```
#include <LiquidCrystal.h>
#include <OneMsTaskTimer.h>

int delaytrigger = 3;
int const delaymax = 250; //minimum delay

void setupShield()
{
  //lcd.begin(16, 2);
  Serial.begin(9600);
}

void loopShield()
{

  if(PAS!=gameOver && PAS!=Gamestart && PAS!=GameInit)
  {
    createShield(); //try to create an obstical, designed to work some of the time
    advanceShield(); //increment all obstical positions by the appropriate amount
    //overLap(); //if two obstical over lap, delete the one thas has a lower type value
    deleteShield(); //mark off the obsticals that are no longer on the screen, can be intergrated into other fucntions but I'm not going to
  }

  if(PAS!=gameOver && PAS!=Gamestart && PAS!=GameInit)
  {
    for(int i=0; i<obstcount; i++) //counting points
      {
        if((HeroLocation.x == obsticals[i].position.x && obsticals[i].checked == 0) ||
        (HeroLocation.x > obsticals[i].position.x && obsticals[i].checked == 0))
        {
          points++;
          delaycnt++;

          Serial.print("Points updated: ");
          Serial.println(points);

          obsticals[i].checked = 1; //prevent a sigular obstical to grant multiple points
        }
      }
   }

  if(PAS!=gameOver && PAS!=Gamestart && PAS!=GameInit)
  {
    if(delaycnt >= delaytrigger && deelaay > delaymax)
    {
      if(maxShields < obstcount)
      {
        maxShields = maxShields + 1; //increment the amount of shields that can be in use when reducing delay
        Serial.print("MaxShields updated: ");
        Serial.println(maxShields);
      }

      deelaay = deelaay - (delaycnt * 20); //reducing the delay
      delaycnt = 0;

      srand(millis());
      delaytrigger = rand()%4 + 2;
      Serial.print("Delay reduced: ");
      Serial.println(deelaay);
    }
  }

  delay(deelaay);
}

void overLap() //this functionality is no longer needed, but since I spent so much time writing it, I will keep it here
```

```
66  {
67    for (int i = 0; i<obstcount; i++)
68    {
69      for(int j = obstcount-1; j>i; j--)
70      {
71        if((obsticals[i].position.x == obsticals[j].position.x) &&
          (obsticals[i].position.y == obsticals[j].position.y))
72        {
73
74          Serial.print("i: ");
75          Serial.print(i);
76          Serial.print("j: ");
77          Serial.println(j);
78
79          Serial.print("x[i]: ");
80          Serial.print(obsticals[i].position.x);
81          Serial.print(" y[i]: ");
82          Serial.println(obsticals[i].position.y);
83
84          Serial.print("x[j]: ");
85          Serial.print(obsticals[j].position.x);
86          Serial.print(" y[j]: ");
87          Serial.println(obsticals[j].position.y);
88
89          if(obsticals[i].type < obsticals[j].type)
90          {
91            Serial.print("Overlap detected 1");
92            obsticals[i].position.x = 16;
93            obsticals[i].active = 0;
94
95            lcd.setCursor(oldObsticals[i].position.x, oldObsticals[i].position.y);
96            lcd.print(" ");
97          }
98          else if(obsticals[i].type > obsticals[j].type)
99          {
100           Serial.println("Overlap detected 2");
101           obsticals[j].position.x = 16;
102           obsticals[j].active = 0;
103
104           lcd.setCursor(oldObsticals[i].position.x, oldObsticals[i].position.y);
105           lcd.print(" ");
106         }
107         else if(obsticals[i].type == obsticals[j].type)
108         {
109           Serial.println("Overlap detected 3");
110           obsticals[j].position.x = 16;
111           obsticals[j].active = 0;
112
113           lcd.setCursor(oldObsticals[i].position.x, oldObsticals[i].position.y);
114           lcd.print(" ");
115         }
116       }
117     }
118   }
119 }
120
121 void createShield()
122 {
123   int r;
124   int l;
125   srand(millis());
126   l = rand()%8;
127
128   if(shieldsInUse < maxShields)
129   {
130     for(int i=0; i<l; i++) //generates obsticals at pseudo-random intervals
131     {
132       srand(millis());
133       r = rand()%obstcount; //generates a seed each loop
```

```
134
135         if(obsticals[r].active == 0) //try to find an inactive obstical
136         {
137           obsticals[r].active = 1; //make it active
138
139           /*Serial.print("Activating element: ");
140           Serial.println(r);*/
141
142           lcd.setCursor(obsticals[r].position.x, obsticals[r].position.y); //set curser
              to location
143
144           /*Serial.print("Drawing: x: ");
145           Serial.print(obsticals[r].position.x);
146           Serial.print(" y: ");
147           Serial.print(obsticals[r].position.y);
148           Serial.print(" State: ");
149           Serial.println(obsticals[r].active);*/
150
151           if(obsticals[r].type == 0) //draw the appropriate obstical
152           {
153             lcd.write(byte(1));
154           }
155           else if(obsticals[r].type == 1)
156           {
157             lcd.write(byte(2));
158           }
159           shieldsInUse++;
160
161           Serial.print("Shield Count Updated: ");
162           Serial.println(shieldsInUse);
163           break; //exits the loop when exactly one obstical has been draw to screen
164         }
165         else
166         {
167           //Serial.println("Nothing to be drawn"); //nothing really needs to be here, but
              I'm having it print something anyway
168         }
169       }
170     }
171   }
172
173   void advanceShield()
174   {
175     for (int i = 0; i<obstcount; i++) //go through the entire array
176     {
177       //logics are put in place to ensure that an obstical of a higher type are always on
          top
178       if (obsticals[i].active == 1 && obsticals[i].type == 0) //find active obsticals
          that's of type 0
179       {
180         /*Serial.print("Current Active Elements: ");
181         Serial.println(i);*/
182
183         eraseShield(i); //advance the appropiate obstical
184         drawShield(i);
185       }
186     }
187
188     for (int i = 0; i<obstcount; i++) //go through the entire array
189     {
190       if (obsticals[i].active == 1 && obsticals[i].type == 1) //find active obsticals
          that's of type 1
191       {
192         /*Serial.print("Current Active Elements: ");
193         Serial.println(i);*/
194
195         eraseShield(i); //advance the appropiate obstical
196         drawShield(i);
197       }
```

```
198     }
199   }
200
201   void drawShield(int i)
202   {
203     if(obsticals[i].type == 0)
204     {
205       obsticals[i].position.x = obsticals[i].position.x - 1; //type 0 obsticals march
        left 1 unit at a time
206       lcd.setCursor(obsticals[i].position.x, obsticals[i].position.y);
207       lcd.write(byte(1));
208       //overLap();
209       oldObsticals[i].position = obsticals[i].position; //remember the position
210     }
211     else if(obsticals[i].type == 1) //if else works fine here, but if the type gets
      numerous, switch statements are better
212     {
213       obsticals[i].position.x = obsticals[i].position.x - 2; //type 2 obsticals march
        left 2 unit at a time
214       lcd.setCursor(obsticals[i].position.x, obsticals[i].position.y);
215       lcd.write(byte(2));
216       //overLap();
217       oldObsticals[i].position = obsticals[i].position; //remember the position
218     }
219   }
220
221   void deleteShield()
222   {
223     for(int i=0; i<obstcount; i++) //check every element of the array
224     {
225       if(obsticals[i].position.x < 0 ) //still erase the obstical if it is offscreen, but
        wait for one more clock cycle for collision detection
226       {
227         eraseShield(i);
228       }
229
230       if(obsticals[i].position.x < -1 ) //-1 to prevent the object disappearing before
        making collision
231       {
232         /*Serial.print("Deactivating element: ");
233         Serial.println(i);*/
234
235         obsticals[i].active = 0; //make the obstical inactive
236         obsticals[i].position.x = 16; //reset position
237         obsticals[i].checked = 0;
238
239         shieldsInUse--;
240         Serial.print("Shield Count Updated: ");
241         Serial.println(shieldsInUse);
242       }
243     }
244   }
245
```