of the selected triangular zones. The intersecting lines **72, 74** selected for such calculation are those lines which are closest to perpendicular.

[**0116**] Virtual Beams

[**0117**] By simple geometric logic, it is possible to detect positions in between hardware-defined beams. With reference to **FIG. 15**, a triangular zone has six beams **56** defined by six LEDs **24** and an IrDA receiver **42**. Five additional "virtual beams"**58** are defined (**1'-5'**), each one midway between adjacent hardware beams **56**. Depending on which hardware beams **56** are blocked, either a hardware beam or a virtual beam is identified as the "centroid", or central logical beam to be used in calculating the touch position. If an odd number of hardware beams **56** are blocked, the center hardware beam is the output. However, if an even number of hardware beams **56** is blocked, a virtual beam **58** in between the two central hardware beams is the output. For example, in **FIG. 15**, a finger **60** blocks beams **2, 3, 4**, and **5**. The touchframe firmware then identifies a virtual beam **58** between beams **3** and **4** (labeled **3'**) and registers that as the output for this triangular zone.

[**0118**] Slope and Endpoint Tables and Refraction

[**0119**] There is not enough memory space in the PCBA **16** to accommodate an exhaustive table lookup for translating the coordinates of intersecting beam pairs into a Cartesian coordinate system. However, the geometric arrangement of the LEDs **24** and IrDA receivers **42** is very regular and symmetrical. Consequently, it is possible to use only two tables along with some mathematical "mirroring" to define the slopes of LED beams comprising all twelve triangular zones. One table defines the slopes of all LED beams comprising the horizontal triangles and another table defines the slopes of all LED beams comprising the vertical triangles. Horizontal triangles are those triangles comprised of a left side **32** (**FIG. 4**) or right side **34** row of LEDs and their associated IrDA receivers **42** on the opposite sides. Vertical triangles are those triangles comprised of a top **26** or bottom **28** row of LEDs and their associated IrDA receivers **42** on the opposite side.

[**0120**] There is a third table that specifies the endpoints of all the lines in all the triangles. Because all lines in each triangle converge on a single point, i.e. the IrDA receiver **42** for that triangle, there are only twelve entries in the endpoint table.

[**0121**] As light travels from a particular LED **24** to its corresponding IrDA receiver **42**, it must pass through the infrared filter bezel that seals and protects the PCBA **16** from the external environment. As it does so, it is refracted to varying degrees depending on the angle of the particular light path and the thickness, angle, and refractive index of the bezel. The effects of this refraction have been calculated and compensated for to the greatest degree possible. The results of this compensation are entirely represented in the data entries in the slope and endpoint tables.

[**0122**] Floating Point Algorithms

[**0123**] When touch events in overlapping triangular zones occur, the touchframe firmware translates those coordinates to Cartesian coordinates using floating point mathematics. As shown in **FIG. 16**, the fundamental equation involved calculates an X/Y output coordinate corresponding to the intersection **62** of two sloped lines **64, 66**. The essential elements input to the equation are the slope and one endpoint for each of the two intersecting lines.

[**0124**] The following is a representation in the 'C' programming language of the floating point coordinate conversion algorithm:

```
float line1__x1,line1__y1; // End one of first line
float line1__x2,line1__y2; // End two of first line
float line2__x1,line2__y1; // End one of second line
float line2__x2,line2__y2; // End two of second line
float xi,yi; // Coordinates of intersection
void convert(void)
{
float slope1,slope2,a1,a2;
   slope1=(line1__y2-line1__y1)/(line1__x2-line1__x1);   //Get slope of line1 (done by table
                                                          lookup in firmware)
   a1=line1__y1-slope1*line1__x1;                        //Compute intercept of line1
   slope2=(line2__y2-line2__y1)/(line2__x2-line2__x1);   //Get slope of line2 (done by table
                                                          lookup in firmware)
   a2=line2__y1-slope2*line2__x1;                        //Compute intercept of line2
   xi=-(a1-a2)/(slope1-slope2);                          //Calculate x intersection
   yi=a1+slope1*xi;                                      //Calculate y intersection
}
```

[**0125**] In virtually all cases, there are multiple pairs of triangular zones that have intersecting lines. This is because of the built in redundancy of the design. Redundant zone outputs are averaged to increase accuracy when all zones are working.

[**0126**] It will be apparent from the foregoing that while particular forms of the invention have been illustrated and described, various modifications can be made without departing from the spirit and scope of the invention. Accordingly, it is not intended that the invention be limited, except as by the appended claims.

What is claimed is:

1. A touchframe system for determining the position of a touch event within a display area, said system comprising:

a plurality of light emitting elements positioned around the perimeter of the display area;