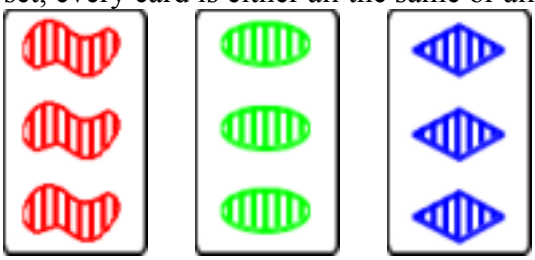
Zach Jablons

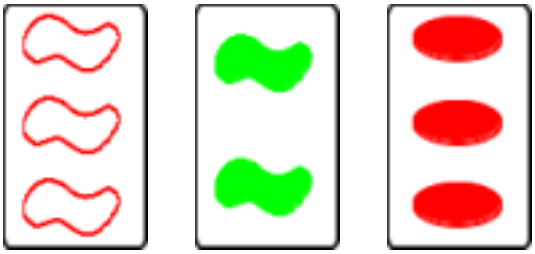
Set AI

This program is a graphical demonstration of the capabilities of the AI Search method by demonstrating its master of the SET card game.

The SET card game features 81 cards with four properties (number, shading, color, shape) and three variations on each property (1, 2, 3; empty, shaded, full; etc.) each. The objective of the game is to identify sets of three cards in a board of twelve cards where, for each property in the set, every card is either all the same or all different. For example, consider this set:



The properties of shape and color are all different, but the properties of shading and number are all the same. This is a valid set. This following set is not a valid set. Can you see why?



If you're stuck, it's invalid since cards 1 and 3 are red, but card 2 is green, meaning that for the color property, they're neither all different nor all the same.

Finding a set in this manner is an exhausting task for humans, but for a Depth First Search algorithm it's trivial. This method works by going recursively from one card to another, adding it to the set if it's not already in the set, then going down another 'level' until it has 3 cards in its set. Once this is done, it sends the set to a checking algorithm which efficiently decides whether or not the provided set is valid.

Unlike in other games, the Depth First Search method and the Breadth First Search method have no real advantages over each other, as a set that has less than 3 cards is invalid, so finding a solution somewhere along the search tree prior to the third level is impossible. It may benefit from some heuristics, but nothing more than a little optimization is possible.

As a challenge, it would be interesting to see a similar game created with a raised the number of sets to compare. This AI Search method is exponential in its upper bound running time, having to compare, for each board of cards, $O(b^n)$ combinations (where b is the size of the board and n is the size of a full set). While the graphical portion of this program would not be able to handle

this, the actual AI solver would. A simple implementation of the AI solver by itself is in setAI.py.

Now obviously, this AI Search method would be unfair if compared to humans, since on top of doing the computations of which sets are valid (which humans seem to screw up on occasion), humans also have to visually process the cards and interpret them, which is confusing. Given a picture of a board, this AI would probably suffer from a buffer overflow trying to read each piece of data in the image as a set, since the current importing method resorts to having neatly processed files of ‘cards’ which are space separated properties separated into cards by newline characters. This simply is not how humans process the SET board at all, instead they try to find patterns in the cards or get a ‘feel’ for which cards make sets. This s clearly less efficient than the AI, but if we were to substitute an experienced SET player as our checking algorithm, arguably they might be able to find a set faster than a human playing SET without any computer aid (that might be fun for future research).