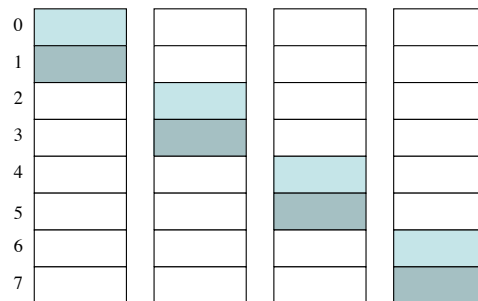
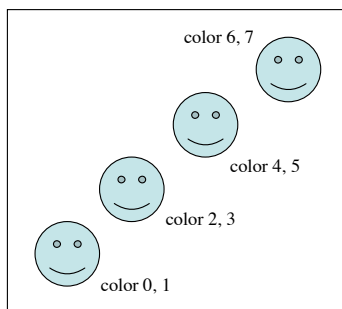


# Animation and Basic Image Manipulation

- Knowing that pixels are ultimately some set of byte/bit values yields a number of fundamental graphics techniques
- Animation
  - Palette-based animation
  - XOR-based animation
  - Sprite animation
  - Full-frame animation ☞ what most computers use today
- Basic image manipulation
  - Color adjustments
  - Brightness and contrast
  - Bit-level effects

## Animation

- Let us count the ways...
- Palette animation
  - Takes advantage of the indexed/indirect method of representing graphics
  - Image stays the same; only the palette changes
  - Older hardware was very good at doing this



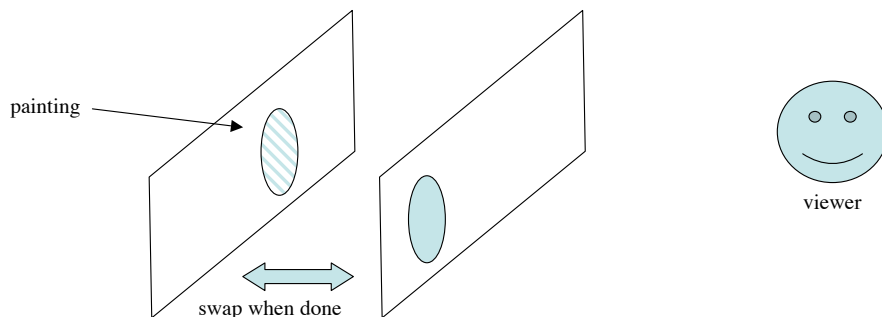
Rotate among these palettes in turn.

## More Animation

- XOR-based animation
  - Takes advantage of the fact that  $((a \text{ xor } b) \text{ xor } b) = a$ 
    - $((0 \text{ xor } 0 = 0) \text{ xor } 0) = 0$
    - $((0 \text{ xor } 1 = 1) \text{ xor } 1) = 0$
    - $((1 \text{ xor } 0 = 1) \text{ xor } 0) = 1$
    - $((1 \text{ xor } 1 = 0) \text{ xor } 1) = 1$
  - Generally works well only on black-and-white graphics
  - Useful for transient effects like rubberbanding
- Sprite animation
  - Small, mini-images are called “sprites”
  - Read target background; paint sprite; paint background over sprite
  - Basis for a whole generation of video games

## Full-Frame Animation (a.k.a. Double Buffering)

- Today’s hardware can do this no sweat, so this is how it’s done pretty much everywhere these days
  - Maintain two “animation frames” — one is on display, and the other one is hidden; each frame corresponds to a block of memory — a buffer
  - While showing the content of one buffer, draw the next frame into the hidden buffer
  - Swap buffers; user can now see the “new” buffer
  - Rinse and repeat



## Full-Frame Animation in GLUT

- GLUT uses this very technique for general drawing. In your sample code, these lines are directly related to full-frame animation:

– in main()

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
```

requests double buffering

requests direct RGB display

– in display()

```
glutSwapBuffers();
```

swaps the buffers; makes the scene that you just drew visible, and makes sure that the next invocation of display() is on the other buffer

– in any code that needs an update

```
glutPostRedisplay();
```

requests that display() be invoked so that the “next” frame can be drawn and displayed

## Basic Image Manipulation

- Since colors are just numbers after all, it stands to reason that manipulating these numbers somehow will result in some recognizable color effects
- “Filtering” — showing only the red, green, or blue elements of an image
- Brightness and contrast — manipulating all three components in a coordinated fashion
- Bit-level effects — combining two images using bit-oriented operations