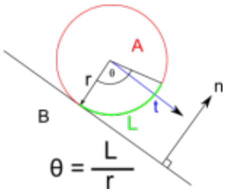
The Wayback Machine - https://web.archive.org/web/20120614004712/http://www.wildbunny.co.uk:80/blog/2012/04/02/the-physics-of-rolling/

**Paul's blog@Wildbunny**

*Knowledge centre for game development*

# The physics of rolling

Posted on April 2, 2012 by Paul Firth



Tweet Like 4 Submit

Hello and welcome back to my blog!

This time I'm going to talk about the physics behind simulating a rolling circle, Super Monkey Ball style but in 2d.

## The key

The key to making this work is understanding how we can take a length (like a movement vector) and turn it into an angle (amount of roll from linear movement).

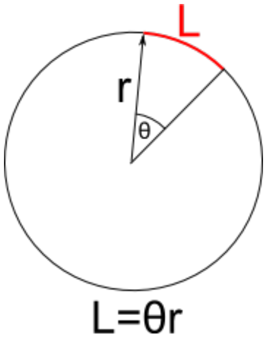


Figure 1

Luckily there is a nice easy way and *Figure 1* shows how it works; as long as you have the arc-length and the radius of the circle you can compute the angle the circle would have rolled in order to cover the distance of the length of the arc-length.

This formula should be familiar – remember how to compute the circumference of a circle?

$$\mathbf{c = 2\pi r}$$

2π is the arc length you would use in *Figure 1* to get the circumference of a circle.

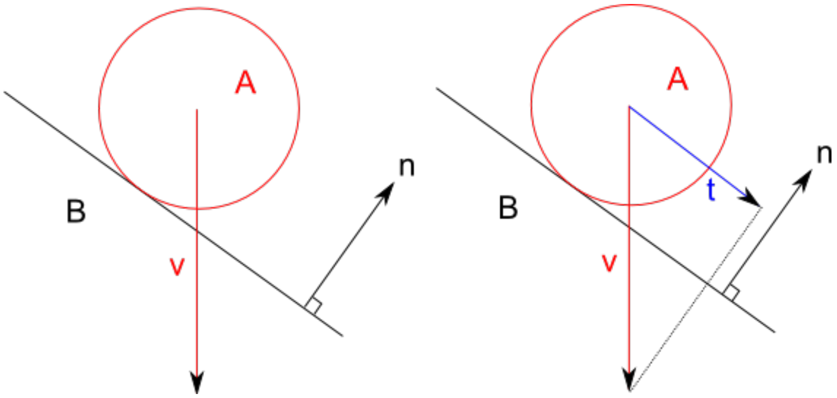


Figure 2

*Figure 2* shows the scenario; we have a circle *A* travelling downwards towards a slope *B* with linear velocity *v*. The circle is not yet rotating. On the right we can see the tangential velocity *t* – that is, the linear velocity of *A* we want to end up with after the collision. It is easy to compute *t* because we have the surface normal *n*:

$$\mathbf{t} = \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$$

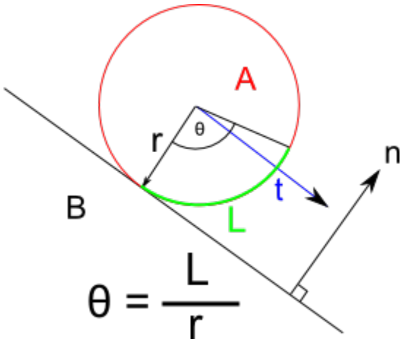


Figure 3

*Figure 3* shows what needs to be done next. By re-arranging the arc-length formula we can find θ but how do we compute *L*?

Actually, what we want is not a straight length value because the circle can roll both clockwise and anti-clockwise. We need to find a signed version of *L*. Luckily we can do this quite easily using the wedge product of *v* and *n*:

$$\mathbf{L}_{\mathbf{signed}} = \mathbf{v} \wedge \mathbf{n}$$

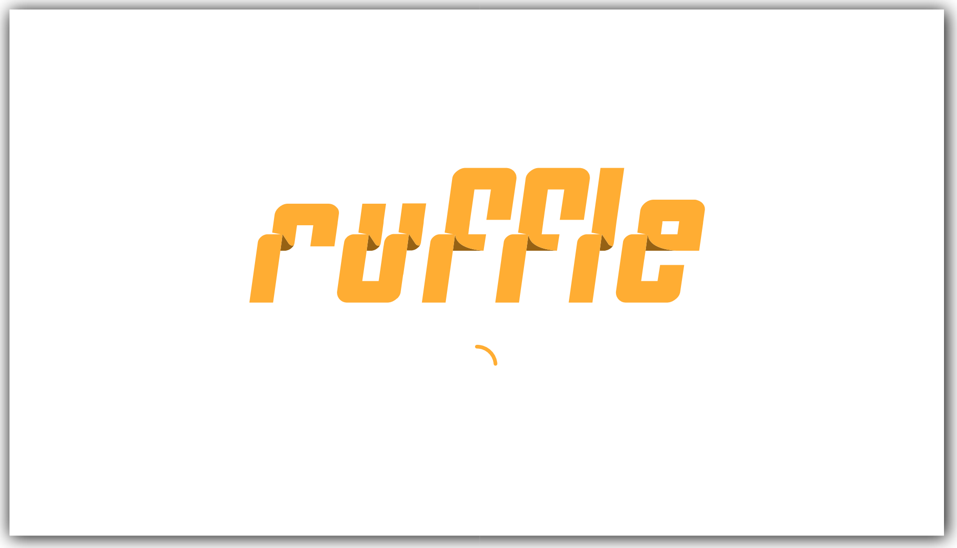
This is the signed magnitude of the tangential velocity of *v*.

Now we can compute *θ* which is our rotational change in velocity, i.e. the resultant rolling motion corresponding to the linear motion of *A* after collision with *B*.

$$\boldsymbol{\theta} = \mathbf{L}_{\mathbf{signed}} / \mathbf{r}$$

## Putting it all together

Now we can compute both the resultant linear and angular velocities for *A* after collision with *B*, we can put it all together in simulation form!



You can drag the circle with the mouse.

## Limitations

The simulation has 100% friction which is a little unrealistic and extending this to handle multiple circles colliding with each other is tricky and beyond the scope of this article.

## The source

Want to get a more detailed idea by downloading the source? No problem!

[Click here to download for free!]

That’s it for this article!

Until next time, have fun!

Cheers, Paul.

Tweet Like 4 Submit

**About Paul Firth**
A games industry veteran of ten years, seven of which spent at Sony Computer Entertainment Europe, he has had key technical roles on triple-A titles like the Bafta Award Winning Little Big Planet (PSP), 24: The Game (PS2), special effects work on Heavenly Sword (PS3), some in-show graphics on the BBC’s version of Robot Wars, the TV show, as well as a few more obscure projects. Now joint CEO of Wildbunny, he is able to give himself hiccups simply by coughing.
View all posts by Paul Firth →

This entry was posted in AS3, Circles, Geometry, Technical and tagged collision detection, contact, physics, physics engine, programming, simulation. Bookmark the permalink.

## 8 Responses to *The physics of rolling*

**Jameson** *says:*
April 2, 2012 at 7:49 pm

Nice article! I can't work out why the cross product of the vectors only gives an approximate arc length, but it is Monday after all.

Also, shouldn't L have the same length as t? I know this isn't too helpful as finding the length of t is a slow operation, but it seems worth mentioning.

Reply

**Paul Firth** *says:*
April 3, 2012 at 9:31 am

Hi Jameson,

Its only approximate because its converting an exact distance moved in a straight line to an angle using a formula which expects to receive a curved length. Its easier to see why if you imagine a much smaller angle than shown in the diagram – one where the straight line actually touches the arc end points.

You're right about L having the same length as t, but if we just take the length, its not a signed value and its a slow operation 🙂

Cheers, Paul.

Reply

**Paul Firth** *says:*
April 4, 2012 at 9:25 am

Hi again,

Thinking about it, I think I might be wrong about the approximation thing – a few people have mentioned that it should be exact 🙂

Cheers, Paul.

Reply

**Jameson** *says:*
June 13, 2012 at 1:03 pm

When I first read this I took the time to think about the maths fairly carefully and I was certain then it was exact. I'd properly re-examine it but I'm at work! It's a really neat solution, and I wish I had any plans for games that could make use of half the stuff you write about – but platform and physics-based games just aren't what I usually work on 🙁

Reply

**Bob** *says:*
April 3, 2012 at 7:15 am

I'm definitely interested in seeing how multiple balls colliding with each other works. That's a great Topic for another blog post.

Reply

**Fuhans** *says:*
April 26, 2012 at 12:20 pm

Sir Paul Firth, i want to ask about collision detection..
i have a blocks (floors and walls) and character (ball), but the problem is when i press space (which is i code to jump the character), the character is falling down through the floors, not stop in top of the floors.. and when i move the character, it always move through whenever the wall in front of it.. how to detect the collision between the character and floors and walls sir? any help will be appreciate.. thank you

Reply

**Paul Firth** *says:*
April 26, 2012 at 1:22 pm

Hi there,

Please see my article about collision detection in a 2d platform game:

http://www.wildbunny.co.uk/blog/2011/12/14/how-to-make-a-2d-platform-game-part-2-collision-detection/

It comes with free source code option as well 🙂

Cheers, Paul.

Reply

**Fuhans** *says:*
April 26, 2012 at 5:18 pm

Thank you sir 🙂

Reply