2d parabolic projectile

Asked 13 years, 9 months ago Modified 13 years, 9 months ago

Viewed 7k times

3

I'm looking to create a basic Javascript implementation of a projectile that follows a parabolic arc (or something close to one) to arrive at a specific point. I'm not particularly well versed when it comes to complex mathematics and have spent days reading material on the problem. Unfortunately, seeing mathematical solutions is fairly useless to me. I'm ideally looking for pseudo code (or even existing example code) to try to get my head around it. Everything I find seems to only offer partial solutions to the problem.

In practical terms, I'm looking to simulate the flight of an arrow from one location (the location of the bow) to another. I have already simulated the effects of gravity on my projectile by updating its velocity at each logic interval. What I'm now looking to figure out is exactly how I figure out the correct trajectory/angle to fire my arrow at in order to reach my target in the shortest time possible.

Any help would be greatly appreciated.

javascript physics game-physics

Share

edited Mar 10, 2011 at 16:38

Improve this question

Follow

asked Mar 10, 2011 at 15:58

ndg

2,645 ● 2 ● 34 ● 59

2 Answers

Sorted by: Highest score (default)

4

Pointy's answer is a good summary of how to simulate the movement of an object given an initial trajectory (where a trajectory is considered to be a *direction*, and a *speed*, or in combination a *vector*).

However you've said in the question (if I've read you correctly) that you want to *determine* the initial trajectory knowing only the point of origin `O` and the *intended* point of target `P`.

The bad news is that in practise for any particular `P` there's an infinite number of parabolic trajectories that will get you there from `O`. The angle and speed are interdependent.

If we translate everything so that O is at the origin (i.e. [0, 0]) then:

```
T_x = P_x - O_x             // the X distance to travel
T_y = P_y - O_y             // the Y distance to travel

s_x = speed * cos(angle)    // the X speed
s_y = speed * sin(angle)    // the Y speed
```

Then the position `(x, y)` at any point in time `(t)` is:

```
x = s_x * t
y = s_y * t - 0.5 * g * (t ^ 2)
```

so at impact you've got

```
T_x = s_x * t
T_y = -0.5 * g * (t ^ 2) + s_y * t
```

but you have three unknowns (`t`, `s_x` and `s_y`) and two simultaneous equations. If you fix one of those, that should be sufficient to solve the equations.

FWIW, fixing `s_x` or `s_y` is equivalent to fixing either `speed` or `angle`, that bit is just simple trigonometry.

Some combinations are of course impossible - if the speed is too low or the angle too high the projectile will hit the ground before reaching the target.

NB: this assumes that position is evaluated *continuously*. It doesn't quite match what happens when time passes in *discrete* increments, per Pointy's answer and your own description of how you're simulating motion. If you recalculate the position sufficiently frequently (i.e. 10s of times per second) it should be sufficiently accurate, though.

edited Mar 10, 2011 at 16:42

answered Mar 10, 2011 at 16:16

Alnitak

340k ● 71 ● 418 ● 502

Other people are always so much better at reading questions than I am. :(– Pointy Mar 10, 2011 at 16:18

That's understood. I suppose my question really is: how can I determine the best angle/trajectory to fire my projectile at in order to hit my target in the shortest time? – ndg Mar 10, 2011 at 16:36

2 @ndg: Without any other bounds, the solution is to aim directly at the target and fire with infinite speed. With an upper bound on speed, the fastest solution will always be firing at *that* speed (which results in not only the fastest arrow but the shortest trajectory), so now you can solve for the angle – Andrzej Doyle Mar 10, 2011 at 16:39

4

I'm not a physicist so all I can do is tell you an approach based on really simple process.

1. Your "arrow" has an "x" and "y" coordinate, and "vx" and "vy" velocities. The initial position of the arrow is the initial "x" and "y". The initial "vx" is the horizontal speed of the arrow, and the initial "vy" is the vertical speed (well velocity really but those are just words). The values of those two, conceptually, depend on the angle your bowman will use when shooting the arrow off.

2. You're going to be simulating the progression of time with discrete computations at discrete time intervals. You don't have to worry about the equations for "smooth" trajectory arcs. Thus, you'll run a timer and compute updated positions every 100 milliseconds (or whatever interval you want).
3. At each time interval, you're going to add "vx" to "x" and "vy" to "y". (Thus, note that the initial choice of "vx" and "vy" is bound up with your choice of time interval.) You'll *also* update "vx" and "vy" to reflect the effect of gravity and (if you feel like it) wind. If "vx" doesn't change, you're basically simulating shooting an arrow on the moon :-) But "vy" will change because of gravity. That change should be a constant amount subtracted on each time interval. Call that "delta vy", and you'll have to tinker with things to get the values right based on the effect you want. (Math-wise, "vy" is like the "y" component of the first derivative, and the "delta vy" value is the second derivative.)
4. Because you're adding a small amount to "vy" every time, the incremental change will add up, correctly simulating "gravity's rainbow" as your arrow moves across the screen.

Now a nuance you'll need to work out is the sign of "vy". The initial sign of "vy" should be the opposite of "delta vy". Which should be positive and which should be negative depends on how the coordinate grid relates to the screen.

edit — See @Alnitak's answer for something actually germane to your question.

Share Improve this answer Follow

edited Mar 10, 2011 at 16:19

answered Mar 10, 2011 at 16:07

Pointy

413k ● 61 ● 594 ● 625

Pointy: thanks for the explanation. I should have mentioned in my original question that I've simulated the arrow movement with a velocity vector that is updated each logic tick to represent gravitational forces. – ndg Mar 10, 2011 at 16:35

OK. Like I said to @Alnitak, I'm pretty bad about reading questions thoroughly. – Pointy Mar 10, 2011 at 16:36

No need for apologies. I appreciate you taking the time to help. – ndg Mar 10, 2011 at 16:38