Question An architect has come up with an award-winning "twisted-tower" design as shown in Fig 1. You are commissioned to generate a 3D graphic model for visualizing such a design. You will use a glutWireCube as the only building primitive. The requirement is that there must be 60 floors. Each floor is in the shape of a rectangular box where the ratio of length to width to height is 100:10:1. The twisted tower must complete two full (360^o) counterclockwise rotation from the 1st floor to the 60th floor. The tower is to be centered at the global origin, with the length aligned with the global x axis and the width aligned with the global y axis (for the 1st floor). The view point must be in the first quadrant, elevated above the tower, and looking down. Recall that `glutWireCube` generates a cube of size 1 centered at the global origin.

- Primitive: glutWireCube(GLdouble size)
- 60 floors, each floor is a rectangular box
  - Length:width:height = 100:10:1
  - Rotate 720 degrees in total, 12/per floor
- First floor:
  - Centered at the global origin
  - Length aligned with X axis
  - Width aligned with Y axis
- gluLookAt()
  - Eye: first quadrant, above and in front of the tower
  - Center: origin
  - Up: Positive Z

Solution 1: Each floor is treated independently

```
for(int i = 0; i < 60; i++) {
        glPushMatrix();
        glTranslated(0.0, 0.0, 1.0 * i);
        glRotated(12 * i, 0.0, 0.0, 1.0);
        glScaled(100.0, 10.0, 1.0);
        glutWireCube(1.0);
        glPopMatrix();
}
```

Solution 2: Construction is to be incremental

```
for(int i = 0; i < 60; i++) {
        glTranslated(0.0, 0.0, 1.0);
        glRotated(12, 0.0, 0.0, 1.0);
        glPushMatrix();
        glScaled(100.0, 10.0, 1.0);
        glutWireCube(1.0);
        glPopMatrix();
}
```

Solution 2: Construction is to be incremental

```
for(int i = 0; i < 60; i++) {
        glTranslated(0.0, 0.0, 1.0);
        glRotated(12, 0.0, 0.0, 1.0);
        glPushMatrix();
        glScaled(100.0, 10.0, 1.0);
        glutWireCube(1.0);
        glPopMatrix();
}
```

drawBox(100.0, 10.0, 1.0);

Solutions

Solution 1:

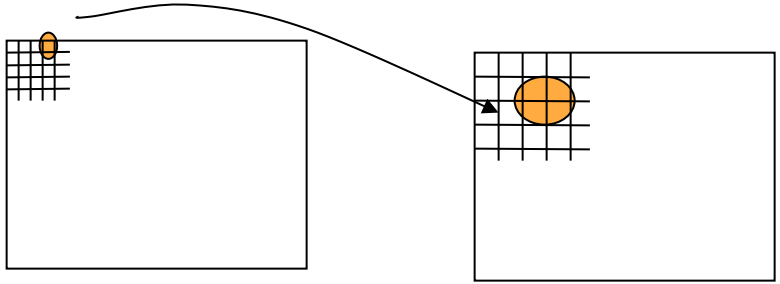
```
for(int i = 0; i < 60; i++) {
        glPushMatrix();
        glTranslated(0.0, 0.0, 1.0 *
i);
        glRotated(12 * i, 0.0, 0.0,
1.0);
        drawBox(100.0, 10.0, 1.0);
        glPopMatrix();
}
```

Solution 2:

```
for(int i = 0; i < 60; i++) {
        glTranslated(0.0, 0.0, 1.0);
        glRotated(12, 0.0, 0.0, 1.0);
        drawBox(100.0, 10.0, 1.0);
}
```

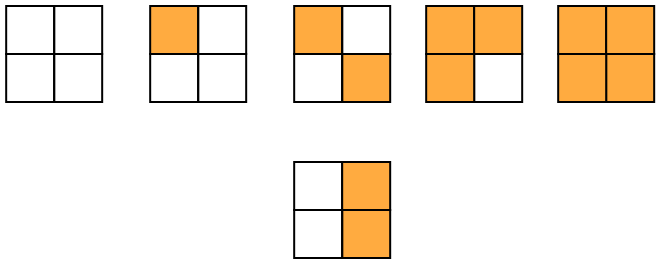
Binary Half Tone

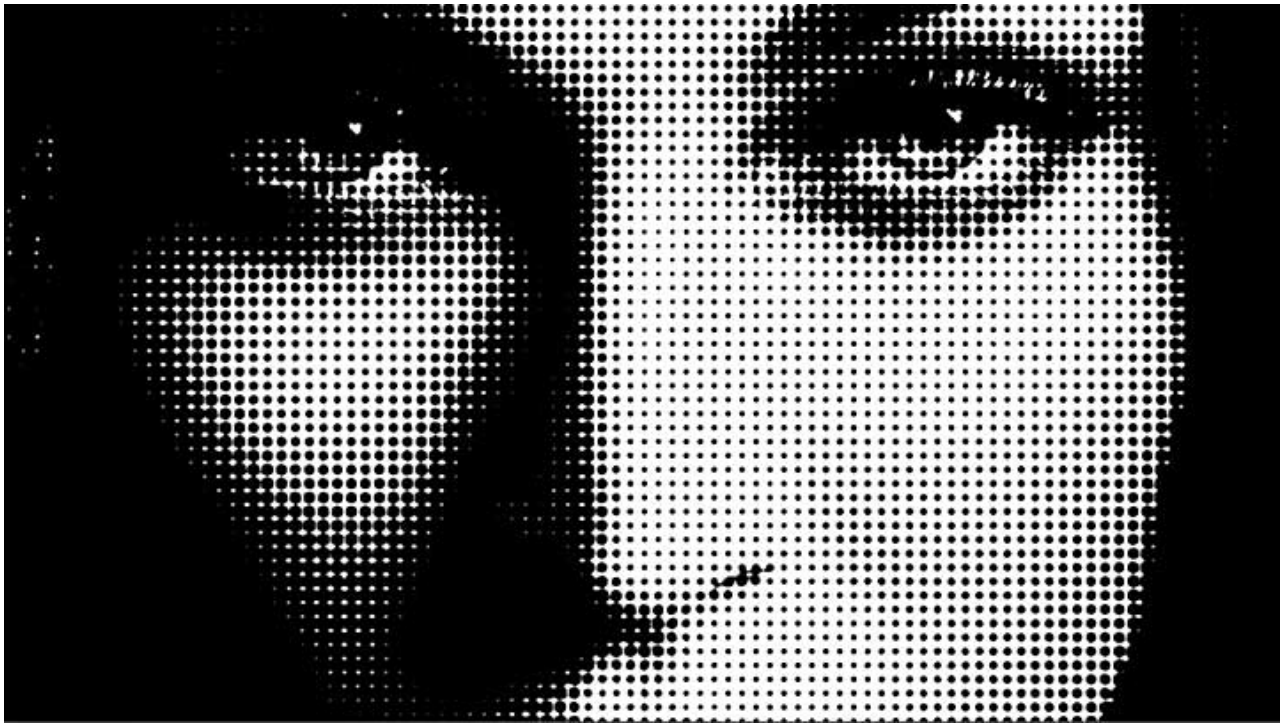
- *High* resolution images to be produced on a low resolution device
  - e.g., a gray scale 8-bit image printed on a B/W paper
- Trade spatial resolution for color resolution



Binary Half Tone

- With a kxk square, k^2+1 intensity levels can be approximated
- Try to avoid artifacts





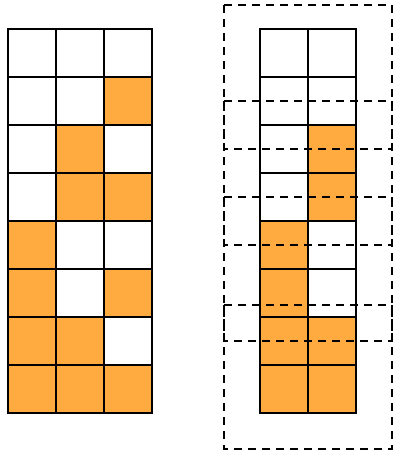
Color Dither

- Bit cut
- Median cut
- etc.

  - input: A[0..m-1][0..n-1] of 2^m;
  - output: B[0..m-1][0..n-1] of 2^n; m>n

Bit cut (Uniform quantization)

- 2^m to 2^n by knocking out the lower (m-n) bits
  - do not adapt to different image contents
  - produce poor results with severe blocking and contouring effects

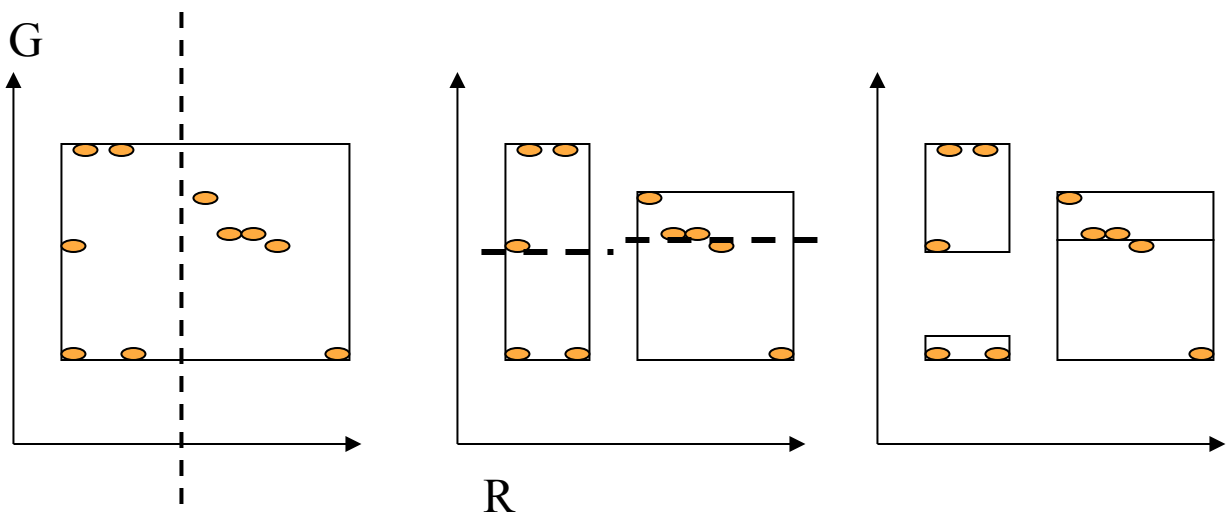


Median Cut

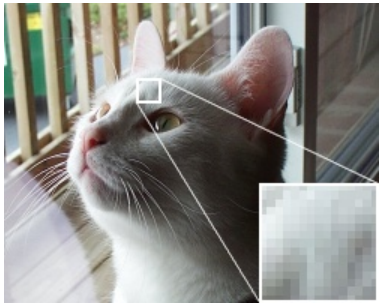
- The quantization should be adaptive depending on the image content

- Usually an image will not have pixel colors distributed uniformly over all visible spectrum

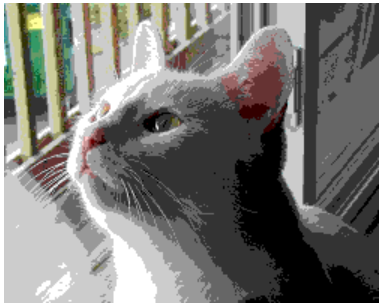
- Reserve more bits for colors which appear more frequently in an image



- At each step
  - select the axis with the largest spread
  - compute median and divide into two groups
- Recursion until C = 2^n boxes
- Use average colors in each box to build lookup table

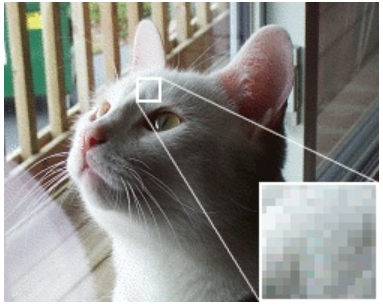


Original photo

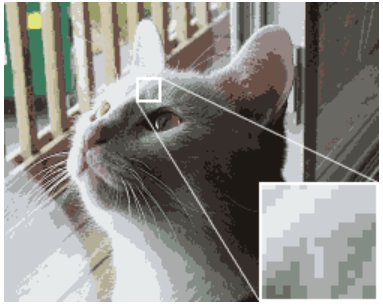


Original image using the web-safe color palette with no dithering applied. Note the large flat areas and loss of detail

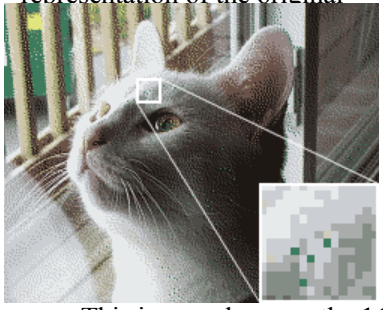
Original image using the web-safe color palette with Floyd-Steinberg dithering. Note that even though the same palette is used, the application of dithering gives a better representation of the original



Original image using the web-safe color palette with Floyd-Steinberg dithering. Note that even though the same palette is used, the application of dithering gives a better representation of the original



Depth is reduced to a 16-color optimized palette in this image, with no dithering. Colors appear muted, and color banding is pronounced



This image also uses the 16-color optimized palette, but the use of dithering helps to reduce banding.

Source: http://en.wikipedia.org/wiki/Dithering