Billiard Paths on Arbitrary Tables

Note: The first time you evaluate a cell in this notebook you will get a popup asking you if you want to evaluate all initialization cells in the notebook - say "Yes" to this, as it will evaluate the code in the closed cell below which is necessary for the two custom commands in this notebook. To see the code simply click on the cell bracket below and fron the "Cell Menu" go to Cell Properties → Open.

This notebook contains the code for a two new commands BilliardsPath and AnimateBilliards which show the path of a billiards ball (treated as a geometric point) on an arbitrarily-shaped table. BilliardsPath shows the path as a single graphic and AnimateBilliards shows the path as an animation. The format for the BilliardPath command is

BilliardPath[*list of inequalities in x and y, x, y, initial point, initial path angle, velocity, maximum time*]

where the following requirements must be met (and are checked for in the program):

1) The inequalities use ≤ or ≥ to define the billiards table.
2) The inequalities define a bounded region.
3) If each inequality is changed into a curve, the curve has a well-defined tangent line everywhere.
4) The initial point satisfies all of the inequalities.
5) *x* and *y* are both symbols.
6) The coordinates of the initial point, the initial angle, the velocity, and maximum time are numbers.
7) The maximum time is positive.

BilliardPath uses NDSolve to find the path of the ball so small numerical errors may creep into the ball path, especially over long run times. In addition there may be some cases where the collision of the ball and the boundary is not properly detected (this may happen in certain cases if the ball approaches an inflection point on the boundary on a near-tangent path).

AnimateBilliards has the same requirements as BillardPath and has the format

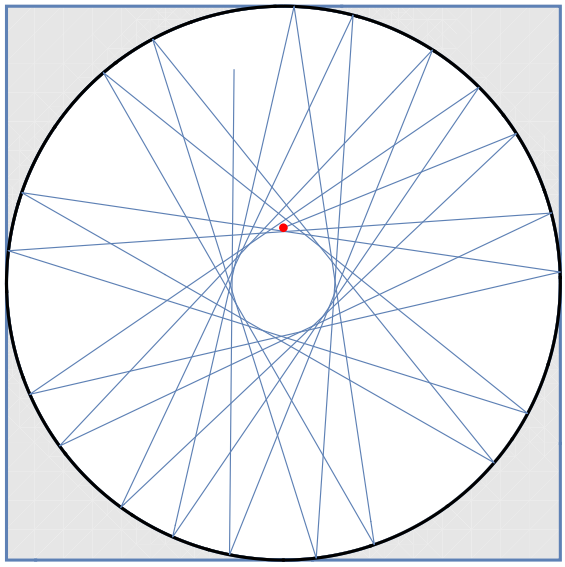
AnimateBilliards[*list of inequalities in x and y, x, y, initial point, initial path angle, velocity, maximum time, frames*]

where *frames* is the number of frames per second used in the animation. AnimateBilliards will take longer than BilliardsPath, as a large number of graphics need to be stored and rendered in an animation.

Both BilliardPath and AnimateBilliards accept various options for ContourPlot such as Frame, PlotRangePadding, and so on - these simply need to be included after the final "required" inputs for either command.

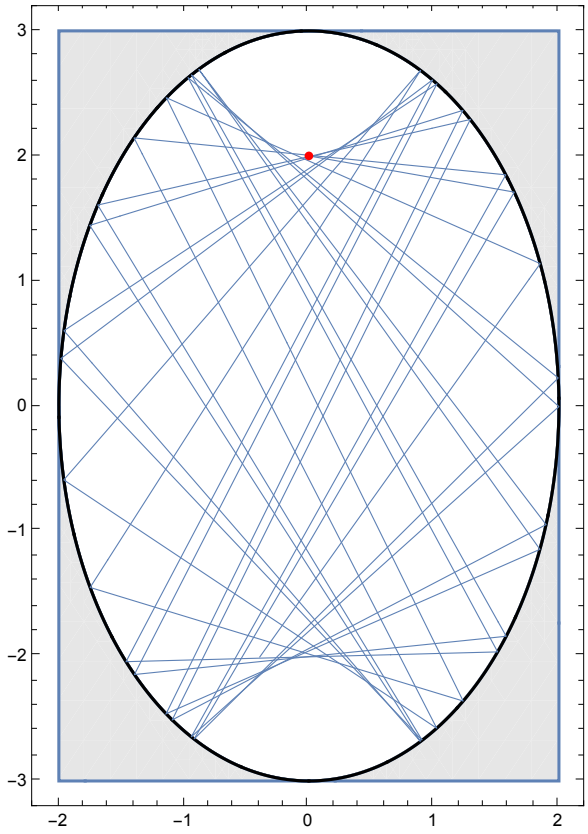
Here is an example of a finding the path on a circular billiard table of radius 5, where the ball starts at (0, 1), has an initial angle of 22 degrees, a velocity of 10 units/second, and a maximum time of 20 seconds:

```
BilliardsPath[{x^2 + y^2 ≤ 25}, x, y, {0, 1}, 22 Degree, 10, 20]
```



Here is an example of an elliptical table defined by $\frac{x^2}{4} + \frac{y^2}{9} \leq 1$, where the ball starts at the point (0, 2), has an initial inclination of $-10\,°$, a velocity of 5 units/second, and runs for 30 seconds, with a frame drawn around the graph for reference:

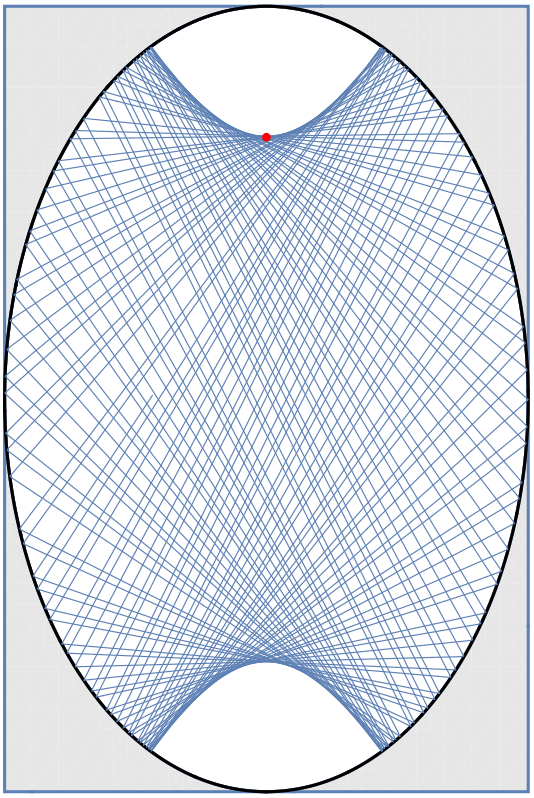
```
BilliardsPath[ {x^2 / 4 + y^2 / 9 ≤ 1}, x, y, {0, 2}, -10 Degree, 5, 30, Frame → True]
```



Here is the same path, but for 300 seconds:

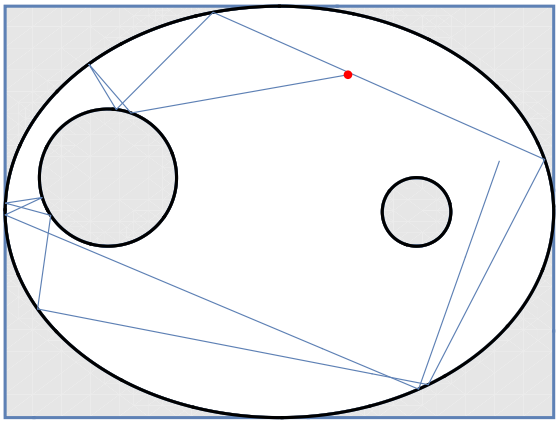
```
BilliardsPath[{ x^2 / 4 + y^2 / 9 ≤ 1}, x, y, {0, 2}, -10 Degree,
 5, 100, PlotLabel → "A long path on an elliptical table"]
```

A long path on an elliptical table



Here is an example of using more than one inequality to define the table - in this case the table is an ellipse with 2 circles removed from it:

```
BilliardsPath[ {x^2 / 16 + y^2 / 9 ≤ 1, (x - 2) ^2 + y^2 ≥ 1 / 4,
   (x + 2.5) ^2 + (y - 1 / 2) ^2 ≥ 1}, x, y, {1, 2}, 190 Degree, 7, 5]
```

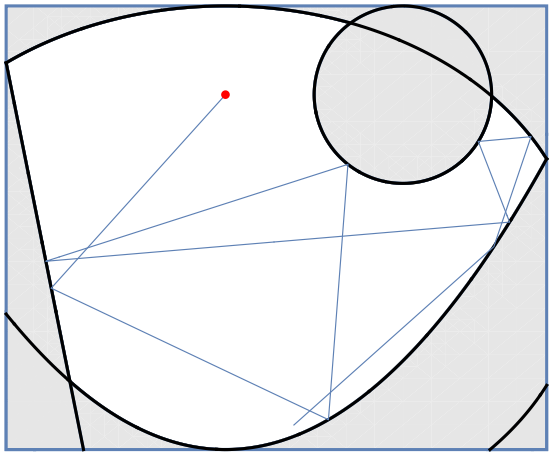


Here is an example of using AnimateBilliards to show the path of a ball bouncing around the same table for a longer time period. You will need to evaluate the command as the animation results in a fairly large file.

```
AnimateBilliards[ {x^2 / 16 + y^2 / 9 ≤ 1, (x - 2) ^2 + y^2 ≥ 1 / 4,
  (x + 2.5) ^2 + (y - 1 / 2) ^2 ≥ 1}, x, y, {1, 2}, 190 Degree, 7, 10, 12]
```

Here is an example in which the table is defined with parabolic, circular, straight, and elliptical edges:

```
BilliardsPath[ {x^2 / 16 + y^2 / 9 ≤ 1, y ≥ x^2 / 4 - 2,
  (x - 2) ^2 + (y - 2) ^2 ≥ 1, y ≥ -5 x - 10}, x, y, {0, 2}, 228 Degree, 8, 3]
```

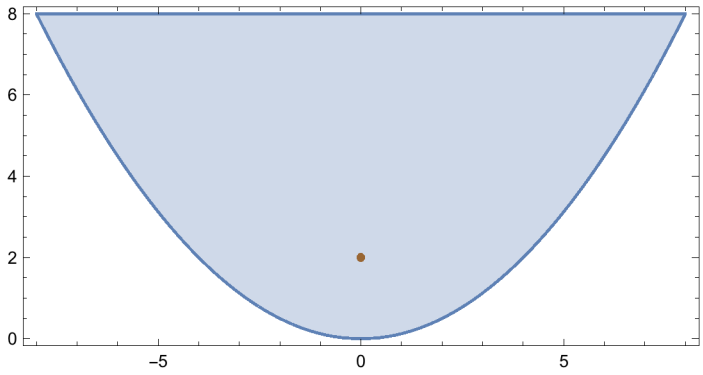


Applications - The Reflective Properties of Conic Sections

The BilliardsPath command can be used to look at some applications of the conic sections.

One important application deals with parabolas. Even though most people tend to think of a parabola as the graph of an equation of the form $y = a\,x^2 + b\,x + c$ the real definition is that a parabola is the set of all points which are equidistant from a fixed point (called the *focus*) and a line that does not go through that point (the *directrix*). For example consider the parabola $y = \frac{x^2}{8}$. The focus for this parabola is (0, 2) and the directrix (not that we will need it) is $y = -2$. Let's consider the billiards table which is defined by $y \geq \frac{x^2}{8}$ and $y \leq 8$ and place the focus in the picture as a brown dot:

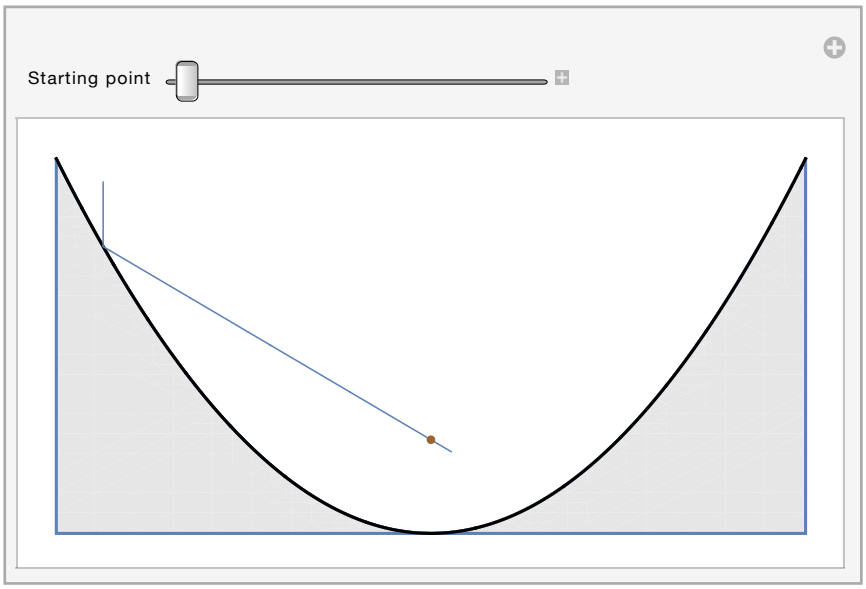
```
RegionPlot[ y ≥ x^2 / 8 ⋀ y ≤ 8, {x, -8, 8}, {y, 0, 8},
 AspectRatio → Automatic, Epilog → {Brown, PointSize[Medium], Point[{0, 2}]}]
```



Now think about placing a ball anywhere near the "top" of the table and hitting it directly down. It will bounce off the parabola, but what path will it take? We can visualize this with a manipulation with a slider that controls the

starting point of the ball:

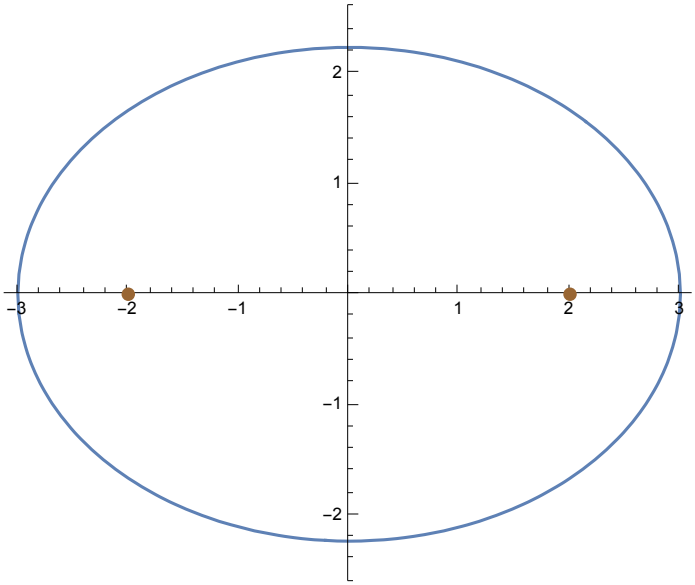
```
Manipulate[BilliardsPath[ {y ≥ x^2 / 8, y ≤ 8}, x, y, {place, 7.5}, 270 Degree, 10,
   1, Epilog → {Brown, PointSize[Medium], Point[{0, 2}]}, ImageSize → 400],
 {{place, -7, "Starting point"}, -7, 7, .1}, SaveDefinitions → True]
```



No matter where the ball starts it will always reflect through the focus. This is why the point is called the focus - if you think of the ball path as a ray of light, then light which is shown directly down the parabola will be focused down to a single point (and the travel time time to the point is also the same). This has lots of applications - if you look at satellite dishes or radio telescopes they all have parabolic cross sections. That is how the amazingly weak signals from satellites or distant galaxies can be picked up - the fait signals received across the entire dish are all focused at a single spot, creating a strong signal which can be picked up by a receiver placed at the focus. The same process works in reverse - if the parabola is a mirror and a bright light is placed at the focus, then the light will bounce off the parabola and the reflected rays will all be parallel as they leave the parabola. This provides the concentrated light beams used in search lights and high-end flashlights.

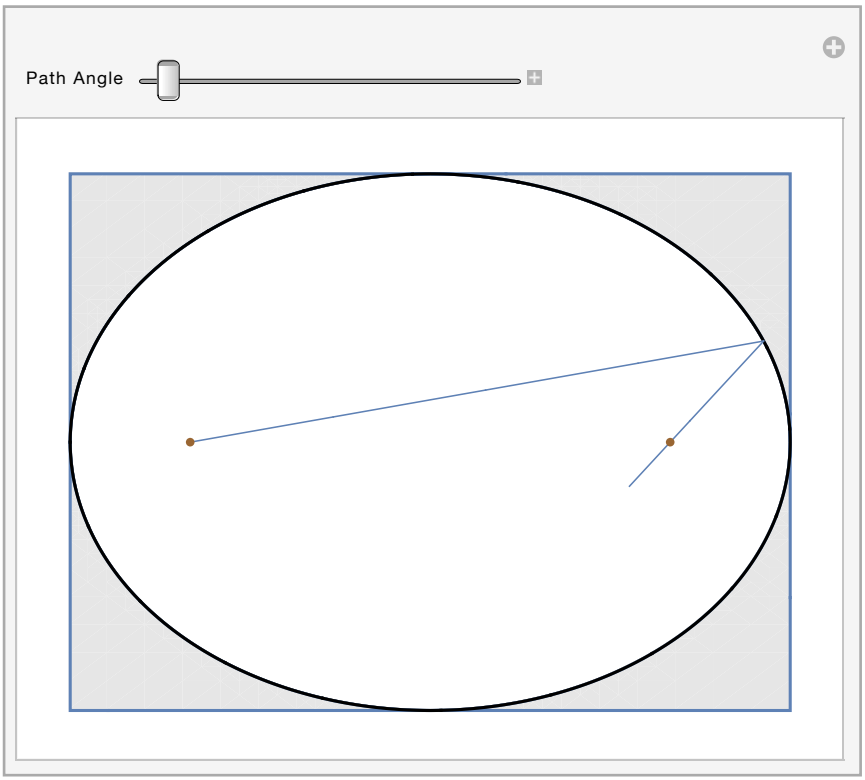
Something similar happens with an ellipse. An ellipse is defined by two points (the *foci*) and a fixed distance d greater than the distance between the two points. The ellipse is then all points P such that the distance from P to the foci add up to d. Here is the ellipse defined by $\frac{x^2}{9} + \frac{y^2}{5} = 1$, which has foci at $(\pm 2, 0)$:

```
ContourPlot[ x^2 / 9 + y^2 / 5 == 1, {x, -3, 3}, {y, -2.5, 2.5},
 AspectRatio → Automatic, Frame → False, Axes → True,
 Epilog → {PointSize[Large], Brown, Point[ {{2, 0}, {-2, 0}}]}]
```



Now place a ball at the left focus and let it go in any direction, with the angle of the path controlled by the slider below:

```
Manipulate[BilliardsPath[ {x^2 / 9 + y^2 / 5 ≤ 1}, x, y, {-2, 0}, eangle Degree, 6.5, 1,
  Epilog → {Brown, PointSize[Medium], Point[{{-2, 0}, {2, 0}}]}, ImageSize → 400],
 {{eangle, 10, "Path Angle"}, 0, 360, 1}, SaveDefinitions → True]
```



No matter what the path angle is the ball will always pass through the other focus (and the time it takes to reach it is always the same, so balls starting at the same time but in different directions would collide at the other focus). If the ball is replaced with a faint sound wave emanating from the left focus that would mean the sound would be reassembled perfectly at the second focus. In architecture this is one way to produce a "whispering gallery" - a room where one person whispering can be heard perfectly at another spot in the room, even if they cannot be heard by someone standing in between them. This is also used with ultrasound in a medical device to shatter kidney stones. The ultrasound generator is placed at the focus of the left half of the ellipse, and then the device/ellipse is moved so the stone is where the right focus would have been. The ultrasonic waves are only tightly focused and in phase at the right focus, where they shatter the stone without harming the intermediate tissue.

Author: Christopher Moretti
Revision date: August 4th, 2015