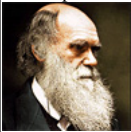

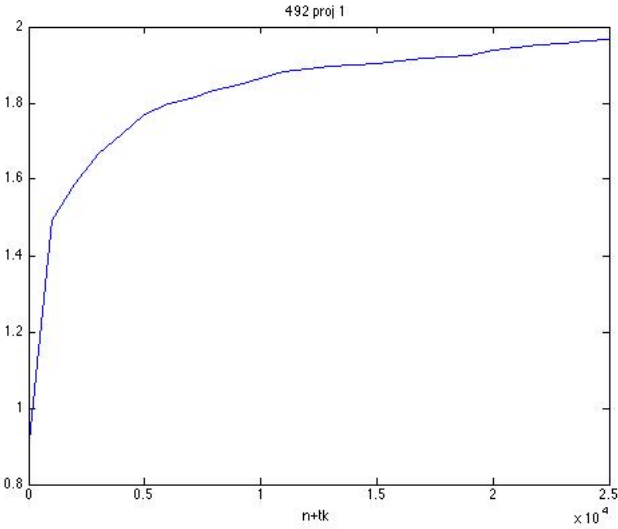
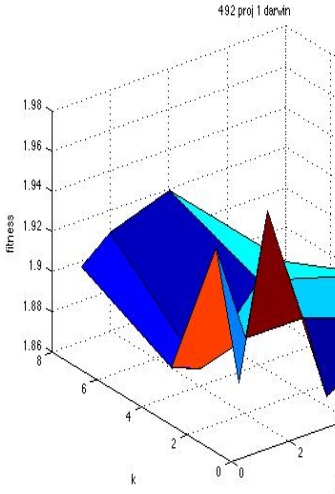

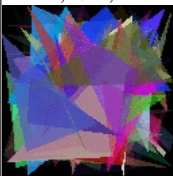
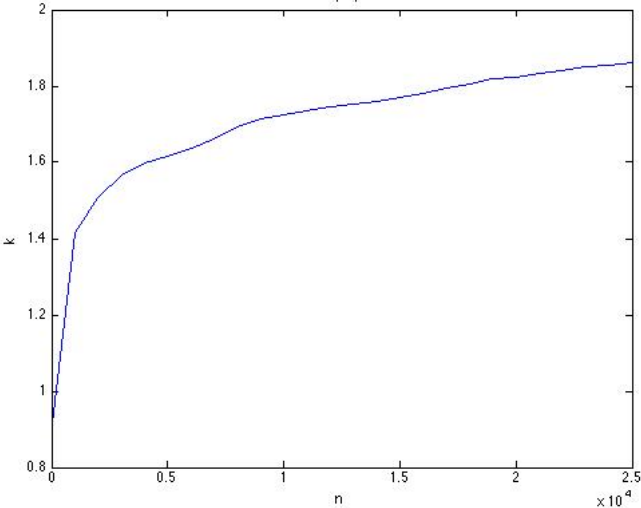
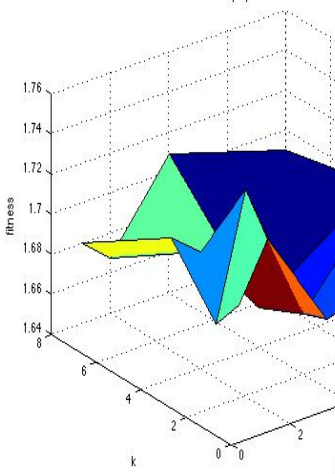


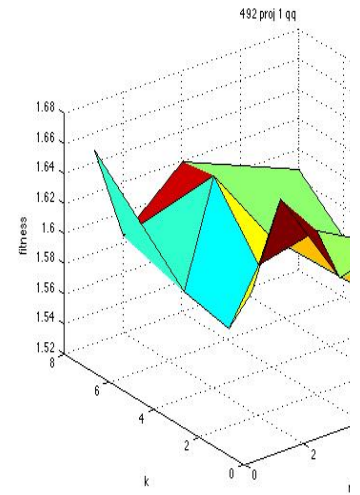
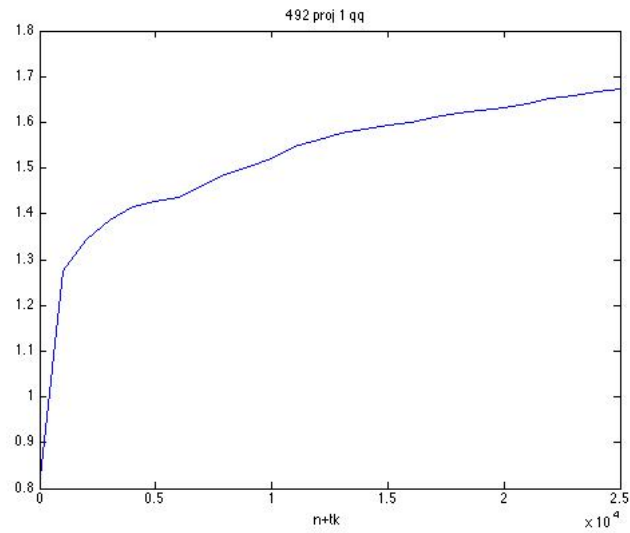


EECS 492 A1 Results
Xi Chen
September 25, 2015

A1 Results

The purpose of this project is to recreate [Roger Alsing's work on genetic programming to create art](#). The population consists of approximations to a given image. Each image uses P polygons, where P=100 for the results on this page. N is the population size and K is the number of new children created per generation. T is the number of generations displayed in each animation. The fitness of the image is a measure of how closely it matches the original image. The learning curve shows how the fitness of the best approximation improves over time. The 3D graphs depict how the fitness after 25000 generations varies with N and K.

Original image	Approximation	Learning curve	
96×96 pixels 	P=100, N=1, K=2 		 N=2, K=1 gave the best result
102×102 pixels 	P=100, N=1, K=2 		 N=2, K=2 gave the best result
102×102 pixels 	P=100, N=1, K=2 		

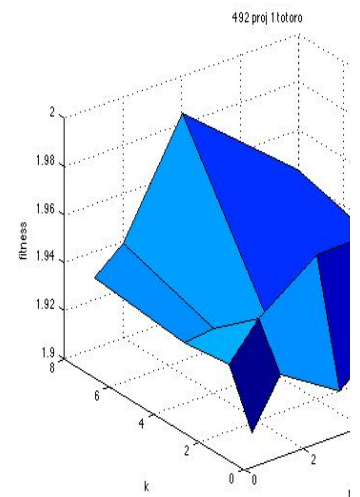
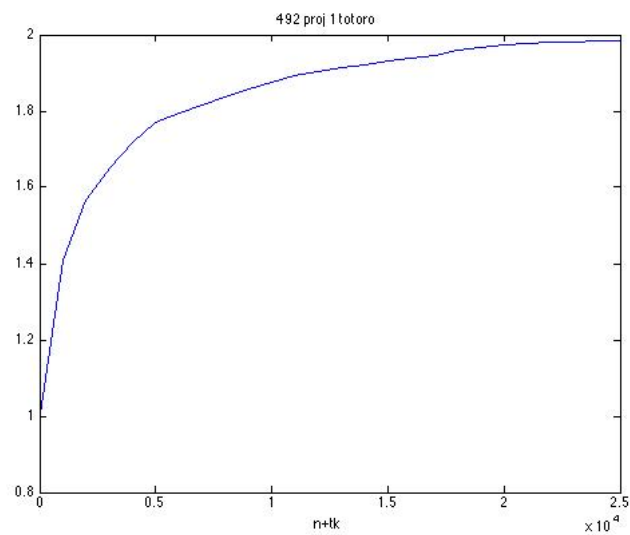


N=2, K=1 gave the best result

128×128 pixels



P=100, N=1, K=2

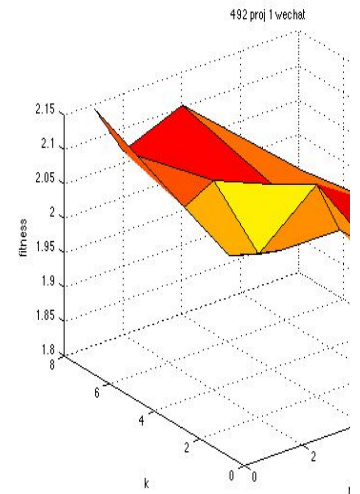
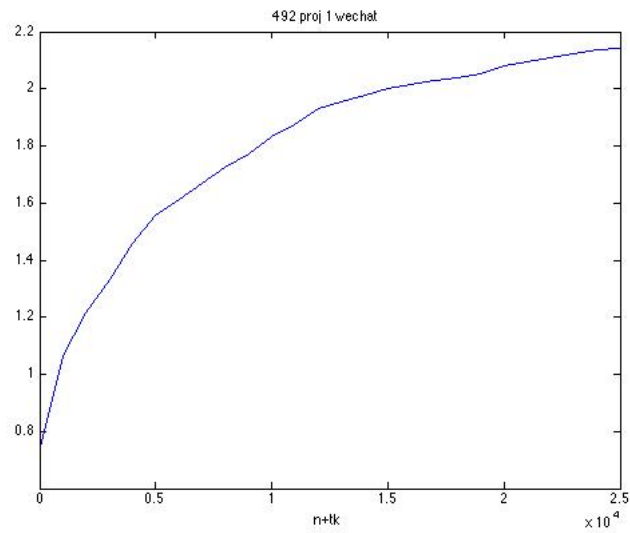


N=4, K=8 gave the best result

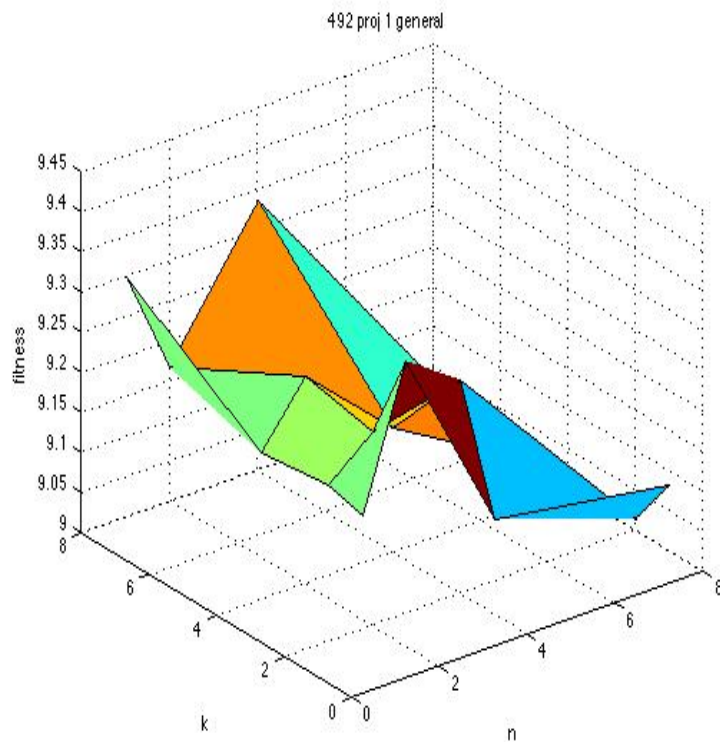
300×293 pixels



P=100, N=1, K=2



N=1, K=8 gave the best result



N=2, K=1 gave the best result overall after 25000 generations

Mutations used:

- With 5% probability, replace a polygon with a random triangle.
- With 5% probability, Swap the positions of two polygons in its ordered list
- With 5% probability, Slightly alter color of a polygon up.
- With 5% probability, Slightly alter color of a polygon down
- With 7*5% probability, combo of any two mutation above (except color up and down), with some variation producing extra mutation
- With 5*5% probability, combo of any three mutation above (except color up and down), with some variation producing extra mutation

Components of colors (0.0 to 1.0 scale) were altered by multiplying 1.2 or 0.8 to the RGB value.

Random triangles were created by picking a point on the image with uniform distribution, Random colors were picked using uniform distribution over the RGB color space.

Parents were selected from an array of individuals sorted in decreasing order of fitness using the index `alphaGen(randEngine)` where `alphaGen(randEngine)` returns a random double uniformly distributed between 0 and 1 inclusive. This made the more fit individuals more likely to be chosen for crossover since I accumulated interval for each possible parent based on their fitness value, so the double is more likely to fall into their intervals (which is larger).