dot-dot-dot

Eric Li, Lindy Zeng, Kaijia Tian

05/16/2017

COS 426: Computer Graphics

http://lindyzeng.me/dot-dot-dot/

Intro

dot-dot-dot is a game based off of the addictive puzzle game Dots, where the player tries to connect dots of the same color on a 6x6 grid of randomly colored dots. A greater number of connections result in more points, and a fully connected set of points receives a special bonus. In dot-dot-dot, we add in particle motion, including particle collision with other particles and with walls to make the game more challenging, but also more fun. Instead of having a static system of just a grid, we now have a dynamic system of moving particles, which hopefully better engages the player. People of all ages can benefit from this game, since it is easy to play, requires little practice, and involves some strategy.

A similar game, called Swipe Off, has been implemented in the past, but it is currently unavailable in both the App store and the Google Play store. The only remaining evidence that this game has ever existed is from a Youtube video. Swipe Off seems to be a simpler version of dot-dot-dot, in that the dots can overlap and do not bounce off each other.

We implement our game dot-dot-dot in JavaScript using Pixi, an efficient 2D-sprite rendering engine. We reorganized some built-in functions of Pixi to make our code more modular and easier for debugging, and also wrote our own method for detecting collisions, and particle physics. Check out dot-dot-dot here.

Methodology

The first step to rendering dot-dot-dot is setting up the stage and all the pieces. We rendered the walls as rectangles along the perimeters of the screen and the dots as one of four randomly colored circles of random sizes. The positions of the dots were determined using random numbers for the x- and y- coordinates. To ensure that the dots do not overlap each other upon initialization, we implemented a naive quadratic

algorithm that iterates through all existing dots and only places the new dot if it does not overlap with any other the other dots. The particles are simulated as undergoing perfectly elastic collisions on a frictionless surface. Their positions are calculated according to the laws of physics, specifically the laws of conservation of mass and conservation of momentum, and are updated 60 times per second to simulate movement. We could have made the collisions inelastic and the surface have friction, but that would decrease the velocity of the particles with each collision, eventually leading to a static game board, which would defeat the purpose of making a dynamic game.

Next, we implemented connections between dots. When the player holds down the mouse and drags to connect dots of the same color, a line connecting the two dots is rendered onto the stage. We also chose to have the line follow the mouse when the click is held, so it is clear to the player that he or she can continue to connect dots. There is additional logic to ensure that a player can only connect dots of the same color, and that closed polygon can only be formed by joining the last dot with the first dot in the connection. In addition, a player can undo the connection between two dots by retracing their path from the most recent dot to its previous dot in the connection. Dots are killed and disappear when the player lets go of the mouse. While the dots are being connected, a bar at the bottom of the screen indicates the amount of “rope” still available and the current score. The score is calculated as a function of the radius of the newly connected dot in the chain multiplied by the total number of dots connected. Additional bonuses are given if the player draws a closed polygon with the connected dots. This gives the player an incentive to connect more dots to form longer chains and optimize the number of large dots in their connection while using the shortest amount of rope. The game ends when either the player runs out of rope or no new connections can be made.

The final touches on dot-dot-dot include a title screen where the user presses a play button to signify the start of the game, a game over screen, sounds to signify dots being killed, and background music (*Life of Riley* by Kevin MacLeod). These visual and auditory supplements contribute to the overall aesthetic of the game and enhances the playing experience. Despite its simplicity, dot-dot-dot is a game that requires a certain amount of strategy and engages the player on multiple levels.

Results

After many hours spent making this project, and several more debugging, we are happy to announce that our game is fully functional and can be found at http://lindyzeng.me/dot-dot-dot/.

Figure 1. Title screen where dots are initialized. The game starts once the player presses the play button.

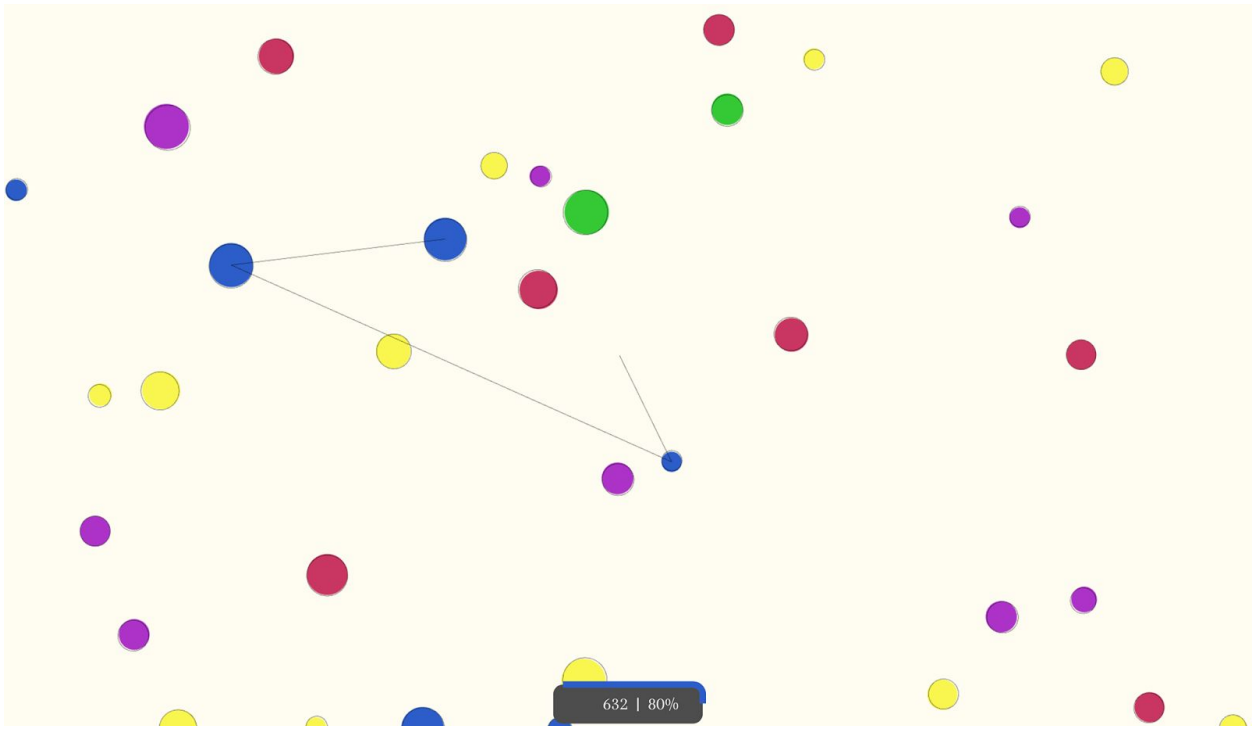

Figure 2. Game play screen. A line is drawn between dots when the player holds down the mouse and connects them. The connected chain disappears when mouse is let go. In the box at the bottom, the number on the left shows the score, and the number on the left shows how much rope is left. The outline of the box is filled depending on the number of dots in the connection.

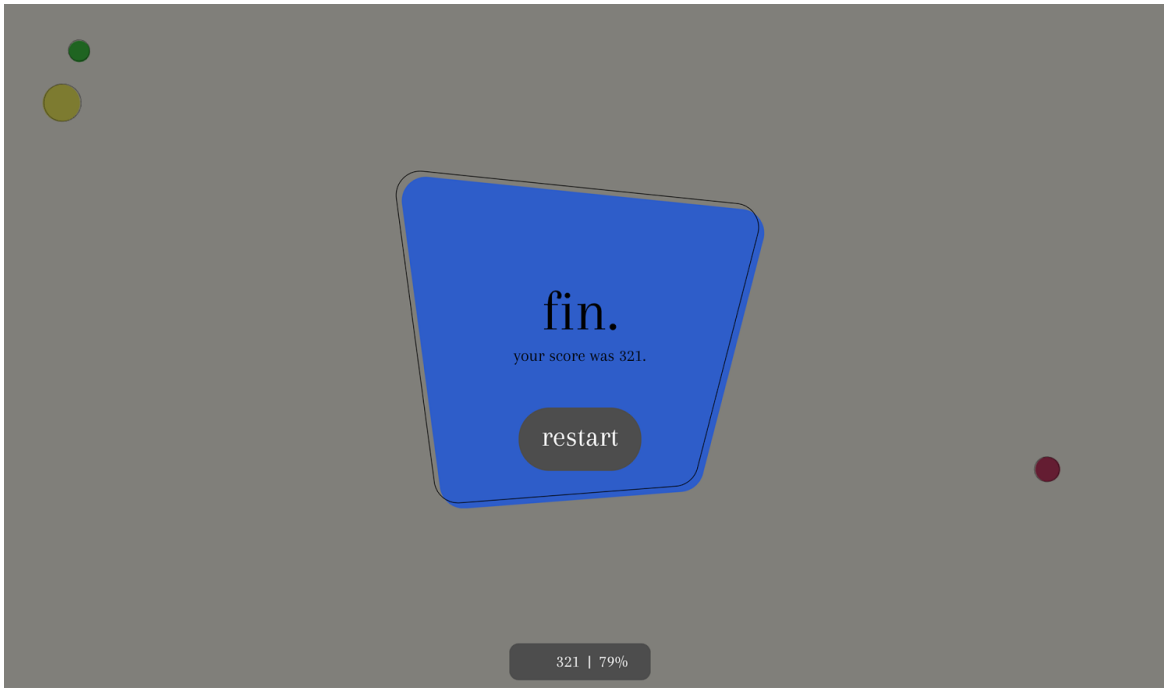

Figure 3. Game over screen. The game ends once there are no more possible connections, or when the player runs out of rope.

Discussion and Conclusion

Overall, we think that this game was a success. While several bugs had to be fixed before the game reached its final mode, we were able to debug the code within reason. Modularizing the code was a great idea. Although additional classes were created as a result of this decision, it simplified the coding experience by keeping individual classes separate from one another, preventing redundancy, and making the code easier to read and debug.

To make the game more complex, special effect dots could potentially be added. These might include color bombs that eliminate a single color from the play space, time bombs that momentarily freeze time, or paint bombs that paints the dots in its immediate vicinity all the same color. Players will have to incorporate the special effect dot into their connection to receive the benefits. In addition, different modes can be added, such as endless mode, battle mode that can be played between multiple players, and level climbing mode where each level introduces new obstacles.

This project gave us an opportunity to develop a game in JavaScript, allowing us to create something tangible using knowledge gained from this class. We also learned how to use

Pixi as a rendering engine, to use the new ECMAScript 6 syntax rather than the JavaScript conventions we were used to in order to make our code more succinct, and evaluate ways in which to engage the audience through gaming mechanics. Overall, this project was a good introduction to game development and design, and we hope you enjoy our game!