Draw Curves

Script Use

Draw_parabola(x1, x2, A, B, C, xstep, relative, xshift)

Draws a parabola within the domain of x1 and x2. A, B, and C are constants pertaining to the equation y=Ax^2+Bx+C. xstep is the change in x between evaluations. An xstep of 1 would mean x is evauluated at x1, x1+1, x1+2, and so on. Relative is a Boolean that, when set to false, will evaluate the function at x1 as x1. When set to true, the function will be evaluated at x1 as 0. xshift is the shift in x position the parabola will have.

Draw_parabola_part(x1, y1, x2, y2, constant, xstep)

Draws a parabola connecting (x1,y1) and (x2,y2). Since an infinite number of parabolas connect two points, you must define the constant used. Keep in mind that two parabolas with the same constant in different locations will look different. xstep is the change in x between evaluations. An xstep of 1 would mean x is evauluated at x1, x1+1, x1+2, and so on.

Draw_nic_function(x1, x2, A, B, C, …, xstep, relative, xshift)

Draws a function of n power within the domain of x1 and x2. The other constants are defined as extra arguments in … argument. xstep is the change in x between evaluations. An xstep of 1 would mean x is evauluated at x1, x1+1, x1+2, and so on. Relative is a Boolean that, when set to false, will evaluate the function at x1 as x1. When set to true, the function will be evaluated at x1 as 0. xshift is the shift in x position the parabola will have.

To draw .01x^5+.02x^4-5x^2+x-5 in the domain (0:100)

draw_nic_function(0, 100, 0.01, 0.02, 0, -5, 1, -5, 1, false, 0);

draw_*_width() functions are the exact same as their counterparts, but you can define a width.

Draw_arc(x1, y1, x2, y2, precision, flip)

Draws an arc from (x1, y1) to (x2, y2). Precision dictates the step size, where the larger the value the greater the precision. Flip is a Boolean that controls which way the arc is drawn. This would be the difference from the arc curving to the right or curving to the left.

Draw_arc_length(x, y, radius, length, direction, precision, outline)

Draws an arc starting at (x,y) with arc length 'length.' This will essentially draw part of a circle. Direction (degrees) is the starting direction the drawing will happen. A direction of 0 means the circle will start drawing as if (x,y) is on the top of the circle. Precision dictates the step size, where the larger the value the greater the precision. Outline is a Boolean when set to true, will draw the just the outline, but false will draw the arc filled to what would be the center of the circle.

Draw_arc_angle(x, y, radius, angle, direction, precision, outline)

Draws an arc starting at (x,y) with arc angle 'angle.' This will essentially draw part of a circle. Angle and Direction arguments both take degrees, not radians. Direction is the starting direction the drawing will happen. A direction of 0 means the circle will start drawing as if (x,y) is on the top of the circle. Precision dictates the step size, where the larger the value the greater the precision. Outline is a Boolean when set to true, will draw the just the outline, but false will draw the arc filled to what would be the center of the circle.

Draw_ellipse_angle(x, y, xaxis, yaxis, angle, direction, precision, outline)

Draws and arc starting at (x,y) with arc angle 'angle.' This will essentially draw part of an ellipse. xaxis and yaxis are the distances from the center to the edge of the ellipse. Angle and Direction arguments both take degrees, not radians. Direction is the starting direction the drawing will happen. A direction of 0 means the ellipse will start drawing as if (x,y) is on the top of the ellipse. Precision dictates the step size, where the larger the value the greater the precision. Outline is a Boolean when set to true, will draw the just the outline, but false will draw the arc filled to what would be the center of the ellipse.

Draw_*_width() functions work the same as their counterparts, but width argument has replaced outline argument.

Draw_throw_arc(x, y, hspeed, vspeed, gravity, step, object, (optional) vspeed max)

Draws a trajectory starting at (x,y). Step is the step size for time. A larger step size will result in a less accurate trajectory. Object is an instance id, object, or all instances that the trajectory can check collisions against (and stop the trajectory drawing based on that collision). Vspeed max is an optional argument that is terminal velocity.

Draw_throw_points(x1, y1, x2, y2, speed, speed type, gravity, step, object)

Draws a trajectory connecting two points. Speed is speed in either hspeed or vspeed. Speed type dictates whether the speed you input is the hspeed or vspeed. A speed type of 1 makes it the hspeed, a type 0 makes it a vspeed where the other point is connected after the peak, and a type negative 1 makes it a vspeed where the other point is connected before the peak. You should use -1 if the second point is close to the first point.

Note: if your speed type is 0 or -1, and your vspeed is not small enough to reach the second point, the program will compensate for this. Similarly, if your hspeed is going the wrong way, the program will move backwards in the trajectory.

Draw_throw_bounce(x, y, hspeed, vspeed, gravity, step, object, hbounce, vbounce, (optional) step max, (optional) vspeed max)

Draws a trajectory starting at (x,y) with bouncing. Step is the step size for time. A larger step size will result in a less accurate trajectory. Object is an instance id, object, or all instances that the trajectory can check collisions against (and bounce based on that collision). Hbounce and vbounce are arguments that tell how 'elastic' the bounce is. A bounce of 0 will result in no bounce, 1 will result in full bounce, and 2 will result in double bounce (you can have any value, but usually keep it between 0 and 1). Step max is the maximum number of steps that will take place. Having a larger step size will not make a larger trajectory. Default is 150 steps. Vspeed max is an optional argument that is terminal velocity.

Draw_throw_*_width() functions work the same as above, but you can define a width.