

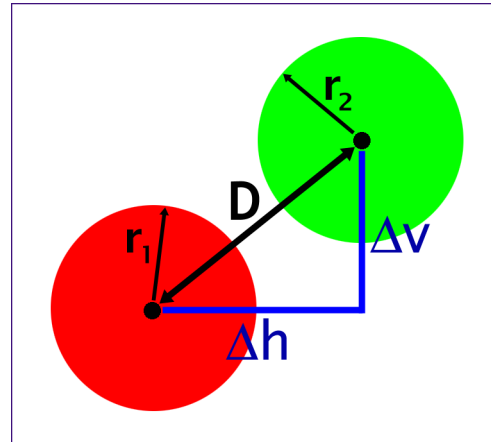
Circles Colliding – Part I

Prepared by: Teton Multimedia (www.TetonMultimedia.com)

Collisions of round objects are the simplest type to deal with. However, the topic requires a basic familiarity with trigonometry and vectors. So it's suggested that you read our lesson on these topics before proceeding.

If you have two round objects, then they are NOT contacting each other IF the distance between their center points is GREATER THAN the sum of their radii. If the distance between their center points is LESS THAN OR EQUAL TO the sum of their radii, then they must be in contact.

Wow! This is easy! All you have to do is add their radii and compare with the distance between their centerpoints.



In the figure shown here, the distance between centerpoints is

$$1.1 \quad D = (\Delta h^2 + \Delta v^2)^{1/2}$$

Note that if you are unfamiliar, the superscript "1/2" is the same as "square root." The two circles will be in contact when

$$1.2 \quad D \leq r_1 + r_2$$

Note that for programming efficiency, we can avoid the square root calculation in equation 1.1 by recasting it as

$$1.3 \quad D^2 = \Delta h^2 + \Delta v^2$$

Similarly, we square both sides of our condition equation 1.2 to arrive at

$$1.4 \quad D^2 \leq (r_1 + r_2)^2$$

In this way, we compare distances squared, rather than distances. Consequently, we do fewer calculation, and this will help our animation proceed smoothly on slow computers.

Substituting the right side of equation 1.3 into the left side of equation 1.4 gives us the final form to determine whether contact has occurred is

1.5

$$\Delta h^2 + \Delta v^2 \leq (r_1 + r_2)^2$$

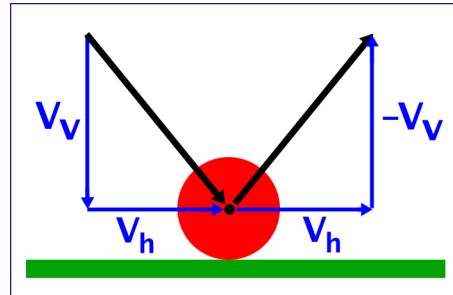
In Director, Δh and Δv are calculated as $(Locv_2 - Locv_1)$ and $(Loch_2 - Loch_1)$, respectively. And r_1 and r_2 are typically $\frac{1}{2}$ of the sprite width. (We assume that the circle is contained exactly within a square sprite.) We simply plug this information into equation 1.5 to determine whether a collision has occurred.

(Note — an alternate simple method is available in Director: Simply test for intersection of the two sprites containing the circles. However, this doesn't deal with the corner areas of the squares very well. If the sprites bump corners, then a collision is implied. And it will probably be obvious to the eye that the circles did NOT collide.)

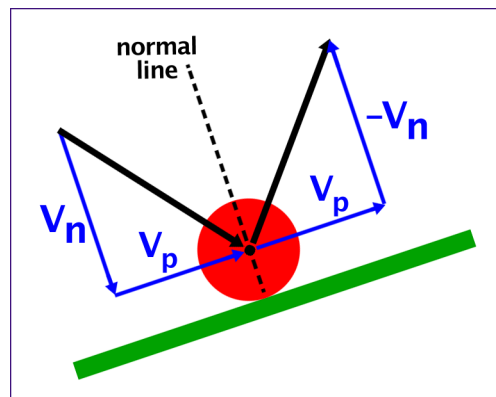
If a collision HAS occurred, then what next?

In the simple case, we assume that one circle is locked in place and the other circle bounces off it in a completely elastic collision. This means that the moving circle changes direction, but its speed does not change.

Simplifying, if a circle bounces off a horizontal surface, we expect that the vertical portion of its velocity will change sign while the horizontal velocity does not change. This is easily illustrated as shown here.



If the plane is *not* horizontal, the bounce is processed in a similar manner. The velocity component that is perpendicular to the surface (v_n) changes sign, while the velocity component that is parallel to the surface (v_p) does not change. You can see this effect here. It is equivalent to say that the velocity is reflected around the line perpendicular to the surface. It is customary to refer to this line as the “normal.” As you can see, the normal is the same as the line connecting the collision point and the circle's center.



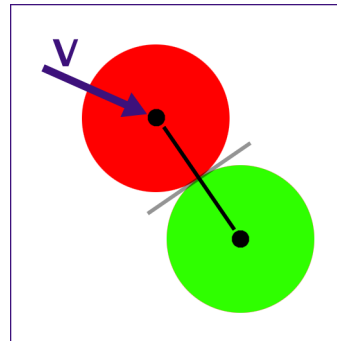
Unfortunately, we typically *do not* know v_n and v_p . Instead, we know only the horizontal and vertical velocity components. Consequently, the math is more complicated than simply changing one sign.

In each of these collisions against a flat surface, what portion of the surface is contacted by the circle? Answer: a single point. This can be seen if you erase all but the contact

point. Yes, it may be true that the contact point *appears* to be a short line segment; however, this appearance is a consequence of our limited ability to draw with sufficient precision. The truth is that there is only a contact *point*.

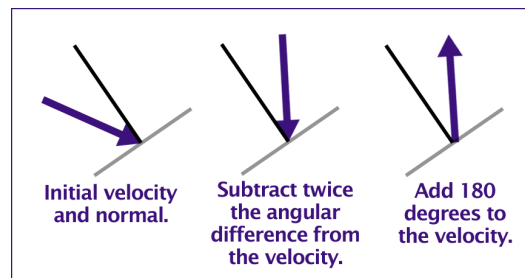
Similarly, when two circles contact, what portion of one circle touches the other? Answer: a single point. And this point is on the line between the centers of the circles. This implies that the line between the centers of the circles is *the same as the normal* mentioned above.

So, to calculate the new velocity after the collision, all we need are (1) the two circles' locations and the normal that we can calculate using these locations, and (2) the moving circle's velocity. All other information becomes extraneous (but only *after* a collision has been determined).



We know the moving circle's location from its Loch and Locv. This gives us the normal line shown above. We assume that we know the circle's velocity because we have been calculating it all along (*our* animation and velocity move the circle). This implies that we have all the information we need to perform the calculation.

To reflect the velocity around the normal line (the connecting line), all we need to do is find the angle between the velocity and the normal, subtract this angle twice from the velocity, and add 180 degrees.



(Wow! This is easy!)

The angle of the velocity (v) is given by:

$$2.1 \quad A_v = \text{atan}(v_v / v_h)$$

The angle of the normal is given by:

$$2.2 \quad A_n = \text{atan}((\text{Locv}_2 - \text{Locv}_1) / (\text{Loch}_2 - \text{Loch}_1))$$

where the subscript 2 and 1 refer to the green circle and red circle, respectively. The difference between the two angles is:

$$2.3 \quad A_d = A_v - A_n$$

The post-collision angle of the velocity is then:

2.4

$$A_{pc} = A_v - 2 * A_d + \pi$$

Where π radians is the same as 180 degrees. The *magnitude* of the velocity (the speed) remains unchanged, so the horizontal and vertical components can be found, respectively, by multiplying the *old* magnitude by the cosine and sine of the A_{pc} .

And that's it! You're done! You just bounced a moving circle off a stationary (locked-in-place) circle.

Note:

1. Recall that if you know the horizontal and vertical components of the old velocity, then the magnitude of the velocity is $v_m = (v_h^2 + v_v^2)^{1/2}$.
2. When programming this method, if you see the moving circle bounce off at a strange angle, the culprit is likely to be a sign error (you've subtracted, rather than adding, or some such thing).