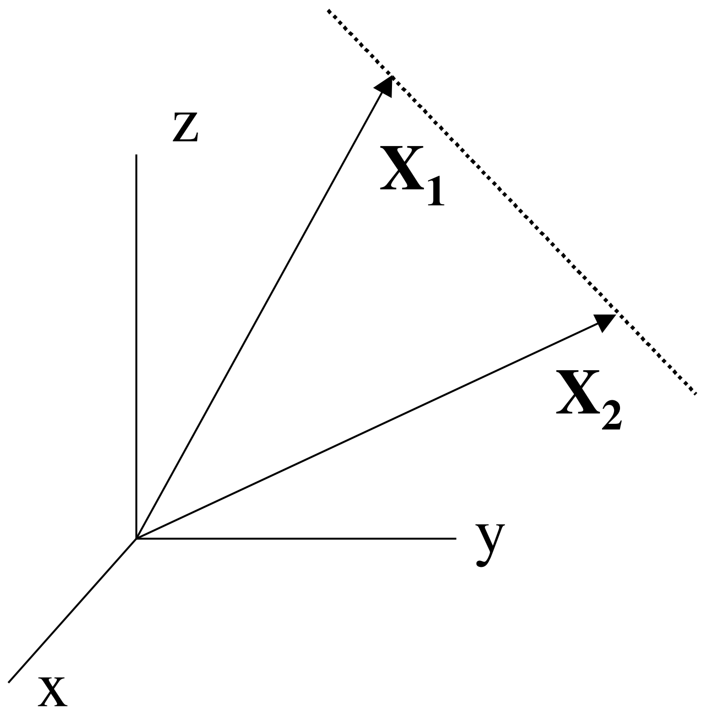
Resources

TA:	Quanfu Fan
Email:	quanfu @ cs
Office Hours:	Monday and Thursday 11-12, BSE.

Web page: www.cs.arizona.edu/classes/433/fall04

Representations for lines and segments

Vector representation



$$t\mathbf{X}_1 + (1 \square t)\mathbf{X}_2$$

Works in any dimension

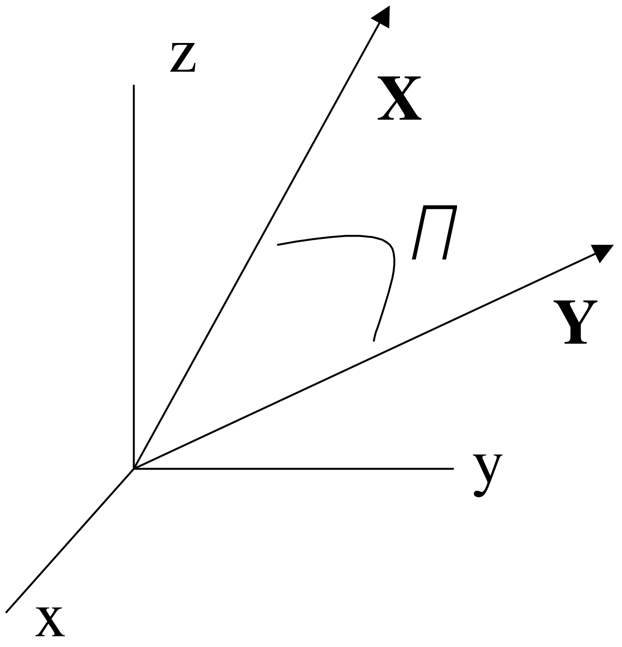
Simplifies representing *segments*

More Vector Operations

Dot Product (any number of dimensions)

More Vector Operations

Dot Product (any number of dimensions)

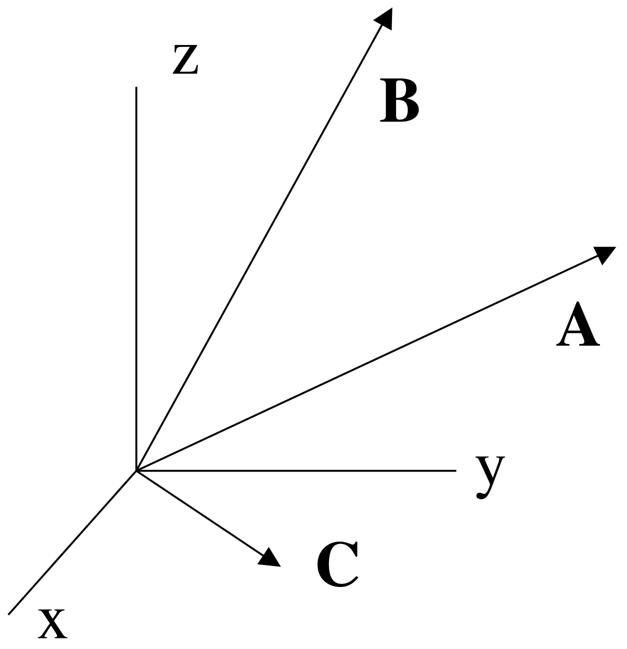


$$\mathbf{X} \bullet \mathbf{Y} = (x_1 y_1 + x_2 y_2 + x_3 y_3)$$
$$= |\mathbf{X}||\mathbf{Y}|\cos\theta$$

Orthogonal $\Leftrightarrow$ $\mathbf{X} \bullet \mathbf{Y} = 0$

More Vector Operations

Vector (cross) product (3D)



$$\mathbf{C} = \mathbf{A} \times \mathbf{B}$$

$\mathbf{C} \perp \mathbf{A}$ *and* $\mathbf{C} \perp \mathbf{B}$

Use Right Hand Rule

$$|\mathbf{C}| = |\mathbf{A}||\mathbf{B}|\sin\theta$$

$$\begin{bmatrix} \mathbf{C_x} \\ \mathbf{C_y} \\ \mathbf{C_z} \end{bmatrix} = \begin{bmatrix} \mathbf{A_y B_z} - \mathbf{A_z B_y} \\ \mathbf{A}_z \mathbf{B}_x - \mathbf{A}_x \mathbf{B}_z \\ \mathbf{A}_x \mathbf{B}_y - \mathbf{A}_y \mathbf{B}_x \end{bmatrix}$$

Representations for planes (1)

A plane passes through a point and has a given “direction”

Representations for planes (1)

A plane passes through a point and has a given “direction”

Direction of plane is given by its normal

$$(\mathbf{X} - \mathbf{X_0}) \bullet \hat{\mathbf{n}} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{ax + by + cz = k}$$

A half space is defined by $(\mathbf{X} - \mathbf{X_0}) \bullet \hat{\mathbf{n}} \geq 0$

Representations for planes (2)

Three points determine a plane

(Can make it the same as previous approach---how?)

Direct vector representation

Representations for planes (2)

Three points determine a plane

(Can make it the same as previous approach---how?)

Direct vector representation

$$v(uA + (1 - u)B) + (1 - v)C$$

$$t = uv \quad and \quad s = v$$

$$C + t(A - B) + s(B - C)$$

(linear combination of two vectors, offset by another)

Typical Graphics Problems

Which side of a plane is a point on?

Is a 3D point in a convex 2D polygon?

OpenGL and GLUT

- Layer between your program and lower levels (hardware, low level display issues)
- Provides primitives
  - points
  - lines
  - polygons
  - bitmaps, fonts
- Provides standard graphics facilities
  - We will learn how some of these work. Some assignments will therefore have some routines “out of bounds”
  - GLUT simplifies interactive program development with intuitive callbacks and additional facilities (menus, window management).

OpenGL and GLUT

Demo and discussion of example program

http://www.cs.arizona.edu/classes/cs433/fall04/triangle.c

OpenGL and GLUT

- Initialization code from the example

```
/* initialize GLUT system */
glutInit(&argc, argv);

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
glutInitWindowSize(400,500);          /* width=400pixels height=500pixels */
win = glutCreateWindow("Triangle");   /* create window */

/* From this point on the current window is win */

/* set background to black */
glClearColor((GLclampf)0.0,(GLclampf)0.0,(GLclampf)0.0,(GLclampf)0.0);
gluOrtho2D(0.0,400.0,0.0,500.0); /* how object is mapped to window */
```

OpenGL and GLUT

- Window display callback. You will likely also call this function. Window repainting on expose and resizing is done for you

```
/* set window's display callback */
glutDisplayFunc(display_CB);
```

```
static void display_CB(void)
{
    glClear(GL_COLOR_BUFFER_BIT);         /* clear the display */

    /* set current color */
    glColor3d(triangle_red, triangle_green, triangle_blue);

    /* draw filled triangle */
    glBegin(GL_POLYGON);

    /* specify each vertex of triangle */
    glVertex2i(200 + displacement_x, 125 - displacement_y);
    glVertex2i(100 + displacement_x, 375 - displacement_y);
    glVertex2i(300 + displacement_x, 375 - displacement_y);

    glEnd();           /* OpenGL draws the filled triangle */
    glFlush();         /* Complete any pending operations */

    glutSwapBuffers(); /* Make the drawing buffer the frame buffer
                          and vice versa */
}
```

OpenGL and GLUT

- User input is through callbacks, e.g.,

```
/* set window's key callback */
glutKeyboardFunc(key_CB);

/* set window's mouse callback */
glutMouseFunc(mouse_CB);

/* set window's mouse move with button pressed callback */
glutMotionFunc(mouse_move_CB);
```

```
static void key_CB(unsigned char key, int x, int y)
{
    if( key == 'q' ) exit(0);
}

/*  /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\   */

/* Function called on mouse click */
static void mouse_CB(int button, int state, int x, int y)
{
    /*
     *  Code which responses to the button, the state (press, release), and where
     *  the pointer was when the mouse event occured (x, y).
     *
     *  See example on-line for sample code.
    */
}

/*  /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\ /\   */

/* Function called on mouse move while depressed. */
static void mouse_move_CB(int x, int y)
{
    /* See example on-line for sample code. */
}
```

OpenGL and GLUT

- GLUT makes pop-up menus easy. We will save development time by using (perhaps abusing) this facility.

```
/* Create a menu which is accessed by the right button. */
submenu = glutCreateMenu(select_triangle_color);
glutAddMenuEntry("Red", KJB_RED);
glutAddMenuEntry("Green", KJB_GREEN);
glutAddMenuEntry("Blue", KJB_BLUE);
glutAddMenuEntry("White", KJB_WHITE);
glutCreateMenu(add_object_CB);
glutAddMenuEntry("Triangle", KJB_TRIANGLE);
glutAddMenuEntry("Square", KJB_SQUARE);
glutAddSubMenu("Color", submenu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

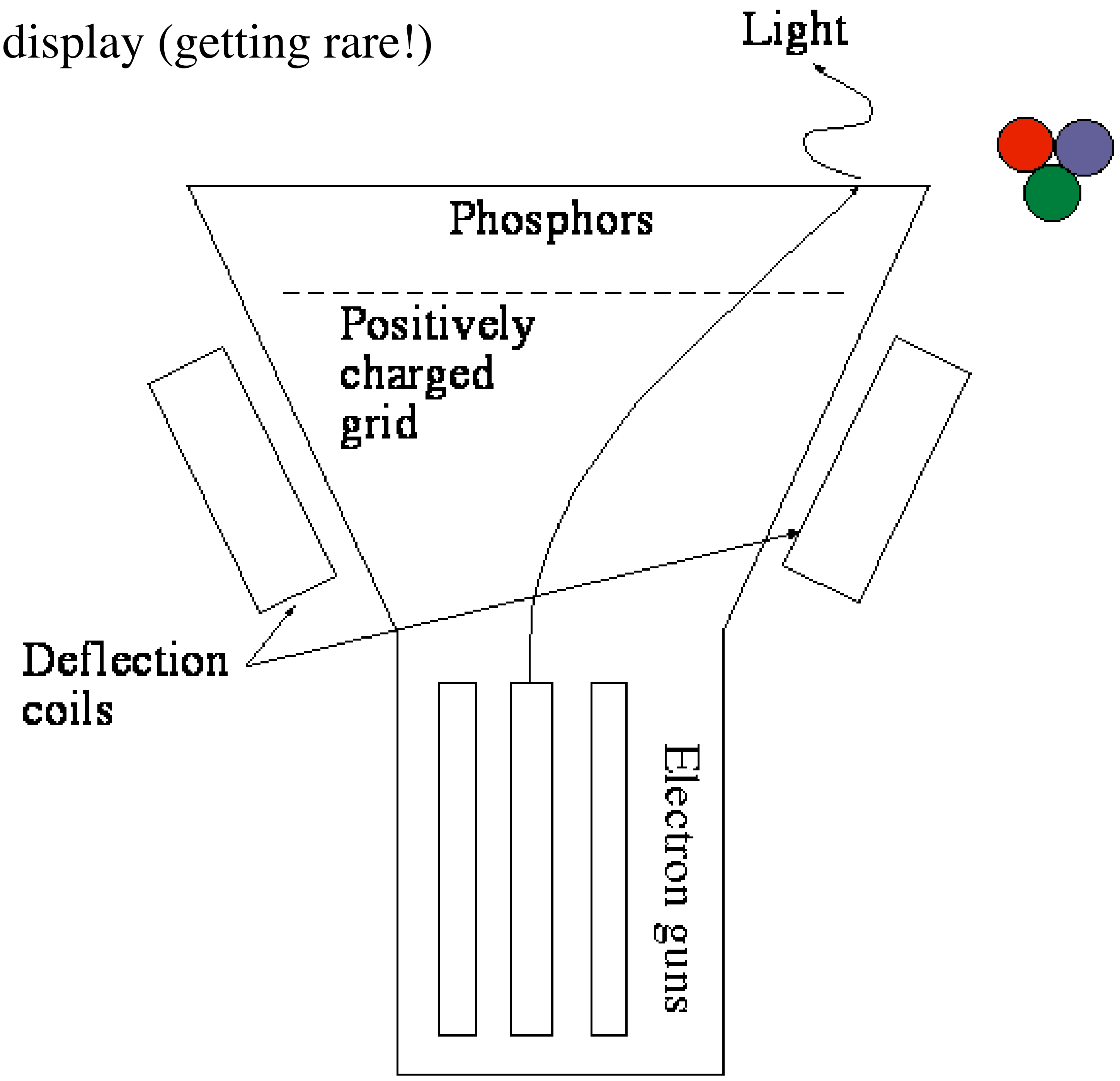
OpenGL and GLUT

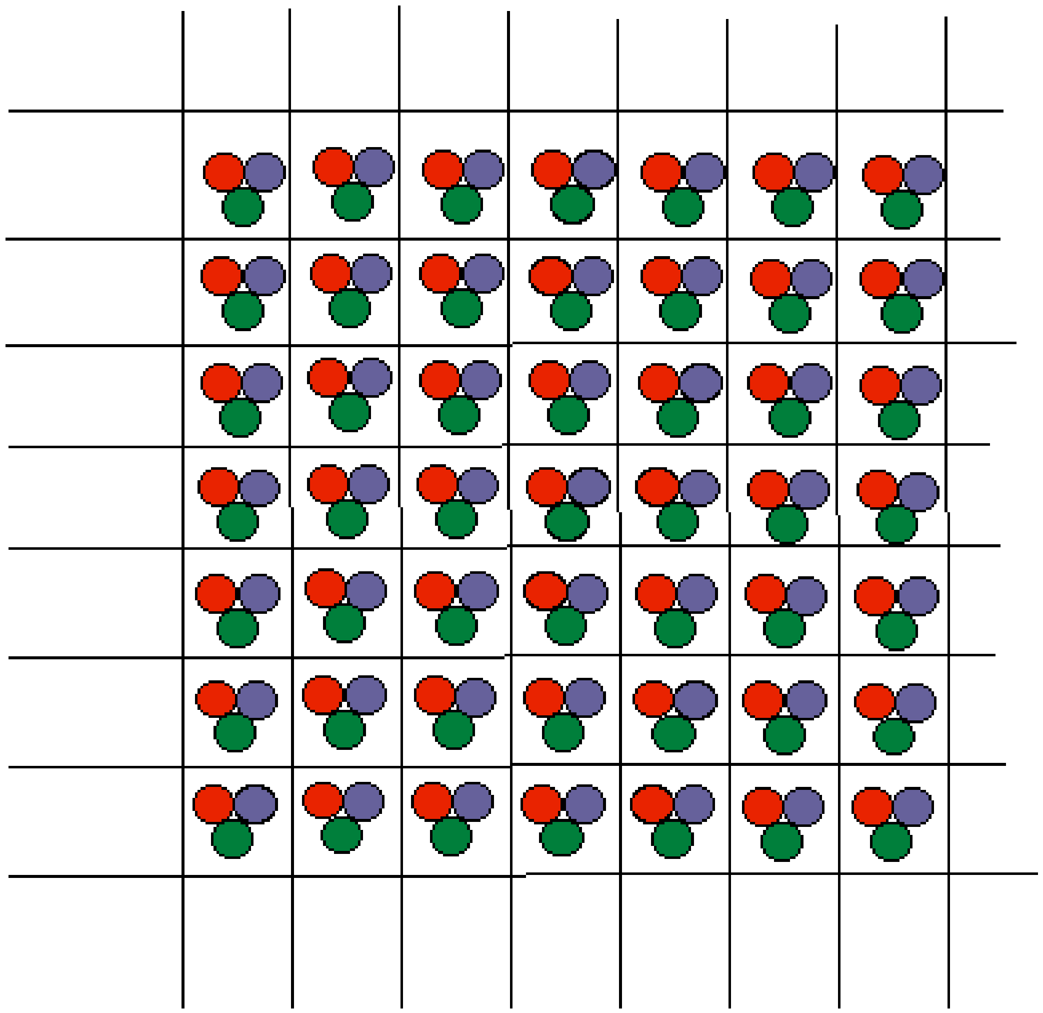
- Ready for the user!

```
/* start processing events... */
glutMainLoop();
```

- For the rest of the code see
  http://www.cs.arizona.edu/classes/cs433/fall03/triangle.c

CRT display (getting rare!)



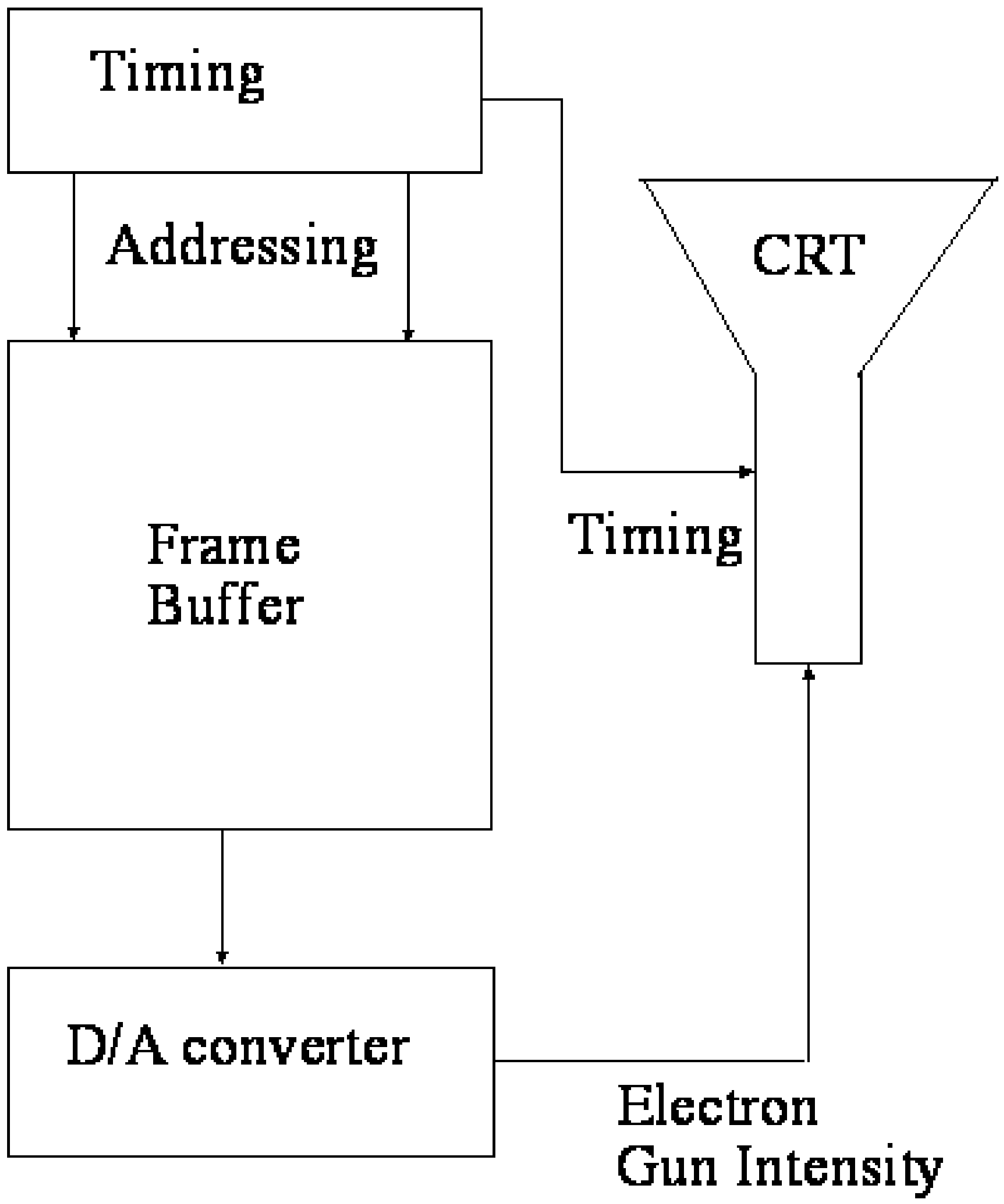


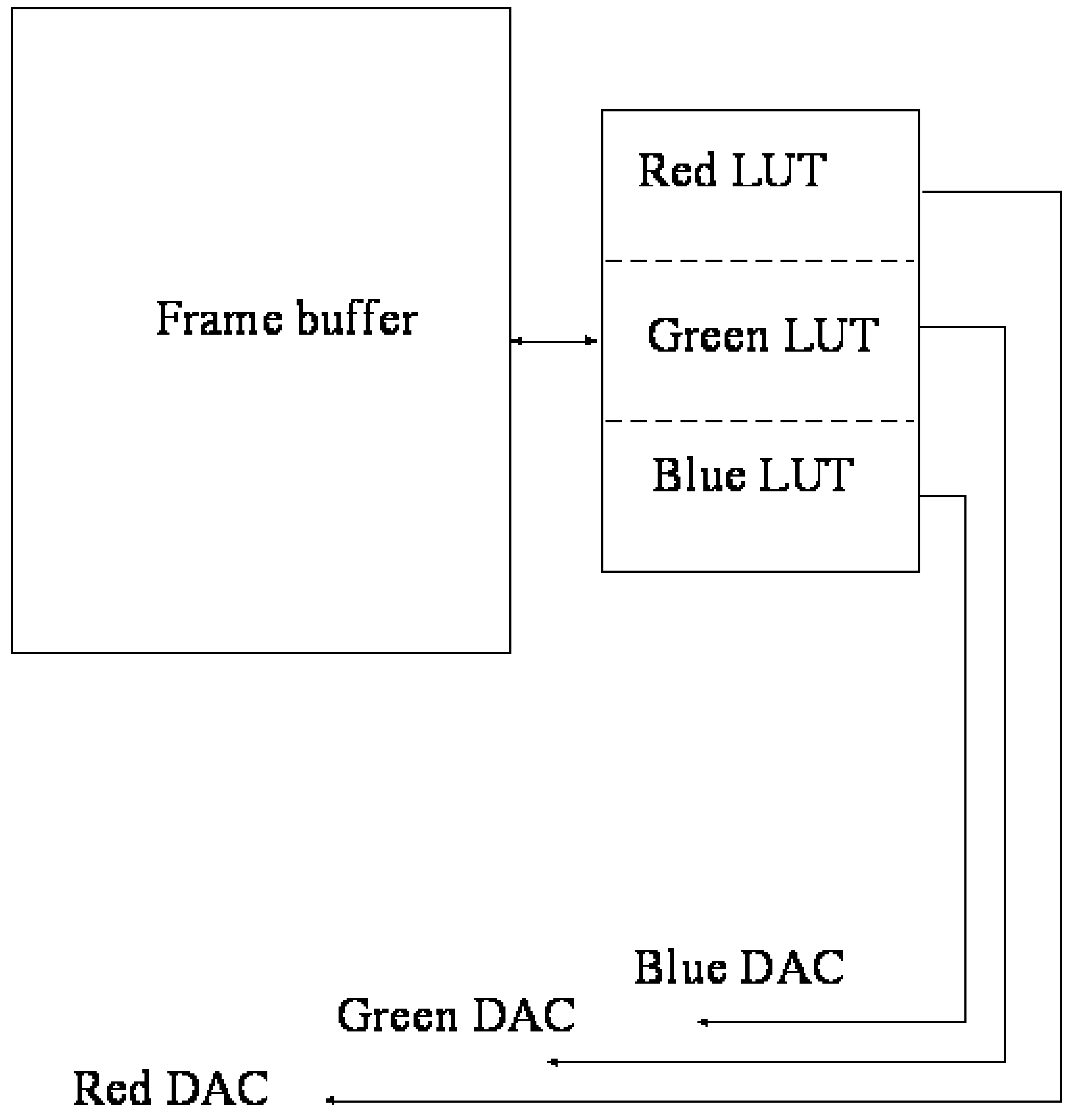
CRT Displays

- Phosphors glow when hit by electron beam.
- Color is adjusted via intensity of beam delivered to each of R,G, and B phosphor
- CRT display phosphors glow for limited time--need to be refreshed
- Raster displays refresh by scanning from top to bottom in left right order.
- Timing is used to make screen elements correspond to memory elements.

CRT Displays

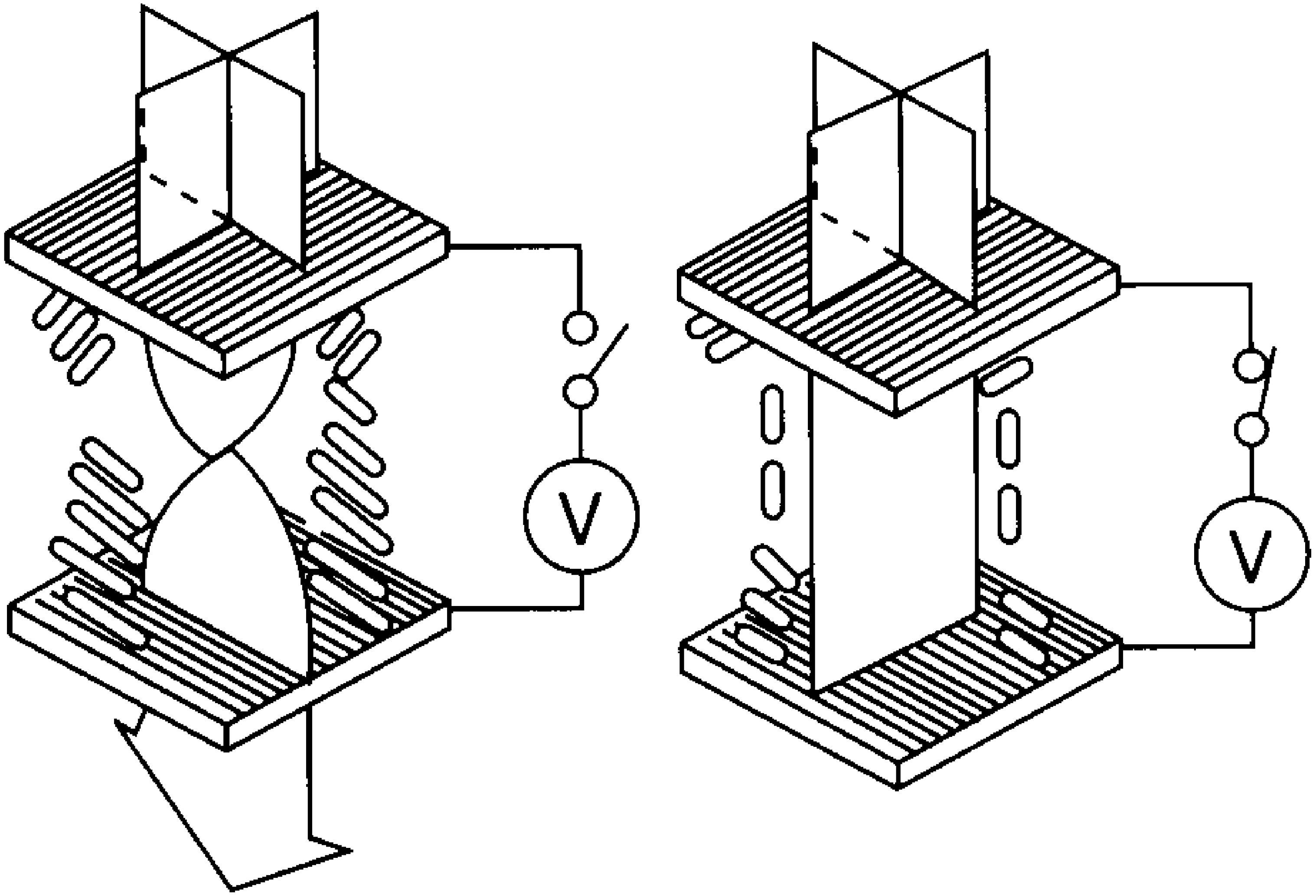
- Typical refresh rate is 75 per second
- May have many phosphor dots corresponding to one memory element (old stuff), but more usually one per phosphor trio.
- Memory elements called pixels
- Refresh method creates architectural and *programming* issues (e.g. double buffering), defines “real time” in animation.


Timing
Addressing
Frame Buffer
D/A converter
CRT
Timing
Electron Gun Intensity

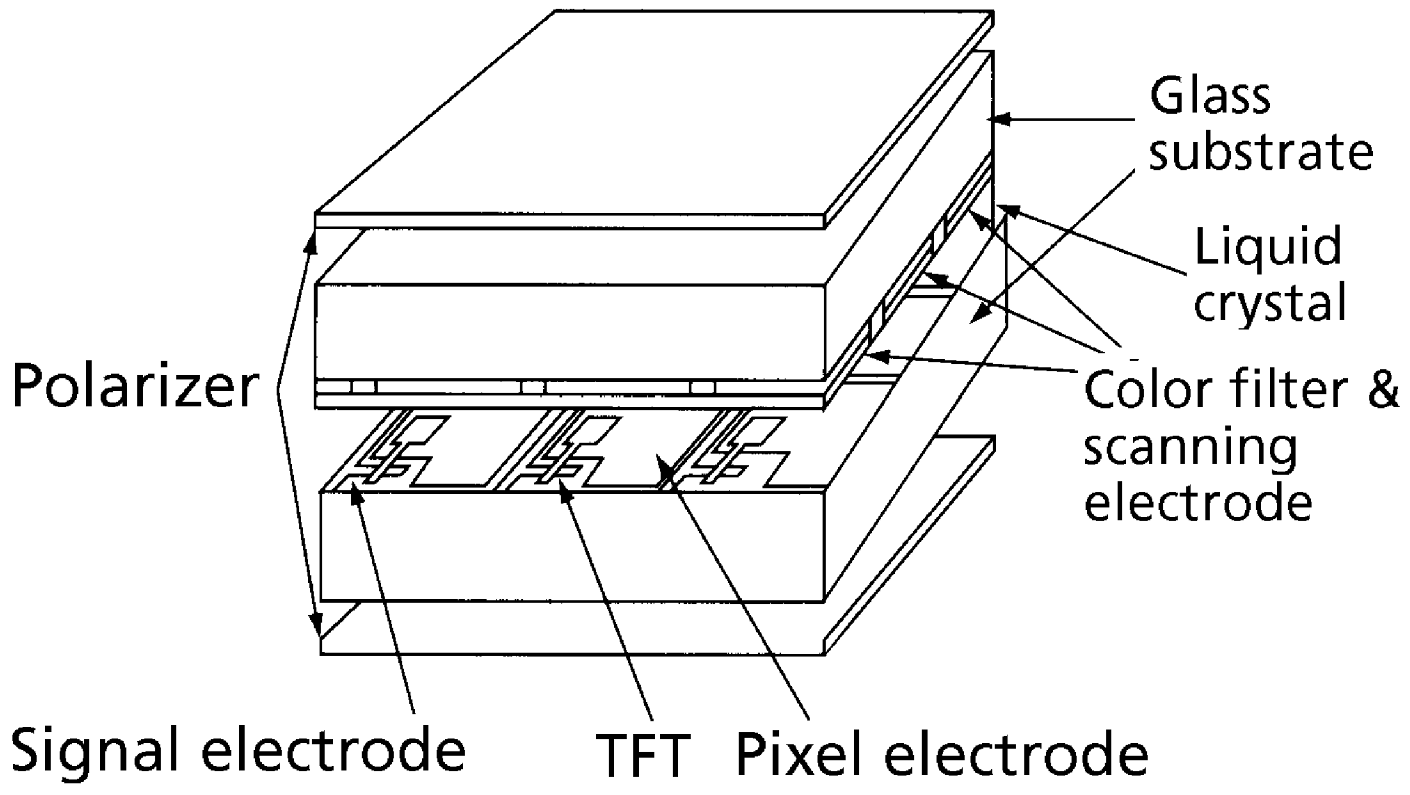

Frame buffer
Red LUT
Green LUT
Blue LUT
Blue DAC
Green DAC
Red DAC

Flat Panel TFT* Displays

*Thin film transistor



From http://www.atip.or.jp/fpd/src/tutorial



From http://www.atip.or.jp/fpd/src/tutorial

3D displays

Use some scheme to control what each eye sees

- Color, temporal + shutter glasses, polarization + glasses