GENETIC ALGORITHM DOCUMENTATION

By Karl Boghossian

GENETICALGORITHM.h

Chromosome structure:

- Contains a random size array of bits (0 or 1) which constitute the chromosome.
- ***Gene_Count*** is a variable that holds the number of genes per chromosome.
- Contains a variable that holds the number of bits per gene. So you can set that all the bits of the chromosome correspond to one gene each.
- ***Bit_Count*** is the size of the array of bits.
- ***Fitness*** is the chromosome fitness score (used by the fitness function).

- ***Default constructor*** that initializes the members
- ***Create_Chromosome*** takes the number of genes per chromosome, the number of bits per gene and a bool to know if the chromosome contains a duplication of genes or not. If we allow duplication we just fill random sequence of bits else we create a gene then insert it in the chromosome if it wasn't previously.
- ***Decode*** is a function that takes an array of bits and transforms it into its corresponding decimal value.
- ***Decode overloaded*** is a function that takes a decimal value to convert it to an array of bits.
- ***DecodeGene_FromGenesArray*** takes a genePos in the array of bits then decodes the gene according to the gene's length.
- ***DecodeChromosome*** decodes gene by gene into decimal and add them to an array of integers to return it.

- ***Get_GenePosition*** takes the decimal gene value and return the corresponding gene's position in the bit array.
- ***Set_GeneValue*** is a function that takes the gene value to set in the gene's array at a destination position.
- ***ValueFoundInArray*** takes an array of decimals and its size then searches for a specific value if it is found in the array of decimals (used in the create chromosome function to know if there is any duplication).
- ***SwapGenes*** is a function that swaps all the bits of 2 specific genes together.
- ***InheritFrom*** is a function that copies all the properties of the source chromosome to the THIS properties.
- ***MoveGene*** takes the gene that must be moved (from) and inserted into the gene position (to). It checks whether we are moving a gene from left to right or vice versa.
- ***Destroy*** plays the role of a destructor by deleting the genes array and setting the pointer to NULL.

Genetic class:

N.B: this is a base class that will have derived class from it.

- Contains the ***population*** that is made of chromosomes (random size).
- ***Population size*** that holds the number of chromosomes in the genetic algorithm.
- Holds the number of genes per chromosome.
- Holds the number of bits per gene in the chromosome.
- ***Generation_Count*** is the number of generations to pass through.
- ***BestFitChromosome*** is the index of the chromosome that has the highest fitness value.
- ***BestFitnessScore*** is the highest fitness value.

- ***WorstFitnessScore*** is the lowest (obvious).
- ***TotalFitnessScore*** is the total fitness value (used by the roulette selection).
- ***Mutate ptr*** is a pointer to a mutate function that will be picked from the used while the application is running. (All the corresponding functions are in the code and fully commented)
- ***Crossover ptr*** is a pointer to a crossover function that will be picked from the used while the application is running. (All the corresponding functions are in the code and fully commented)
- ***Replacement ptr*** is a pointer to a replacement function that will be picked from the used while the application is running. If the pointer is NULL → total replacement will be used. (All the corresponding functions are in the code and fully commented)
- ***Selection ptr*** is a pointer to a selection function that will be picked from the used while the application is running. (All the corresponding functions are in the code and fully commented)

- The ***constructor*** that takes the population size to know how many chromosomes to allocate, the number of generations to loop through, the number of genes per chromosome, the number of bits per gene, if we allow a duplication of genes in the chromosome or not and the probability (percentage) of the chromosome to be mutated. It initializes the members, then allocates the population then creates the initial population.
- ***FitnessFunction*** computes the best and worst fit along with the total fitness score.
- ***Generation*** it is a virtual function because if we have only one population so the genetic algorithm will work in a normal form so the code is set, else it must be overwritten according to the problem criteria. This function is the back bone of the algorithm, it updates the fitness values checks if we have a roulette wheel selection so the value sent in the global pointer would be the total fitness score, else it would be a percentage of the elitism function

(which is 20 in this case). Then it checks if we have a replacement or a total replacement must be used, it loops to create the number of children equal to the population size, picks 2 parents using any selection method, crossover to create 2 new children, mutate the children then add them to the array of new children. After we have a new population we destroy the old one because everything was using a deep copy the swap pointer and finally update the number of remaining generations.

- ***DestroyPopulation*** is a function that loops through all the chromosomes to destroy them one by one.

RACE.h

- Enumeration to hold all the colors in the pool (16 colors).

Cell struct:

- Contains the ***color*** of the cell itself.
- Contains the ***delay*** the cell has.
- ***ColorName*** is the enumeration of the color to know if the cell has which color by just checking the name without checking the R,G,B true color.

RacerPos struct:

- Contains the position in the board (row & column).

Racer class:

- Contains the ***position*** of the racer.
- Contains the ***pointer to which chromosome holding the color priority it is assigned***.
- Contains the ***pointer to which chromosome holding the gene position it is assigned***.

- A bool to know if he's still ***executing a move***.
- The ***delay*** that the racer has to wait on the cell.
- An ***array of 10x10 bools*** to know if the chromosome has visited a cell to not visit it again.

Race class:

N.B: is derived from the genetic class.

- The ***2D array of pointer to cells*** which are picket randomly from the pool of color.
- The ***array of cells*** that is the pool (16 colors).
- The bool that indicates if we found the best solution.
- The ***number of finished racers*** to know if we call the generation function.
- The ***racers*** that will move on the board.
- The ***Population used for the color priority***.
- The ***population used for the position***.

- ***Race constructor*** that calls the genetic constructor, then it initializes its members, creates the pool of 16 colors, allocated the racers and sets their properties and initializes the board to random cells.
- ***ExecuteBestRacerMove*** takes the racer that will make a best move, decodes the chromosome into an array of decimals, checks a top, right and bottom move then takes the best destination position along with the best color priority, at the end it executes the move then sets the executingmove bool to true coz the next time is is called it will wait until the delay reaches 0.
- ***UpdateRacers*** loops through the racers, checks if they are behind the finish line so the racer will execute a move. If the racer has reached the finish line we update the number of finished racers. Then if the number of finished racers is equal to the population size → call the generation loop and reset the racers to their initial position and checks if we reached the base case.

- ***Generation*** is the overwritten function of the virtual function in the Genetic class, because we first are allocating space for 2 arrays of new children (1 for the colors and 1 for the position), then we are checking for replacement, then we are doing the selection to pick 2 parents successively, the same 2 parents are used to crossover to create 2 new children for the color population and 2 new children for the position population, then we check depending on the percentage of the mutate ratio to know if we do some mutation for the colors babies and for the position babies, then we add the 2 colors babies to the population of colors and the 2 position babies for the population of position, finally we update the generation counter.
- ***ResetRacers*** loops through the racers to initialize their members.
- ***Draw*** to draw the board along with the soldiers.