```
 1: ----------------------------------------
 2: Report Cannonball Simulation for #112
 3: ----------------------------------------
 4:
 5: Today we make a detour into the world of ballistics, simulation &
    training.
 6: One of my friends on Delphi for Fun gave me the idea to port the
    executable to a script for windows 11 as a preparation for the 64bitbox.
 7:
 8: So ballistics is the study of the motion of projectiles, such as bullets,
     shells, and rockets, in our script we deal only with balls. It is a
    branch of mechanics that deals with the behavior of objects in motion.
    Ballistics can be divided into three main categories: internal
    ballistics, external ballistics, and terminal ballistics.
 9: So I translated the Delphi program with a few improvements into that
    script:
10:
11: http://www.softwareschule.ch/examples/cannonball.txt
12: image1: 1_1234_cannonball4.png
13: {$
    https://softwareschulecode.files.wordpress.com/2023/09/cannonball4.png }
14:
15: Of course we can simulate a cannonball using a physics simulation
    software or in our case integrate that model with Pascal. This
    simulation allows you to blast a ball out of a cannon and challenge
    yourself to hit a moveable target.
16: You can set parameters such as angle (elevation), initial speed (powder
    charge), and mass (gravity), and explore the vector representations.
17:
18: Also an explainable statistic is part of the script, as summary or
    detailed (our standard case which hit the target):
19:
20: Summary of study case
21: ------------------------------------------------------------------
22: Barrel Len 87, Angle   45.0, Initial V   24.0, gravity    1.0
23: Time in barrel    3.8 seconds
24: X distance at end of barrel   61.5
25: Y distance at end of barrel   61.5
26: Time to top of freeflight arc   15.1,   18.9 total
27: X distance to top of freeflight arc  226.5,  288.1 total
28: Height above barrel to top of freeflight arc  113.3,  174.8 total
29: Time to reach ground from max height   18.7,   37.6 total
30: X distance from top of freeflight arc to end  281.4,  569.5 total
31: ------------------------------------------------------------------
32:
33: image2: 2_1234_cannonball4_studycase.png
34: https://softwareschulecode.files.wordpress.
    com/2023/09/2_1234_cannonball4_studycase.png
35:
36: The interesting thing is that this simulation shows how the motion of a
    projectile like a cannonball is fundamentally the same as the orbit of a
    celestial body like the moon!
37: The rotate and translate routines developed are used here to elevate the
    cannon. The ball movement loop is similar to a Bouncing Ball program
    with the addition of a horizontal component.
```

Initial velocities in the X and Y direction are proportional to the cosine and sine of the elevation angle respectively.
The barrel is a bit tricky; We do assume that the cannonball inside the barrel is "rolling up a ramp" with the component of gravity acting parallel to the barrel being the force acting to reduce the velocity of the cannonball in both x and y directions, so we keep an eye on the distance function:

```
  function distance(p1,p2:TPoint):float;
  begin
    result:= sqrt(sqr(p1.x-p2.x)+sqr(p1.y-p2.y));
  end;
```

Two functions, Rotate and Translate, do the rotation of points. Rotation about an origin point of (0,0) is rather straightforward as we can see from the code below:

```
  procedure rotate(var p:Tpoint; a:float);
  {rotate a point to angle a from horizontal}
  var t:TPoint;
  begin
    t:=P;
    p.x:=trunc(t.x*cos(a)-t.y*sin(a));
    p.y:=trunc(t.x*sin(a)+t.y*cos(a));
  end;

  procedure translate(var p:TPoint; t:TPoint);
  {translate a point by t.x and t.y}
  Begin
    p.x:=p.x+t.x;
    p.y:=p.y+t.y;
  end;
```

Once we have the point rotated to the desired angle relative to then origin, Translate() can move the point by adding the new x and y origin coordinates to the x and y values of the point of type TPoint.
The other logic is to determine whether the cannonball has hit the target, which is moveable by a trackbar. “Collision detection” is a common (and also complicated) problem in most animated graphics apps. The implementation is checking if the distance from the center of the cannonball is less than its radius from the left or top edges of the target after each move or hit. The problem is that, for low angles, a horizontal movement may take the ball from one side of the target to the other side in one loop increment, so we never know that we went right through it!
A funny thing is the storage of cannonballs; Spherical objects, such as cannonballs, can be stacked to form a pyramid with one cannonball at the top, sitting on top of a square composed of four cannonballs, sitting on top of a square composed of nine cannonballs, and so forth.

image3: 3_1234_cannonball4compiler1.png
https://softwareschulecode.files.wordpress.com/2023/09/3_1234_cannonball4compiler1.png

```
 71: In PyGame for example, collision detection is done using Rect objects.
     The Rect object offers various methods for detecting collisions between
     objects. Even the collision between a rectangular and circular object
     such as a paddle and a ball can be detected by a collision between two
     rectangular objects, the paddle and the bounding rectangle of the ball.
     Now we can summarize the theoretic results in a procedure of our
     statistic:
 72:
 73: {************ TheoreticalCalc **********}
 74: procedure TheroreticalCalc;
 75: var
 76:   root,T1, Vf:float;
 77:   Vxf, Vyf:float;
 78:   X1,Y1:float;
 79:   TTop, Xtop,Ytop:float;
 80:   Tlast, VyLast, Xlast:float;
 81:   floor:float;
 82: begin
 83:   with {stats.}amemo1.lines do begin
 84:     clear;
 85:     add(format('Barrel Len %d, Angle %6.1f, Initial V %6.1f, gravity
     %6.1f',
 86:                                              [barrellength,180*theta/pi,v1,
     g]));
 87:     if g=0 then g:=0.001;
 88:     root:= v1*v1 - 2*g*sin(theta)*Barrellength;
 89:     if root>=0 then begin
 90:       T1:=(v1 - sqrt(root))/(g*sin(theta+0.001));
 91:       Vf:= v1 - g*sin(theta)*T1;
 92:       Vxf:=Vf*cos(theta);
 93:       Vyf:=Vf*sin(theta);
 94:       X1:=Barrellength*cos(theta);
 95:       Y1:=Barrellength*sin(Theta);
 96:       floor:=(origin.y+ballradius)-groundlevel;
 97:       {out of barrel, Vx remains constant, Vy := Vyf- g*DeltaT}
 98:       {Vy=0 then Vyf-g*Ttop=0 or Ttop=Vyf/g}
 99:       Ttop:=Vyf/g;
100:       {x distance at top} Xtop:=Vxf*Ttop;
101:       {height at top = average y velocity+ time}   Ytop:=(Vyf +
     0)/2*TTop;
102:       {Time to fall from ytop to groundlevel, descending part of
     projectiles path}
103:       {speed when ball hits ground}
104:       TLast:=sqrt(2*(Y1+YTop-floor)/g );
105:       Xlast:=Vxf*TLast;
106:       add(format('Time in barrel %6.1f seconds',[T1]));
107:       add(format('X distance at end of barrel %6.1f',[X1]));
108:       add(format('Y distance at end of barrel %6.1f',[Y1]));
109:       add(format('Time to top of freeflight arc %6.1f, %6.1f total',
     [Ttop,T1+Ttop]));
110:       add(format('X distance top of freeflight arc %6.1f, %6.1f total',
     [Xtop,X1+Xtop]));
111:       add(format('Height above barrel to top of freeflight arc %6.1f,
     %6.1f total',
```

```
112:                                                                [Ytop,
    Y1+Ytop]));
113:       add(format('Time to reach ground from max height %6.1f, %6.1f
    total',
114:                                                                [TLast,
    T1+Ttop+TLast]));
115:       add(format('X distance from top of freeflight arc to end %6.1f,
    %6.1f total',
116:                                                                [XLast,
    X1+Xtop+XLast]));
117:     end else add('Velocity too low, cannonball does not exit barrel');
118:   end;
119: end;
120:
121: By the way I asked ChatGPT how can I program cannonball in Pascal and
    the answer:
122: To program a cannonball in Pascal, you can use the following steps:
123:
124: 1. Define the initial position and velocity of the cannonball.
125: 2. Calculate the acceleration of the cannonball due to gravity.
126: 3. Update the velocity and position of the cannonball using the
    calculated acceleration.
127: 4. Repeat step 3 until the cannonball collides with an object or reaches
    a certain height.
128:
129: In this example code snippet, CircleRectCollision() is a custom function
    that detects collision between a circle and a rectangle. You can modify
    this function to suit your needs; the main part of the script has only 4
    procedures:
130:
131:   processmessagesOFF;
132:   loadStatForm();
133:   loadmainForm();
134:   UpdateImage();
135:
136: Links and References:
137:
138: http://www.softwareschule.ch/examples/cannonball.txt
139: http://delphiforfun.org/Programs/bouncing_ball.htm
140: https://en.wikipedia.org/wiki/Ballistics
141:
142: ------------------------------------------------------------------------
    -----
143: ------------------------------------------------------------------------
    -----
144:
145:
146:
147:
148:
149:
150:
151:
152:
```

maXbox4exe

Digitally signed by maXbox4exe
Date: 2023.09.04 16:33:27 +02'00'

```
153:
154:
155:
156: PROGRAM Demo_App_mX64;
157: //CONST
158: //<Constant declarations>
159:  // TEXTOUT = 'hi world of code rage';
160:
161: {TYPE
162: <Type declarations>}
163:
164: //Var
165: //<Variable declarations>
166: //  i: integer;
167:
168: //<FUNCTION>
169: //<PROCEDURE>
170:
171: BEGIN  //Main
172: //<Executable statements>
173:  // for it:= 1 to 3 do
174:   //  Writeln('TEXTOUT+CRLF');
175:   writeln(inttostr(9));
176:   //maXcalcF('2^64 /(60*60*24*365)')
177: //<Definitions>
178:
179:  with TPythonEngine.Create(Nil) do begin
180:     pythonhome:= 'C:\Users\User\AppData\Local\Programs\Python\Python310-
    32\';
181:     try
182:       //loadDLL;
183:       opendll('C:\Users\User\AppData\Local\Programs\Python\Python310-
    32\python310.dll');
184:       execStr('import http.client as httplib');
185:       execStr('connection=httplib.HTTPSConnection("api.parse.com",443)');
186:       execStr('connection.connect()');
187:       execStr('connection.request("GET","api.parse.com/echo")');
188:       println(evalStr('connection.getresponse().read()'));         //}
189:     except
190:       raiseError;
191:     finally
192:       unloaddll;
193:       free;
194:     end;
195:   end;
196: END.
197: ----app_template_loaded_code----
198: ----File newtemplate.txt not exists - now saved!----
```