1 #include<windows.h>  
 2 #include<GL/glut.h>  
 3 #include<stdlib.h>  
 4 #include<math.h>  
 5 #include<conio.h>  
 6 #include<stdio.h>  
 7 #include <iostream>  
 8 #include <iomanip>  
 9 using namespace std;  
 10   
 11   
 12 /\*  
 13   
 14 This is an example of a 2 dimensional annimation. Two icons, a square and triangle are loaded. Then the program utilizes  
 15 the MODELVIEW Matrix to move and rotate the figures before putting them into the graphics pipeline for rendering.  
 16 The program also utilizes a small annimation driver found in TimerFunction. This driver changes the global variables   
 17 (theta, dx and dy for the square);(thatat, dxt, and dyt for the triangle) before pushing the icons through the MODELVIEW   
 18 MATRIX.   
 19 //\*\*\*\*\*\*\*\*\*\*\* Global values\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 20 /\* These values are global because the timing call back functions will only take certain parameters  
 21 hence their needs to be global variables to communicate with these functions \*/  
 22 float theta=50.0;//global angular value for rotationn  
 23 float scale1=1.0;//global scaling value for square  
 24 float dx=7.0,dy=-3.0;//global movement value for dx and dy/  
 25 float thetat=-50.0,dxt=-6.0,dyt=-3.0;// global values for the triangle  
 26 int frame=1;  
 27   
 28 void init(void);//this is a function to initialize the window clear color  
 29 void RenderScene(void);//this is a function to draw a square in an opened window  
 30 void loadicon(float[],float[],float[],float[],float [],float [],float [],float []);  
 31   
 32 /\* loads the square icon \*/  
 33 void drawicon(float[],float[],float[],float[]);/\*  
 34 draws the icon \*/  
 35 void settrans3(void);/\*sets the rotation/translation matrix the MODELVIEW MATRIX for the triangle\*/  
 36 void settrans2(void);/\* sets the rotation/translation matrix the MODELVIEW MATRIX for the square\*/  
 37   
 38 void drawtriangle(float [],float [],float [],float []);  
 39   
 40 void SetupRC(void);//sets up the clear color  
 41 void TimerFunction(int );//this call back function is call each 30 ms and changes the location,scale and rotation  
 42 // of the square.  
 43   
 44 //Main Program  
 45   
 46 int main(int argc, char\*\* argv)  
 47 {//set up window title  
 48   
 49   
 50 char header[]="Square and Triangle by Joe Student";  
 51   
 52 /\*glutInit() initializes GLUT. Takes the command line arguments which are used to initialize the native   
 53 window system. This function must be called before any other GLUT functions. \*/  
 54   
 55 glutInit(&argc, argv);  
 56 // Set up the display mode with a single buffer and RGB colors  
 57 glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB);  
 58 //Initialize window size and position  
 59 glutInitWindowSize(560,440);  
 60 glutInitWindowPosition(140,20);  
 61 //Initialize background color in window to red  
 62 SetupRC();  
 63 // Open and Label Window   
 64 glutCreateWindow(header);  
 65 glutDisplayFunc(RenderScene);  
 66 glutTimerFunc(30, TimerFunction, 1);   
 67 //Now draw the scene  
 68   
 69 glutMainLoop();  
 70   
 71 return 0;  
 72 }  
 73 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RenderScene Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 74 void RenderScene(void)  
 75 {float xdel=0.25;  
 76   
 77 float px[5],py[5],plx[2],ply[2]; /\* these variables hold the  
 78 pattern for the square icon. Note that px,py hold the square, plx,  
 79 ply hold the line \*/  
 80 float pxt1[3],pyt1[3],pxt2[3],pyt2[3];/\*these variables hold the pattern for the triangle \*/  
 81   
 82 //clear the window with the current background color  
 83 cout<<"in renderscene"<<endl;  
 84 //set the current drawing color to white  
 85 glColor3f(1.0,1.0,1.0);  
 86 glMatrixMode(GL\_PROJECTION);  
 87 glLoadIdentity();  
 88 //set the viewport to the window dimensions  
 89 glViewport(0,0,540,440);  
 90 //Establish the clipping volume in user coordinates  
 91 glOrtho(-7.0,7.0,-7.0,7.0,1.0, -1.0);  
 92 loadicon(px,py,plx,ply,pxt1,pyt1,pxt2,pyt2);  
 93 /\* draw the i  
 94 con untransformed \*/  
 95   
 96   
 97   
 98 // Clear the window with the background color  
 99 glClear(GL\_COLOR\_BUFFER\_BIT);  
100 //set the current drawing color to white  
101 glColor3f(1.0,1.0,1.0);  
102 //Set the MODELVIEW MATRIX for the Square  
103 // settrans2();  
104 /\*now draw the square. Note that in the call to drawicon below, we use the values   
105 returned from loadicon (px,py,plx,ply.pxt1,pyt1,plt2,pyt2). No need to transform   
106 them, the MODELVIEW MATRIX set in settrans2() will transform them \*/  
107 //drawicon(px,py, plx,ply);  
108 // Now Set the MODELVIEW MATRIX for the Triangle  
109 settrans3();  
110 /\*Now Draw the Triangle again note the direct use of pxt1,pyt1,pxt2,pyt2 from loadicon. Again   
111 we depend on the MODELVIEW MATRIX set in settrans3() to transform the triangle pattern \*/  
112 drawtriangle(pxt1,pyt1,pxt2,pyt2);  
113 settrans2();  
114   
115 drawicon(px,py, plx,ply);  
116   
117 glEnd();  
118   
119 glutSwapBuffers();  
120   
121   
122 return;  
123   
124 };//end of render scene  
125 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Load Icon Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
126 void loadicon(float px[],float py[],float plx[],float ply[],float ptx1[],float pty1[],  
127 float ptx2[],float pty2[])  
128 /\* this procedure loads a square icon and the triangle icon \*/  
129 {  
130 /\* Swt the coordinates of the square \*/  
131   
132 px[0]=0.0; py[0]=1.0;  
133 px[4]=0.0; py[4]=1.0;  
134 px[1]=1.0; py[1]=0.0;  
135 px[2]=0.0; py[2]=-1.0;  
136 px[3]=-1.0; py[3]=0.0;  
137   
138 /\* set the line \*/  
139 plx[0]=-1.0; ply[0]=-1.0;  
140 plx[1]=1.0; ply[1]=1.0;  
141   
142 /\* load the left triangle \*/  
143 ptx1[0]=1.0; pty1[0]=0.0;  
144 ptx1[1]=0.0; pty1[1]=2.0;  
145 ptx1[2]=0.0; pty1[2]=0.0;  
146   
147 /\* load the right triangle \*/  
148   
149 ptx2[0]=0.0; pty2[0]=2.0;  
150 ptx2[1]=-1.0; pty2[1]=0.0;  
151 ptx2[2]=0.0; pty2[2]=0.0;  
152   
153 return; } /\* end of load icon \*/  
154 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function drawicon \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
155   
156 void drawicon(float pxp[],float pyp[], float plxp[],float plyp[])  
157 {  
158 /\* this function draws the square icon at the transformed position. The model view matrix previously set before we enter this   
159 function will draw the square with the transformations\*/  
160 int i;  
161   
162 cout<<"in drawicon"<<endl;  
163 glBegin(GL\_LINE\_STRIP);  
164 //move to first point in icon  
165 glVertex2f(pxp[0],pyp[0]);  
166 //now draw the rest of the box  
167 for(i=1;i<=4;i++)  
168 glVertex2f(pxp[i],pyp[i]);  
169 glEnd();  
170 //now draw the line  
171 glBegin(GL\_LINES);  
172 glVertex2f(plxp[0],plyp[0]);  
173 glVertex2f(plxp[1],plyp[1]);  
174 glEnd();  
175 //now fill the rectangle which is made by half of the square  
176 //set the shading color to green  
177 glColor3f(0.0,1.0,0.0);  
178 glShadeModel(GL\_FLAT);  
179 //redraw the polygon  
180 glBegin(GL\_POLYGON);  
181 // note the colored rectangle must be redrawn to render it.  
182 //first point is where the line intersects the top part of the square  
183 glVertex2f((pxp[0]+pxp[1])/2.0,(pyp[0]+pyp[1])/2.0);  
184 //right corner upper  
185 glVertex2f(pxp[1],pyp[1]);  
186 //right corner lower  
187 glVertex2f(pxp[2],pyp[2]);  
188 //left intersect   
189 glVertex2f((pxp[2]+pxp[3])/2.0,(pyp[2]+pyp[3])/2.0);  
190 glEnd();  
191 return;  
192 }//end of draw icon  
193   
194 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function drawicon \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
195   
196 void drawtriangle(float ptx1[],float pty1[], float ptx2[],float pty2[])  
197 {  
198 /\* this function draws the square icon at the transformed position because the triangle   
199 MODELVIEW MATRIX is in effect \*/  
200 int i;  
201   
202 cout<<"in drawtriangle"<<endl;  
203   
204 //now fill the rectangle which is made by half of the square  
205 //set the shading color to red  
206 glColor3f(1.0,0.0,0.0);  
207 glShadeModel(GL\_FLAT);  
208 //redraw the polygon  
209 glBegin(GL\_POLYGON);  
210 // note the colored rectangle must be redrawn to render it.  
211 for (i=0;i<=2;i++) glVertex2f(ptx1[i],pty1[i]);  
212 glEnd();  
213 glColor3f(0.0,0.0,1.0);//set the shading color to blue  
214 glShadeModel(GL\_FLAT);  
215 glBegin(GL\_POLYGON);  
216 // note the colored rectangle must be redrawn to render it.  
217 for (i=0;i<=2;i++) glVertex2f(ptx2[i],pty2[i]);  
218 glEnd();  
219   
220 return;  
221 }//end of drawtriangle  
222   
223   
224   
225   
226   
227 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function settrans2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
228 void settrans2(void)  
229   
230 /\*Sets the MODELVIEW MATRIX for the square. Note that the calls are done backqards   
231 that is if we want to rotate and move the pattern, call glTranslate first and then glRotate \*/  
232 { cout<<"in settrans2"<<endl;  
233 glMatrixMode(GL\_MODELVIEW);  
234 glLoadIdentity();  
235 glTranslatef(dx,dy,0.0);  
236 glRotatef(theta,0.0,0.0,1.0);// note that the angle theta is in degrees, not radians  
237 return;  
238 }  
239 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function settrans3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
240 void settrans3(void)  
241   
242 /\*Sets the MODELVIEW MATRIX for the triangle. Note again that the calls are done backward.   
243 Further note that we must have a MODELVIEW MATRIX for each figure \*/  
244 { cout<<"in settrans3"<<endl;  
245 glMatrixMode(GL\_MODELVIEW);  
246 glLoadIdentity();  
247 glTranslatef(dxt,dyt,0.0);  
248 glRotatef(thetat,0.0,0.0,1.0);// note that the angle theta is in degrees, not radians  
249 return;  
250 }  
251   
252 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function SetupRC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
253 // Setup the rendering state  
254 void SetupRC(void)  
255 {// this function sets the clear color of an open window and clears the open window  
256 // Set clear color to blue  
257 glClearColor(0.0,0.0,1.0,1.0);  
258   
259 return;  
260 }//end of SetupRC  
261   
262 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Functioner Timer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
263 void TimerFunction(int value)  
264 //this call back function is call each 30 ms and changes the location,scale and rotation  
265 // of the square.  
266 {  
267   
268 switch(frame)  
269 {case 1: //frame 1 square starts at right (7, -3) and rolls the square to middle right (3.5,-3)  
270 // triangle starts from the left)(-7.-3)  
271 //square parameters  
272 theta+=5.0;//add 5 degrees to square rotation  
273 dx-=0.15;  
274 //triangle parameters  
275 dxt+=0.15;  
276 thetat-=5.0;//add -5 degrees to triangle rotation  
277 //we key on the square's position to change the frame  
278 if(dx<=3.5) {dx=3.5;  
279 frame=2;  
280 }  
281 break;  
282   
283 case 2:// frame 2 the square rises to y=5 triangle also rises to yt=5  
284 // first the square  
285 dy+=0.2;  
286 // now the triangle  
287 dyt+=0.2;  
288 if(dy>5.0)  
289 {dy=5.0;  
290 frame=3;  
291 }  
292   
293 break;  
294 case 3:// frame 3 square rotates at x=3.5,y=5.0 triangle also rotates  
295 //first the square  
296 theta +=5.0;  
297 // now the triangle  
298 thetat-=5.0;  
299 if(theta>=720.0)  
300 {frame=4;  
301 theta=0.0;  
302 }  
303   
304 break;  
305 case 4: // frame 4 square moves down to x=0.0, y=-3.0, triangle also comes back down  
306 //first the square  
307 dy-=0.2;  
308 //now the triangle  
309 dyt-=0.2;  
310 if(dy<=-3.0)  
311 {dy=-3.0;  
312 frame=5;  
313 }  
314   
315 break;  
316 case 5:// frame 5 square rolls off stage to left triangle rolls to right  
317 // first the square  
318 dx-=0.15;  
319 theta+=5.0;  
320 // now the triangle  
321 dxt+=0.15;  
322 thetat-=5.0;  
323 if(dx<=-6.5)dx=-6.5;  
324 if(dxt>6.5)dxt=6.5;  
325 break;  
326 }  
327   
328   
329   
330 // Redraw the scene with new coordinates  
331 glutPostRedisplay();  
332 glutTimerFunc(30,TimerFunction, 1);  
333 }  
334   
335   
336   
337   
338

A screenshot of a computer

Description automatically generatedGraphical user interface, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence