Yarn Wars

Created by: Shannon Paul

02/07/2021

Focus Sentence

With the world now ruled by felines and yarn supply lower than ever, two feral cats find themselves throwing paws as they try to collect as many balls of yarn as possible whilst avoiding an out of control laser pointer.

Gameplay Overview

Yarn Wars is a two player game with a concrete **goal** - to collect more balls of yarn than the opposing player. Players have the ability to throw paws at each other to try and take the lead. The **rules** are minimal and simple; if a player collects a ball of yarn their score increases by one, if a player is hit by an opposing paw they lose one ball of yarn and if a player fails to avoid the laser pointer they will lose multiple balls of yarn. Once the laser has completed 10 full rotations the game ends and the player with the most balls of yarn wins. The first time a player tries the game there is an element of **surprise.** After 5 rotations a second laser is added to throw in an extra level of difficulty that can completely change the outcome of the game. This extra difficulty helps with **fairness** because it automatically slows all players down which can even out skill levels.

Players can improve their **skills** the more they practice. Moving while throwing/avoiding paws and lasers can be tricky to manage but can be mastered over time. Players can **choose** to play with a couple different strategies. The faster they move around, the more balls of yarn they are likely to collect however this can be high risk as it increases chances of running into the laser. Moving around quickly might also make it harder to avoid oncoming paws. If the player chooses to move around slower and hang out in “safer zones” they will collect yarn at a slower rate but will be less likely to run into the laser and have an easier time spotting oncoming paws.

Fun

Fellowship plays a massive part in the overall design of the game. Yarn Wars was designed around being a two player game to create a fun social component. In order to win, players have to collect more yarn than their opponents. Players can take away their opponents yarn by throwing paws at them, which can be a very personal attack. Both of these game mechanics have the purpose of invoking competitiveness, a trait that many players look for in games. The element of **challenge** is also very present in Yarn Wars. The game is centered around an obstacle - the rotating laser pointer. Running into this obstacle is very costly creating a fun and challenging aspect to the game. There are also other obstacles such as paws which players must try to avoid.

Yarn Wars explores some aspects of **narrative**. The narrative involves two floating cat heads that are at each other's throats over balls of yarn. Since cats love playing with laser pointers - but ultimately research has shown it can lead to anxiety in cats - it is only fitting that the main obstacle is a laser pointer. This is a very silly and lighthearted narrative that only adds to the fun of the game - especially for cat fans. The easy-goingness of the game also plays into **submission.** The game is very simple so

anybody could play it, as well each round goes by very quickly which makes it easy to play as a pastime game.

Technology

Yarn Wars was coded using the class methods approach. The goal of this approach is to reuse as many functions as possible. Projectile, playerOne and playerTwo classes were all designed to be compatible with GameCircle and Rotating line functions.

playerOne/playerTwo:

Data:

- Parameters: position, radius
- Movement attributes: currMovement, movement, left/a, right/d, up/w, down/s
- Other: velocity, colliding, score, point, img, shot
- List of projectiles

States:

- Colliding: if its colliding with a line
- Movement: whether stationary or moving
- Point: collected a ball of yarn
- Shot: was hit by projectile

User Input:

- Left: left arrow key or a moves player's position to the left
- Right: right arrow key or d moves player's position to the right
- Up: up arrow key or w moves player's position up
- Down: down arrow key or s moves player's position down

Updates:

- Colliding: check if colliding with line
- Point: collected a ball of yarn
- Shot: was hit by projectile
- currMovement/movement
- Projectiles: 3 allowed at a time

Rendered:

- Score is displayed on screen and updated when colliding/shot or increasing
- If colliding with line it turns red
- self.img is displayed

Projectile:

Data:

- Parameters: position, radius
- Other: facing, velocity, shooting, img

States:

- Shooting: if projectile is still shooting

User Input:

- m/q: adds projectile to player's list of projectiles and shoots it

Updates:

- Shooting: if projectile has collided with a player/wall it is no longer shooting and is removed from the player's list of projectiles

Rendered:

- If projectile is shooting self.img is displayed

GameCircle:

Data:

- Parameters: position, radius
- Other: colliding, hit, img

States:

- Colliding: if colliding with line
- Hit: if colliding with player

Updates:

- Colliding: check if colliding with line
- Hit: check if collected by player
- Position: reset position if colliding, hit or if line completes rotation

Rendered:

- If circle has not been hit or collected self.img is displayed

RotatingLine:

Data:

- Parameters: origin, angle, intervals, segments, collided, color, length (made parameters so they could easily be changed when initialized)
- Other: rotation

States:

- Collided: check if collided with ball

Updates:

- Line_did_reset: Check if line has completed a full rotation

Rendered:

- Two segments of the line are displayed in red
- The amount of rotations remaining are displayed