A project by Alejandro Gutierrez and Isaac Burnett

We have created an interactive canvas that allows users to generate and move balls, which collide with user drawn lines.

The user may left click and hold to create a ball; by releasing the mouse, the ball is given a velocity calculated from the distance between click and release events. Balls can be deleted by right clicking, or pressing backspace/delete to remove the newest ball. By left clicking and holding on an already existing ball, the velocity can be reassigned.

The user may press space to swap to line mode, where left clicking creates a vertex. Each subsequent left click generates a new line segment; right clicking while drawing a line ends the current line. Right clicking on a vertex removes any connected lines.

Pressing P opens up a “pause” (the sketch does not actually pause) menu, where the user may mute the sound or save the current lines to a .csv file. Saving creates a .csv file, with the specified filename, in the /data directory. This file can then be loaded after relaunching the sketch. Saving adds a .cvs file extension if the user did not include one in the filename.

The start screen includes a button to list instructions as well as load a previously saved drawing. Loading requires a .csv file in which each row is composed of x1,y1,x2,y2 points to generate one line segment. The user is alerted if they fail to provide a valid filename; a try-catch block is used to prevent the sketch from crashing when the user attempts to open a file that does not exist.

The line class is a simple class that tracks the two points that comprise each line segment as well as the angle between them. Almost all functionality for lines is handled by the mouse events and the draw() loop. Line drawing and deletion has a “snapping” functionality, where clicking within a small area around a vertex will use the vertex position, rather than the exact mouseX and mouseY positions, for generation or deletion. While drawing a line, a trace line between the last vertex and the current mouse position is drawn on the screen to show the user what the line will look like.

The ball class keeps track of the ball’s current x and y positions, and their x and y velocities. There is a display function to draw the ball normally, a function to animate the ball in mid bounce, a move function to move the ball according to its x and y velocities, an end reflect function to bounce the ball off vertices, and a reflect function to bounce the ball off lines and the boundaries of the screen. The creation and storing of balls in the arrayList is handled in the main file.

The tail class creates the tails of the ball. The tail’s position, length, and direction is based on the ball’s position and velocity. The tail has an update function to change its length and direction if the ball’s velocity changes, and a display function to draw it. The creation and storing of tails is handled in the main file alongside the ball creation and storage.

References:

Sound effect courtesy of djfroyd on freesound.org:
https://www.freesound.org/people/djfroyd/sounds/317247/

Simplifica font courtesy of FONTOM:
https://www.behance.net/gallery/14209843/SIMPLIFICA-Typeface-Free

Ball collision code adapted from Jeffrey Thompson:
http://www.jeffreythompson.org/collision-detection/line-circle.php

Line intersection code adapted from the example at processingjs.org:
http://processingjs.org/learning/custom/intersect/