Artificial Intelligence	**Programming Project 2**
CSC3335 Spring 2023(Stuetzle)	**Due Date: February 28th, 2023**

You will work in groups of 2 or 3. Create a repository on github for the project and add me as a contributor. Name your repository with your last names, the class number, and the project number, for example: $< lastnames\_3335\_project1 >$.

1 Description

In this project, you will practice solving an optimization problem using genetic algorithms. Genetic algorithms can be used to generate imagery with particular features. In this project, you will produce a simple image of a circle using a simple fitness function. The process you will implement in this project, however, can be used to generate any images for which there is a fitness function. The fitness function (a measure of how close the current image is to the goal image) is incredibly difficult to determine for complex imagery, but fairly straightforward for something like a single circle.

The image you will create will look something like this:

Credit to shutterstock, obviously.

The way this is done is through implementing the genetic algorithm procedure described in Unit 4. You will produce *and visualize* (see below) a set of 8 random 200 x 200 pixel images. That means, at the start of your algorithm, each pixel in each member of the population is given a random color. Then, you will run the standard genetic algorithm process provided in the book/notes, using the fitness function below for a circle. Additionally, you will visualize your currently population at each time step, meaning that you will produce a sort of "animation" of all eight current images as they converge on circles.

2 Program Development

Use the following guidelines when creating your program:

- Use good object-oriented design principles. The game should be run by a driver object. Parts of the game, like "cards" should be objects. The agent should be its own object that interacts with the game state.
- You *must* use the standard genetic algorithm from the textbook and the unit 4 notes. Throw away your 4 least fit population members and replace them with cross-breeds of your 4 fittest population members each iteration of the algorithm.
- The fitness function for a circle image is the maximum among the *number of non-white pixels equidistant from the center pixel* minus the *number of nonwhite pixels elsewhere*. To do this, you will need to calculate a value for this for each possible radius (distance from the center pixel) and take the maximum over all the radii tested. You can/should do this for at least 10 radii, but don't need to try everything (but be consistent in which 10 you are choosing). They should be spread out, so don't just use 10 through 20 or you won't be able to tell the difference.
- You will likely generate not particularly pretty or uniform colors in your circle. For extra credit, you can add to the fitness function terms that will help generate a circle that is more like the example image above, where neighboring pixels have similar colors.
- Please note that this is a *general image generation* algorithm, meaning that if you modularize your code and set the fitness function off to the side (which you absolutely should), you can substitute it for any other fitness function that produces a particular type of image and just run your code to generate different kinds of images. You can look up papers that discuss fitness functions for generating images of oranges, apples, etc. But they are complicated and take longer than the time we have in this class. But it's a cool framework to have access to.

3 Rubric (out of 20 points)

The following is the breakdown of points assigned to the areas of this project. Remember that a project that does not compile will receive none of the available points, and only if the code earns those points in the style guidelines and commenting portion of the rubric.

Points	**Criteria**
5	Initial population built and visualized correctly
10	Algorithm iterates correctly using the process for a genetic algorithm
5	General efficiency, object-oriented principles, and code quality
3	**Extra Credit:** Circles generated produce color patterns