as a smoothed version of the drawing in progress. When the density reaches a user-specificied threshold, the remaining strokes in that area are dropped. The strokes must be ordered somehow by importance, as a random ordering will cause undesired results such as in the middle image. Ordering the strokes by an importance metric can improve the results by making sure the more important strokes (here, the strokes that lie along large depth discontinuties) get drawn first.



There are a number of tricks that can be performed in the image domain. One is to simulate indication by drawing strokes more densly around the edges of the object. This effect can be achieved by first drawing the entire scene and blurring it to get a smooth version. The smooth version is then run through an edge detector, which produces an image with smooth dark areas around the edges of the shape. This image can then be used as a target density image to locally control the stroke density across the image. Unfortunately, dropping strokes at an arbitrary threshold generally leads to terrible temporal coherence. Superior results can be achieved by smoothly merging strokes instead of dropping them.



Techniques to merge dense strokes are generally called stroke simplification techniques. The general idea is to identify a set of similar, nearby strokes, merge them into a cluster, and then replace the cluster with a single representative stroke. The subtly arises from the problem of choosing a representative stroke. Sometimes long, curly strokes (such as the stroke near the lion's shoulder) need to be broken into a series of strokes.

Stroke Simplification

- Naively, at least an $O(n^2)$ problem
- Can be improved with proper datastructure
  - "(1+ε)-deformable Spanner"
  - Roughly $O(n)$ for reasonable drawings

[Shesh 2008]

A naive algorithm for clustering strokes is at least O(n^2), since every stroke needs to be compared to every other stroke to find the closest matches. Shesh 2008 proposes to use an advanced datastructure called a 1+epsilon deformable spanner, which is essentially a graph structure that allows finding all pairs of nearest points in roughly linear time for reasonable graphs. Shesh demonstrates the system at interactive framerates with good temporal coherence. However, the implementation of the system is somewhat complex due to the use of the spanner structure.

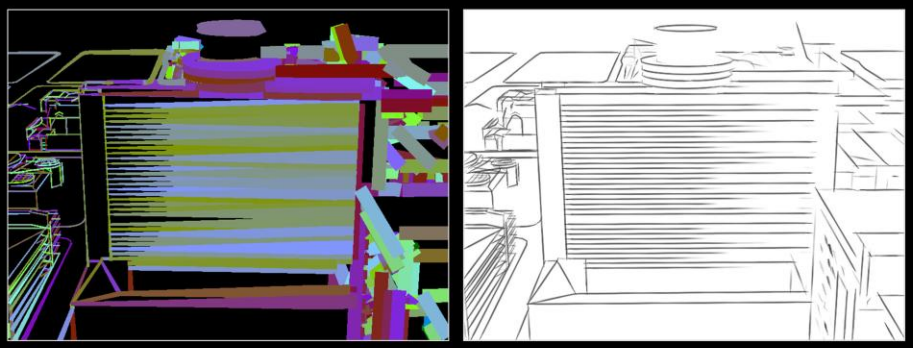
Priority Buffer

- Yet another density control algorithm
  - Inspired by item buffer
- Good temporal coherence
- Stroke based
- Relatively simple
- Fast

The final line density control scheme that I will mention is the priority buffer, which is inspired by the item buffer Adam mentioned before. The priority buffer test has several important benefits. It has temporal coherence suitable for animation. It is stroke based so it handles all types of lines, with the caveat that the priority ordering must be well defined. It is also fast and relatively easy to implement, especially if you are already using an item buffer to compute line visibility.

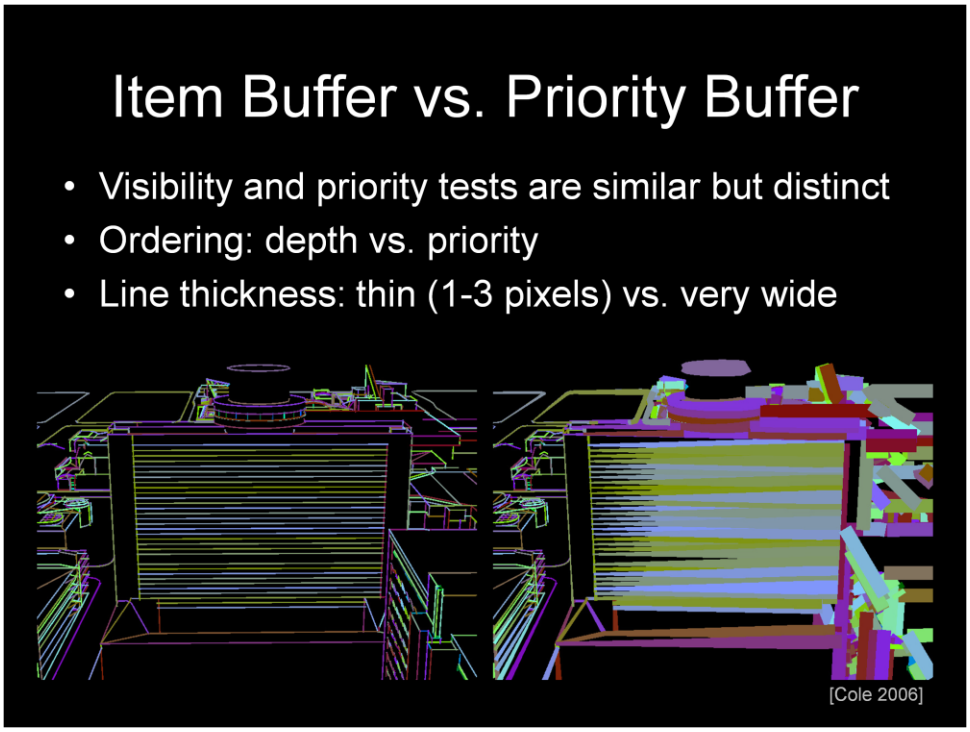
Priority Buffer

- Lines ordered by priority instead of depth
- Wide, high priority lines cover low priority lines
- Final weight proportional to visibility in buffer

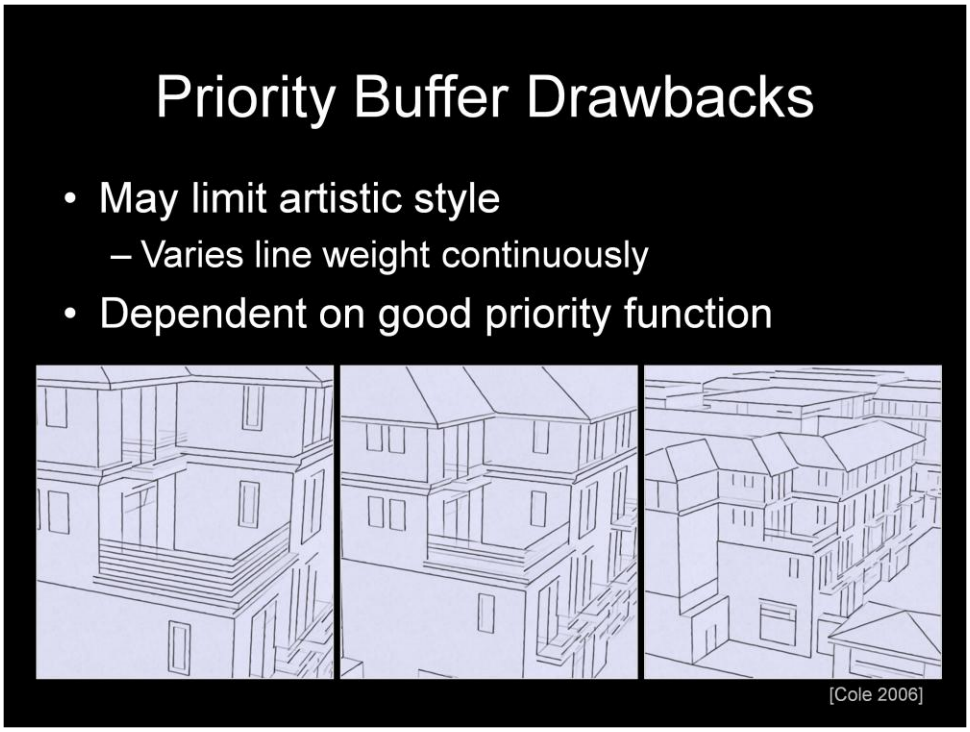


[Cole 2006]

The priority buffer is a second offscreen buffer, where lines are ordered by a priority value rather than depth. We locally vary the width of the lines so that the lines are thin where line density should be high, and wide where line density should be low. The effect is that in areas of low density, wide, high priority lines cover low priority lines and obscure them. We vary the final weight of each line according to its visibility in the priority buffer. So for example, the lines on the left here are thin, corresponding to high density in the final rendering, and the lines on the right are wide, corresponding to low density.



The item buffer and priority buffers are similar, but they are necessarily separate. Of course, the ordering of the lines is by depth in the item buffer and by priority in the priority buffer. Somewhat more subtly, however, lines in the item buffer should be as thin as possible while avoiding rasterization errors. If lines are drawn wide in the item buffer, the visibility test will be inaccurate and we will get haloing and other artifacts. The priority buffer buffer needs wide lines, but it also needs accurate visibility to function, so we have to perform the visibility test first followed by the priority test.



This method also has some drawbacks. To achieve the temporal coherence, we vary the line weight continuously, but this can look odd for styles such as pen and ink.

Second, we need to have a good priority ordering. Currently, we use 3D line length for priority: that is, a line's priority is proportional to its length in world space, prior to any visibility clipping. this function tends to bring out long lines like the rooflines in the right picture. our function can have problems where a large number of nearby lines have exactly the same length. you can see the problem in dense bars in the middle picture. in this case the priority is effectively random.

Of course, our priority function can be arbitrary, and we could choose other functions to, for example, prioritize silhouettes or lines with high depth discontinuity. For really high quality results the user could actually go in and manually set the priority in these kinds of situations.

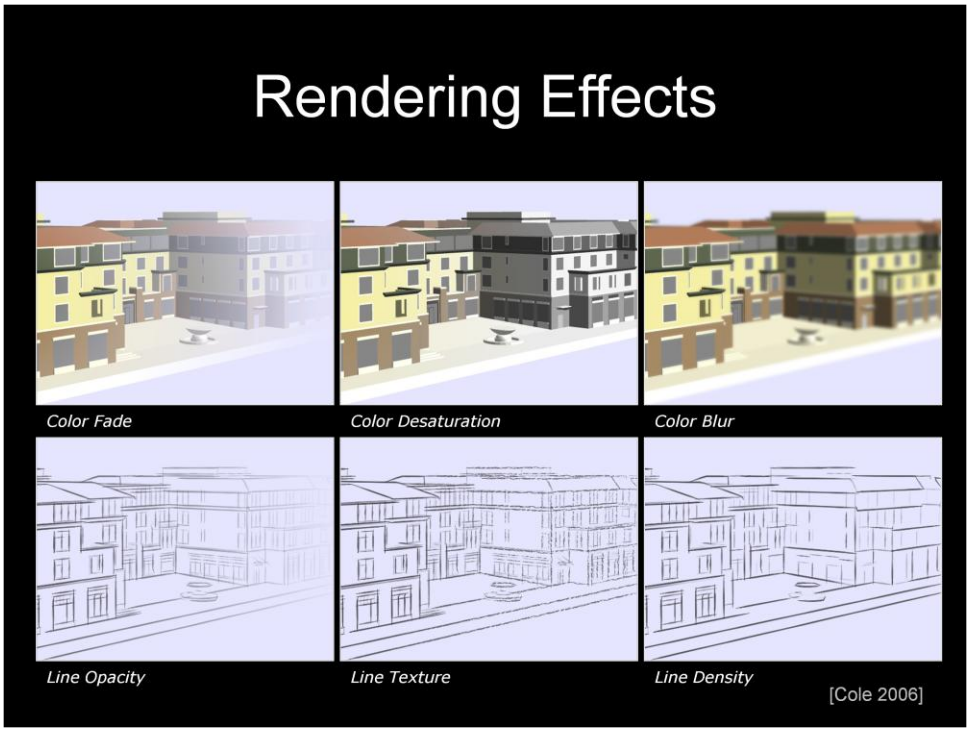
Beyond Line Density

- Line style
- Line texture
- Color saturation
- Color intensity

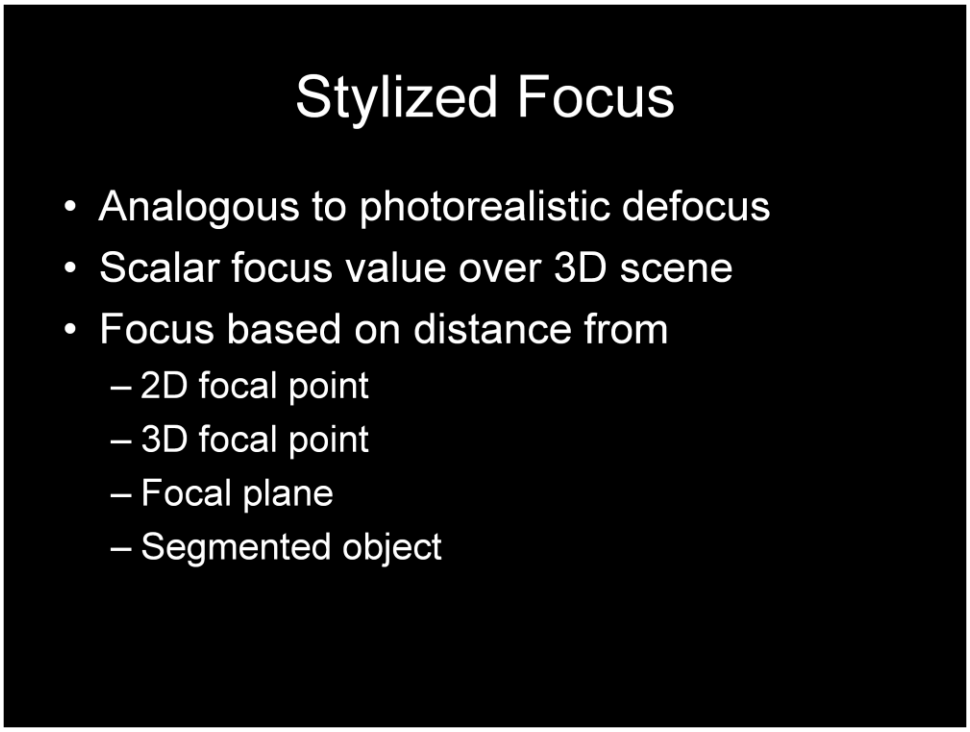
Rembrandt

Winslow Homer

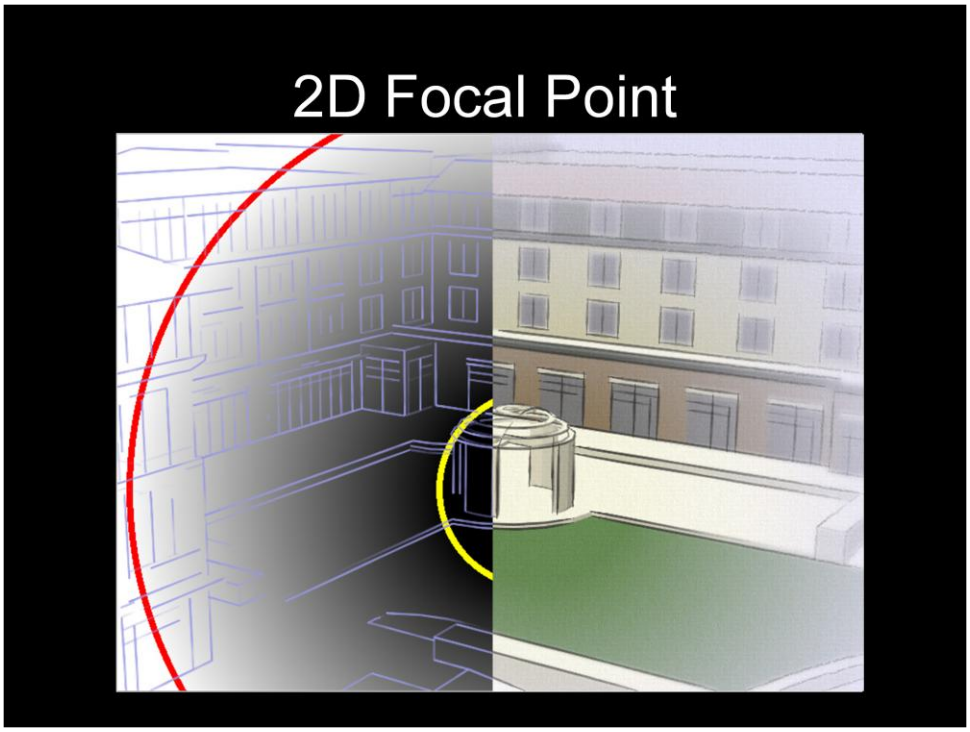
Of course, density is not the only way to control abstraction in line drawings. Line style and texture, such as in the Rembrandt sketch, are important. Color saturation and intensity, while not strictly line drawing techniques, can also be very effective. In the Homer watercolor the region to the right is faded and desaturated, and therefore deemphasized, while the boat right next to it is emphasized withbold colors.



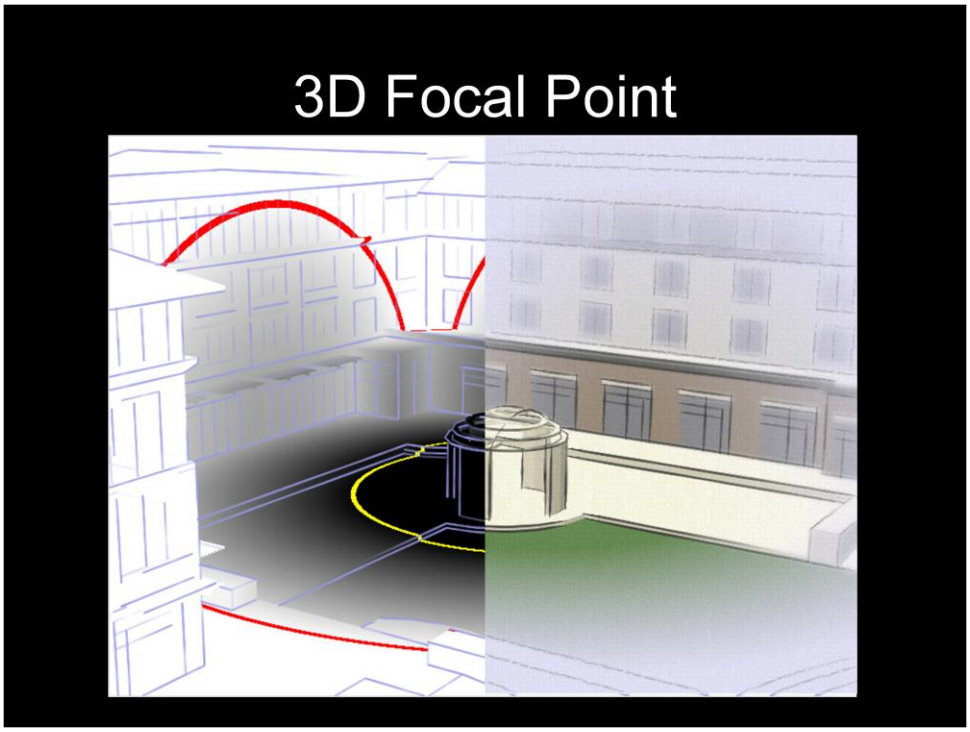
These are the main rendering effects that we use to control abstraction. They can be broken down broadly into color effects and line effects. For color, we can fade out the color away from the point of emphasis, desaturate the color, and blur the color. We can similarly fade out the lines, change the line texture, and locally adjust the line density using the priority buffer.



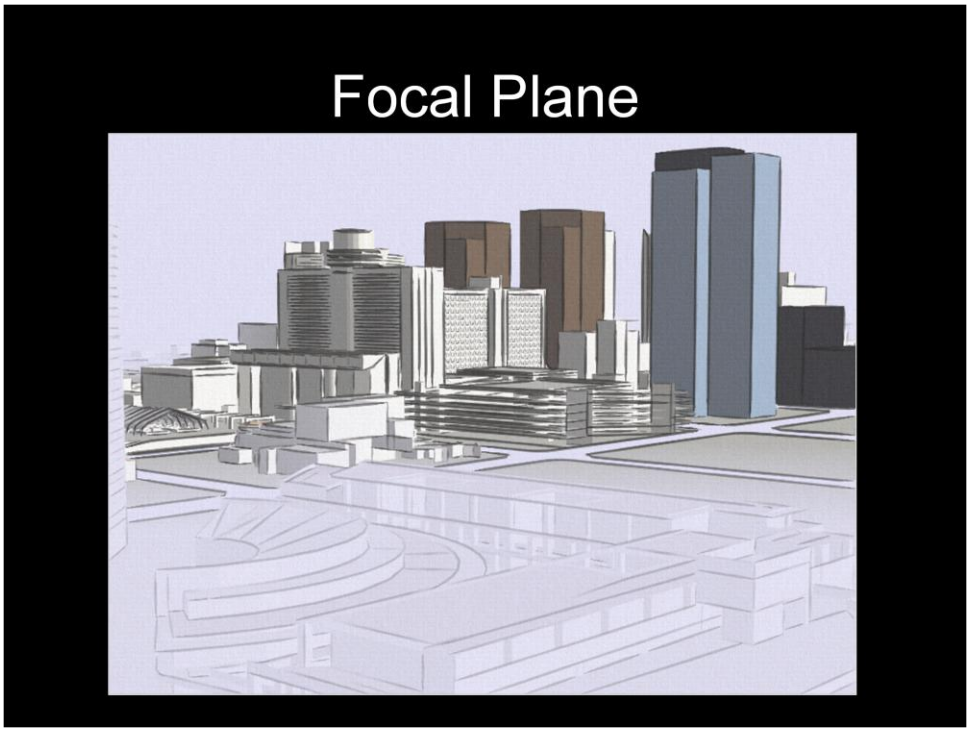
To control all these effects it is helpful to define a single scalar value, which we call stylized focus. We can define the focus value pretty much arbitrarily, so long as it is relatively smooth. In the 2006 paper, we define four methods: distance from a point in image space (2D), distance from a point in world space (3D), distance from a focal plane (meant to simulate camera defocus) and segmentation, where each model section gets a constant focus value.



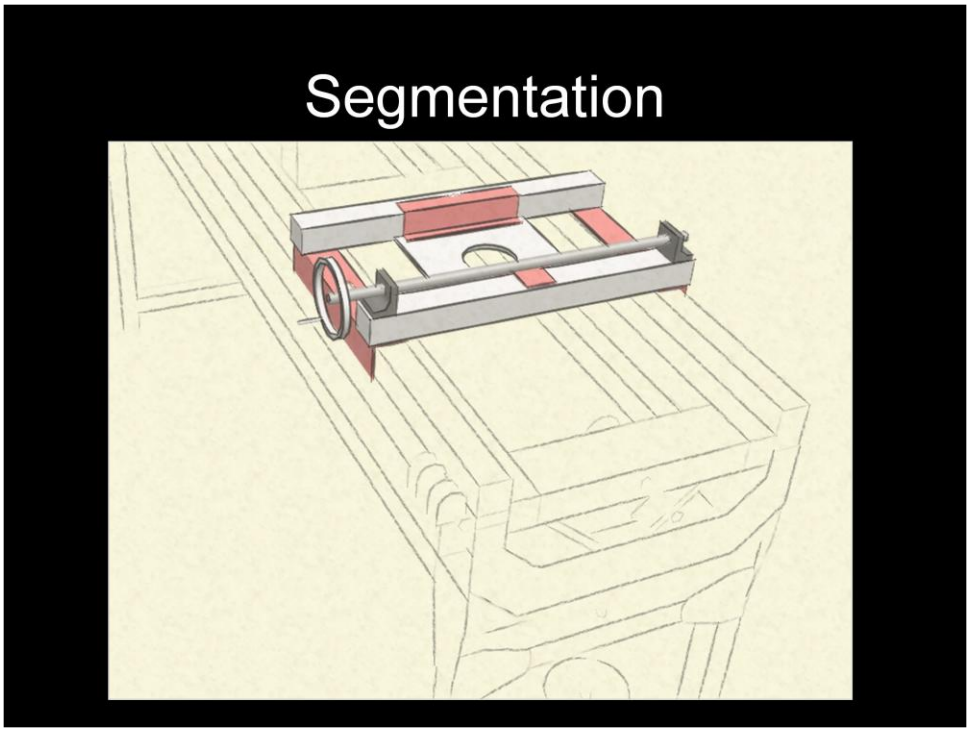
This is an example of the 2D mode, where the focus value falls off linearly from 1 inside the yellow ring to 0 outside the red ring.



The same visualization for the 3D, world space method



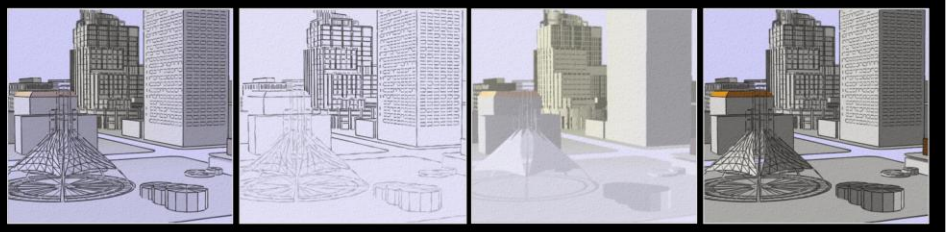
The camera focal place method, where the focal plane is placed in the distance.



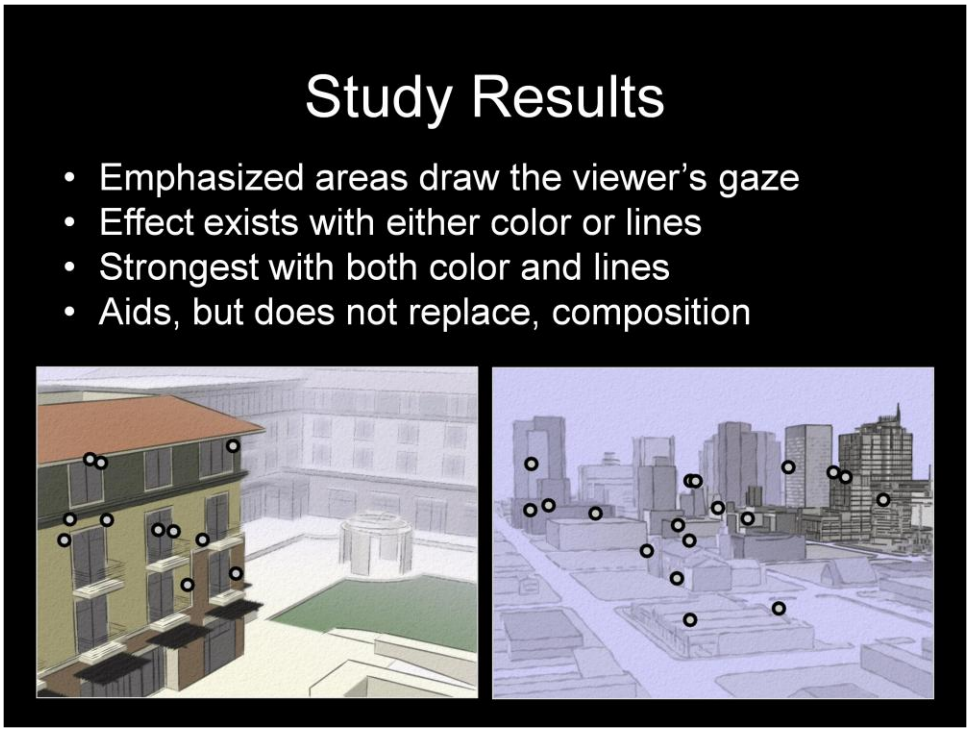
Finally, a simple example of segmentation.

Evaluating Effectiveness

- Eyetracker study similar to [DeCarlo 2002]
- Scenes created to isolate effects
  - Color only, lines emphasized, color constant, etc.
- Compared with and without emphasis



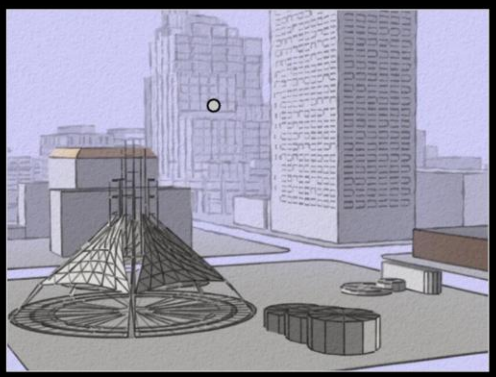
Once we have created the stylized focus effect, a natural question to ask is how well it actually works. We ran a study using the same eye tracking hardware that Doug described earlier. The subjects were shown a range of scenes and effects. The scenes were created to isolate individual rendering effects, so for instance only colors or only lines, or both color and lines, but where only the lines are emphasized. We then compared the eye tracking data for the emphasized versions with unemphasized control images.



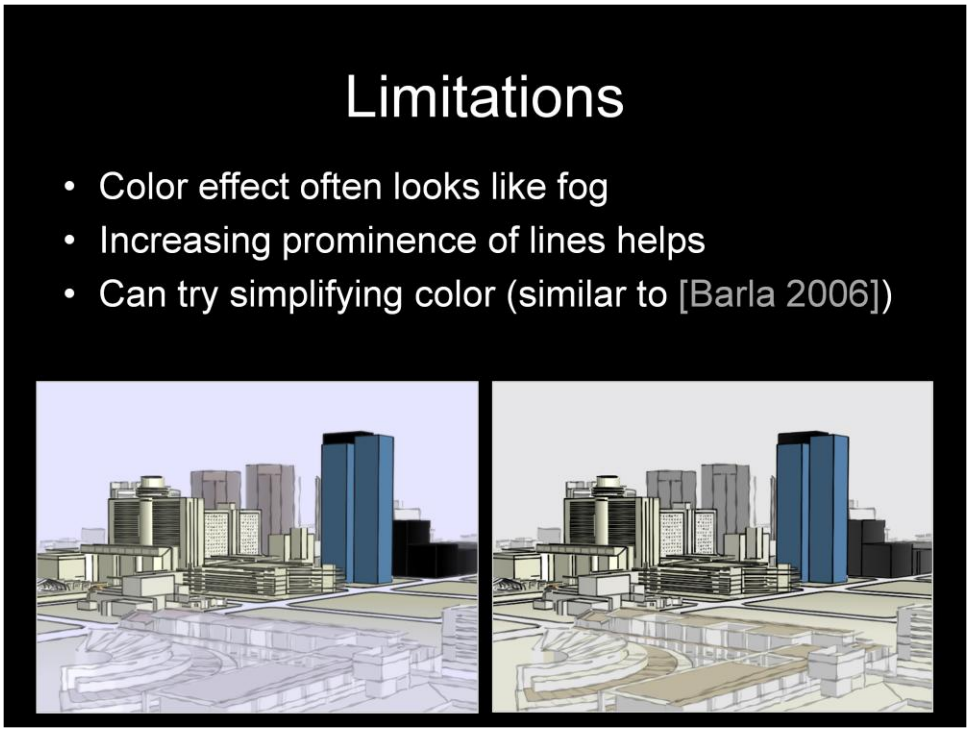
The results show that our emphasis effect does draw the viewers eye in all cases, though the effect can be small. For example, the left image shows a strong case, while the right image shows a weak one. The circles indicate eye fixations. The effect exists when either lines or color are emphasized alone, though it is strongest when both are used together. Composition of the scene is still important – the stylized focus effect does not prevent the viewer from looking at deemphasized areas, as shown by the right image.

Limitations

- No Feedback
  - Some areas require more deemphasis than others
  - No way to know if deemphasis is effective



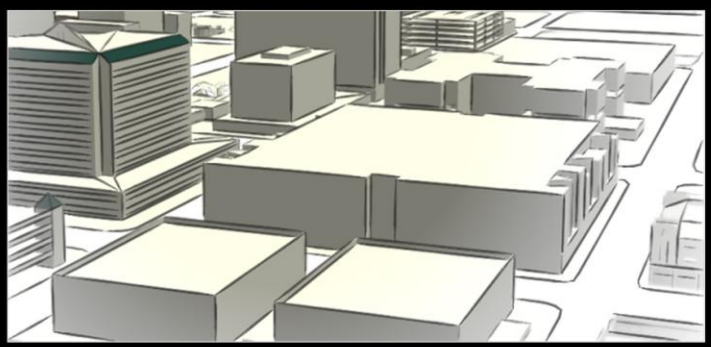
This image shows an eye fixation on a heavily deemphasized object. We are never going to eliminate fixations in deemphasized areas. However, some areas (such as this skyscraper, with lots of line detail) need more deemphasis than others. We currently have no way to tell when this is the case. There are various algorithms for automatically calculating some kind of saliency map for the image. You could imagine running our algorithm, calculating a saliency map from the result, and using that saliency map to determine where further deemphasis was required, and then iterating until the emphasized areas had the desired saliency.



The color fading effect can resemble fog, and look bad in some cases. A better effect can be gained by simplifying the color without fading it out. Basically, we quantize the colors into a couple of desaturated buckets. This effect was also proposed by Barla in relation to toon shading.

Limitations

- Only works if detail already exists
  - Can only “defocus”
  - No way to invent line detail



A big limitation of this, and most other abstraction approaches we examined, is that If there is a big area of the image without much detail cannot be effectively emphasized. Possible solutions to this include using some kind of hatching scheme to add line detail even where there is none originally, though hatching of course affects the tone of the image. This situation may also not arise very often, because it is rare that we want to draw attention to a big, blank area of the scene.

Thank You

Questions?