

QUIZ #6

FIREWORKS WITH MATHEMATICA

I am sure you marveled about the displays of fireworks on the 4th of July. In this quiz you will learn how to get the MATHEMATICA to do fireworks on the 18th of November.

As usual we will have to decompose the task into smaller subtasks. To begin, we need to review a bit the mechanics of a single firework shot. Basically, a rocket propels a package containing an exploding device up to a certain point in space. We usually barely see that particular phase, and it is only a minor concern for us here. We shall mostly be concerned with what happens after the explosion. The device, which is basically a bomb, ejects a number of incandescent metallic particles in random directions emanating radially from the center of the explosion. Each of these particles follows a trajectory dictated by the initial velocity vector, and by the gravity force.

Your task here is to write a package of procedures that results in an animated display of a single fireworks shot. Basically your final image should be similar to the one exhibited here on the right. Your animation will start from the big bang and progressively obtain the final image. Of course you may do this any way you want and produce your own personal firework shot. However, to simplify your task in the lines that follow I have put together an outline of the step that yielded me my firework shot, together with the final parameters I used to obtain the display.

Let us recall that the law of motion (position as function of time) of a particle $P(t)$ (traveling in the x, y plane) ejected from an initial point

$$P_o = (x_o, y_o)$$

with initial velocity vector

$$V = (v_x, v_y)$$

is $P(t) = (x(t), y(t))$ with

$$\begin{aligned} x(t) &= x_o + v_x t \\ y(t) &= y_o - g t^2/2 + v_y t \end{aligned} \quad (1)$$

where g gives the downward acceleration due to gravity. On earth $g = 9.81m/sec^2$, but on the computer we may have to use a different value to achieve our desired visual effect.

Thus our first step is to write a procedure with heading “**law**[**Po**-, **V**-, **t**-, **g**-]” which returns $P(t)$. To test this procedure and for the final display, You may use the following MATHEMATICA procedure

```
disp[obj-] := Show[Graphics[obj], AspectRatio- > Automatic,  
Frame- > True, PlotRange- > All, ImageSize- > {400, 800}]
```

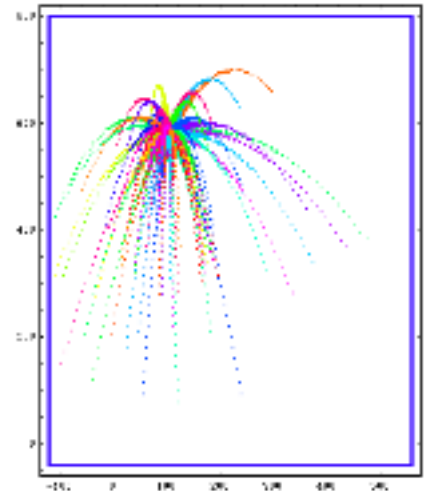
Note the graphic option “**ImageSize**” allows you to control the size of the resulting display.

To get a view of the trajectory of a single particle. Just choose an initial velocity vector of length 1 pointing upwards with an angle of 80° with the x -axis, and get a table of values of $P(t)$ with $P_o = (0, 0)$, $g = 3.1$ for

$$t = .1 \times i \quad \text{for } i = 1, 2, \dots, M$$

with M large enough to get back to the ground. That is for $y(M) = 0$. For later purposes I should point out that the trajectory will reach its highest point at the time t_o when $y'(t_o) = 0$. Thus from (1) we derive that

$$t_o = v_y/g$$



Our next step is to write a procedure with heading “**ravec**[**v**_]” which constructs **alpha** = 2π **Random**[] and returns the vector

$$\{\mathbf{v} \text{Cos}[\text{alpha}], \mathbf{v} \text{Sin}[\text{alpha}]\}$$

(recall that the MATHEMATICA command “**Random**[]” returns a random number in the interval $[0, 1]$) this procedure will be used to produce random initial vectors for each of the particles emitted by the explosion. It will be convenient to declare the abbreviation

$$\mathbf{ra}[] := \mathbf{Random}[]$$

and similar abbreviations for long MATHEMATICA commands to avoid long phrases in your procedures.

Next we need the procedure

$$\mathbf{colpt}[\mathbf{P}_-, \mathbf{c}_-, \mathbf{s}_-] := \{\mathbf{PointSize}[\mathbf{s}], \mathbf{Hue}[\mathbf{c}], \mathbf{Point}[\mathbf{P}]\}$$

That constructs the graphic object consisting of a point of size s and Hue value c at the point P . The reason for including the size as an input variable is to produce the visual effect (commonly observed in fireworks displays where the incandescent particles diminish in size as they go down. We also want the color to vary at random from particle to particle, so an invocation of **colpt**[**P**, **ra**[], .2] will give a point at P of size .2 and color given by whatever r turned out to be.

Our crucial procedure has heading “**bang**[**Po**_, **SV**_, **scolors**_, **t**_, **so**_]”. Given an input sequence **SV** of random velocity vectors and a sequence **scolors** of random colors, **bang** should return the position of “**Length**[**SV**]” particles at time t whose common initial position at time $t = 0$ (the moment of the explosion) was at the point P_0 . Note that the color of particle i should be given by **scolors**[[**i**]] and the initial velocity vector of the same particle should be **SV**[[**i**]]. Note further that the input variable “**so**” is included to allow the user to control the initial size of all the particles at the moment of the explosion. However within the procedure you must invoke the procedure “**colpt**” with **s** a fraction of the initial size which becomes smaller and smaller as t increases. Experimentally I found that to achieve the desired effect I had to take the size at time t to be **so**/ $t^{2/3}$.

The procedure **bang** only returns the collection of colored points yielded by **colpt** once its input data is given. To construct the input data we need a procedure with heading “**shot**[**Po**_, **v**_, **s**_, **k**_, **M**_]”. Here “**Po**” is a user chosen point of explosion, “**v**” is an experimentally determined size of the initial velocity vectors, “**s**” gives the time unit we select, “**k**” gives the number of particles and finally **M** gives the total number of time units that the animation will consist of.

This given, **shot** should start by constructing **k** random vectors of length **v** by an invocation of **ravec**. Next **shot** constructs a sequence of **k** random Hue values. These will be part of the input needed for the invocations of **bang**. This done we are ready for the big show. In fact “**shot**” should return the whole history of the fireworks shot. This should be a table of tables “**trags**” with **trags**[[**r**]][[**i**]] for $i = 1, 2, \dots, r$ giving the positions of each of the **k** particles together with their colors and sizes at each of the first **r** time units.

To get the animation of your firework shot. You need only three more commands. First type

<< **Graphics**‘**Animation**‘

Then on your first try you may use as starting parameters **g** = 3, **v** = 25, **s0** = .06 the time unit to be .4, the number of particles **k** = 8 (the display above was obtained with **k** = 60) and **M** = 140. With these choices you would type

```
boo = shot[{200, 600}, 25, .4, 8, 140]
```

Then finally type

```
disp[Table[{box, boo[[r]]}, {r, 1, M}]]
```

with `box` giving the blue rectangle in the display in the previous page. The purpose of this rectangle is to make sure that the animation is not affected by the unpleasant wobbling that MATHEMATICA produces when the display varies in size. You can only determine the size of the box experimentally, because this device will not work if any trajectory at any time falls out of the box. Try at first your program with

Please don't refrain to get help since this is a rather complex task and you may need guidance to successfully carry it out. If you wish to include in your shot also the trajectory of the rocket that produces the shot, ask me and I will tell you how I did it. But I must warn you that this adds further complications. It might be more fun to arrange so that several shots are displayed within a single animation. Enjoy.