Project Proposal: Grow

Simin Li

Introduction

Project Goal

The goal of this project is to make a game that implements machine learning like this game Google featured during Halloween https://www.google.com/doodles/halloween-2016. The player is required to draw with their mouse or ideally a tablet the shapes that appear on the screen before the bugs that carry those shapes harm the player's plant. If the player is too slow to correctly draw the shapes, the bugs will harm the plant. Players may use basic operators like "+" and "×" to save time and get extra points. For example if you see three triangles in a row, you could draw one triangle and add a "× 3" in the end. The system will have to act as an image calculator, which increases the difficulty of this project. Similarly, you can use "+" to eliminate two shapes at once. This will not necessarily save you time, but it will boost your score.

Personal Interest

I was interested in doing something related to Machine Learning because I like drawing and thought it would be really cool to make a drawing game. The recognition of simple shapes, operators and numbers is a huge challenge for me but I am sure the results will be meaningful and rewarding. I played the game Google had on Halloween and thought that it was very unique. It is a great way to implement machine learning and game play.

Organization

Main Menu

This will contain a play button that links to the gameplay mode, and a help button that links to the help screen. The buttons may be replaced with a picture from GIMP.

Help Screen

This will contain the instructions on how to play the game.

Paused

This will be a screen that says "Paused" and "Hit 'h' for help and 'p' to continue ".

Game Play Mode

Each Level will have a plant in the middle of the screen and bugs that carry different shapes awaiting to be drawn heading towards the plant. As the level increases, the combinations get harder and the bugs crawl faster. You can also see the plant grow in size as you progress. There will be 5 levels in total. The number of lives will be displayed at the top left corner (Each player starts with 5), and the score will be displayed at the top

right corner. At the bottom of the screen display “Hit ‘h’ for help and ‘p’ to pause ”.

Game Over

This will contain a replay button and a help button. The replay button will redirect to the main page and the help button will link to the help screen. This screen will display the player’s final score.

Graphics Implementation

Plant

This will be a class containing the information of the plant including level, lives, score, location, a draw(self, canvas) method, and isDead(self) method. Lives and score will be reset each level and level will be updated.

Bug

This will be a class containing the information of a bug including location, speed, remainingShapes (0-9), a draw(self, canvas) method, eliminateShape(self, event) and isDead(self) method. This is the meat of the project, which includes detecting legal input and removing the shapes from the remainingShapes list. This will be further discussed in the Machine Learning section.

Background

This will be green.

Button

This will be made with tkinter and possibly replaced with images made with GIMP.

Machine Learning

Testing Data

This project attempts to distinguish numbers 1-9 and simple shapes including triangles, circles, squares and stars and the operators “+” and “×”. This will require lots of testing data. I will have 100 examples for each type of symbol.

Linearization

Each pixel is either colored or not. Use 1 to represent a colored pixel, use 0 to represent an uncolored pixel. Once each pixel has been replaced by a number, the whole image can be represented uniquely by a tuple containing 0’s and 1’s.

Data Storage

I will store this information in 3 lists: flattenedImages, types and output. flattenedImages contains tuples containing 0’s and 1’s (e.g. [(0,0,0,1,1,0,1,1,0,0…),(0,1,0,1,1,0,1,1,0,0…),(1,0,0,1,1,0,1,1,1,0…)…]). Types contains

the corresponding type of the images (e.g. [“triangle”, “triangle”, “+”, “8”…]). Output contains the “distance” between an image and each image in flattenedImages by using the distance formula (math.sqrt((x1-y1)**2 + (x2-y2)**2 + (x3-y3)**2) + (x4-y4)**2 + …(x10000-y10000)**2) and the type inside a tuple (e.g. [(29, “cirlce”), (50, “+”), (90, “9”)…]).

K-Nearest-Neighbors

Takes the 5 closest data points and see which group they belong to: shape, number or operator. Once the type is determined, find the 5 closest data points within that group and determine which specific thing it is. “Nearest” is can be determined easily by sorting Output.