EXERCISE 1 - ALIFE AND GENETIC ALGORITHMS: EVOLUTION VIA USER SELECTION (PART 2)

1. Complete genetic algorithm

Last week we started developing a Flash movie for the initialisation of genotypes. Today we will continue working on the genotype exercise to develop a complete program for the artificial selection of the organisms.
These are the basic operations that constitute a genetic algorithm:

1) Random **initialisation** of the initial population (generation 0)
2) **Phenotype** visualisation.
3) **Selection** of best organisms.
4) **Reproduction**
5) **Mutation**

The cycle from step 2 to 5 is called "generation cycle". It must be repeated until we evolve the bets organisms we are looking for.
With last week's assignment exercise you have already written the two functions for initialising random genotypes (`random_genotype_initialisation`) and for showing the organisms' phenotypes (`showPhenotype`). In the following sections we will write the new handlers for steps 3 (selection), 4 (reproduction) and 5 (mutation). We will not apply crossover in this exercise.

2. Preparation of Animate Movie

Continue to use the Animate movie developed for last week's exercise. Note that the following ActionScript functions and instructions will only work for the previous week exercise of 4 stickman organisms with the selection of the best 2 agents, and the reproduction of 2 offspring for each selected individual (2 x 2 to generate new population of 4 agents). You have to adapt the code to make it work for your own exercise of **12 organisms**, with selection of the best 4 agents and the reproduction of 3 offspring for each selected individual (4 x 3 to generate new population of 12 agents).
We will now need to add the following components and code:

- We will need a new Layer called "dialogbox" to use for a dialog box. This will be used in the Selection procedure when advising the user that an incorrect number of organisms has been selected for reproduction. Add a new layer to the movie, and move it to the position Layer 2 so that the objects in it will be over all the other objects in layers 3 to 7 (i.e. the dialog box hides the organisms and buttons).
- Create a new movieclip symbol named "alert_mc". This symbol will consist of (1) a text field named "alert_text" that will be used to show a message to the user; (2) the instance of a new button with a label saying "OK" and its name "alertOK_btn"; (3) a graphics for the dialog box background (it is advisable to use a blank large button symbol, instead of a pure graphics, to make sure that the dialog box background hides the objects behind it);
- Put an instance of the "alert_mc" into the "dialogbox" layer. Name the instance "alertOK_mc".
- In the "actionscript" Layer add the following initialisation instructions at the top of the scripting window:

```
alertOK_mc.visible=false;// dialogbox not visible
```

- Also add the following action for the “alertOK_btn” button of the dialog box.

```
alertOK_mc.alertOK_btn.addEventListener(MouseEvent.MOUSE_UP, alertOK_off);
       function alertOK_off(event:MouseEvent) {
          alertOK_mc.visible=false;
       }
```

3. Selection

In the current genetic algorithm program with only 4 organisms, the selection procedure requires the selection of 2 parent organisms. After you select the best two agents, click on the reproduction button so that a new generation of 4 organisms is created (2x2, i.e. 2 offspring for each of the 2 selected parents).
You can choose and select the parent organisms by clicking directly on the selection buttons that we created last week: “select_0”, “select_1”, “select_2”, “select_3”.
Type the following function that is activated when you click on an organism:

```
function org_selected(event:MouseEvent) {
     var buttonname:String=event.target.name;
     var selected_organism=Number(buttonname.substr(7,1));[1]
// position 7th of the button name contains the organism’s number
     if (list_selected[selected_organism]==0) {// not selected
          number_selected++;
          list_selected[selected_organism]=1;
     } else {// already selected; now deselect it
          number_selected-=1;
          list_selected[selected_organism]=0;
     }
     if (number_selected>MAX_PARENTS) {
          alertOK_mc.visible=true;
          alertOK_mc.alert_text.text="You have selected more than 2
⇔organisms. Press OK and deselect one";
     }
}
```

This function will be executed every time you click on the button of an organism. To link it to the four organism selection buttons, add the following code in the “actionscript” Layer:

```
  select_0.addEventListener(MouseEvent.MOUSE_UP, org_selected);
  select_1.addEventListener(MouseEvent.MOUSE_UP, org_selected);
  select_2.addEventListener(MouseEvent.MOUSE_UP, org_selected);
  select_3.addEventListener(MouseEvent.MOUSE_UP, org_selected);
```

4 Reproduction

After you have selected the two patent organisms, you will have to activate the processes of reproduction and mutation by clicking on the button “reproduce_btn” created last week.

[1] Mote that the instruction `substr(7,1)` selects `1` character from position `7`. This is because the name of the button is “select_1”. If you used a different button name, or you have numbers needing 2 digits (e.g. select_10), you need to adjust the `substr` parameters.

Add the following function to the Animate movie:

```
  function replicate_new_generation():void {
     generation++;
     var parentgenotype:Array = new Array();
     // temporary list for parents' genotype
     // let's copy the genotypes of the selected parents
     var parentcounter:int=0;
     var org:int;
     var parentnum:int;
     var childnum:int;
     var newchildnum:int;
     for (org=0; org<MAX_ORGANISMS; org++) {
          if (list_selected[org]==1) {
               parentgenotype[parentcounter]=genotype[org].slice();
               // copy organism_num's genes into parents' list
               list_selected[org]=0;
               parentcounter++;
          }
     }
// let's copy parents' genes into the new children's genotype list
     for (parentnum=0; parentnum<MAX_PARENTS; parentnum++) {
          for (childnum=0; childnum<MAX_CHILDREN; childnum++) {
               newchildnum = (parentnum*MAX_CHILDREN)+childnum;
          genotype[newchildnum]=parentgenotype[parentnum].slice();
               // make clone of parent's genes
          }
     }
     number_selected=0;
  }
```

5. Mutation

Now type the function for the mutation of genes. Remember that this function only works for the first genes, but you'll have to adapt it so that all 5 genes are mutated.

```
  function mutate_new_generation():void {
     var neworgnum:int;
     var mutatedvalue:Number;
     for (neworgnum=1; neworgnum<MAX_ORGANISMS; neworgnum++) {
          // 1st gene
     mutatedvalue=genotype[neworgnum][0]+randomRangeInteger(-1,1);
          // it will add +/- 1 to gene1 values
          if ((mutatedvalue>=1) && (mutatedvalue<=3)) {
               // it is within range
               genotype[neworgnum][0]=mutatedvalue;
          }
          // REPEAT THIS FOR THE REMAINING GENES
     }
  }
```

Now attach a call to both the functions `replicate_new_generation` and `mutate_new_generation` to the button “replicate_btn”. This will also include a call to the `show_phenotype` function that visualises the phenotypes of the new generation.

```
replication_btn.addEventListener(MouseEvent.MOUSE_UP,
replicate);

function replicate(event:MouseEvent) {
   if (number_selected==MAX_PARENTS) {
        // proceeds with replication and mutation
        replicate_new_generation();
        mutate_new_generation();
        show_phenotype();
        info_text.text="Replicated & Mutated. Generation
⇔"+String(generation);
   } else {// wrong number of selected organisms - use dialog box
        alertOK_mc.visible=true;
        if (number_selected<MAX_PARENTS) {
             alertOK_mc.alert_text.text="You have selected only
⇔"+number_selected+" organism. Press OK and select more up to a
⇔maximum of 2";
        } else {
             alertOK_mc.alert_text.text="You have selected
⇔"+number_selected+" organisms. Press OK and deselect some down
⇔to a maximum of 2";
        }
   }
}
```

This program can now be used to test of the whole genetic algorithm and evolve your population of ideal organisms.

6. Crossover

Write your own code for a new function that adds a crossover method to the selection. The code will control the paring and gene exchange for the 4 selected agents. You can decide if you want to use a single-point crossover, or other more complex methods.

Allow the user to activate and deactivate the crossover function.

ASSESSMENT EXERCISE (Exercise 1 - Part 2)

The program you developed today must be saved as part of the final coursework. This Animate movie should successfully allow the artificial selection of organisms through the process of selection, reproduction, and mutation. Remember not to use stickman-like organisms, as you are require to design new populations of agents in a creative way. Also modify the exercises so that the user receives feedback (e.g. text on a text filed, or speech comments) during the evolution process, such as knowing how many and which organisms are being selected, how many generations have occurred etc.
Note that although in this example we only select 2 organisms out of 4, in your own exercise with a total of 12 organisms you will have to adapt the code to allow the selection of 4 organisms and the reproduction of 3 copies each per selected organism.

Marking criteria

This second part of the Assignment exercise is worth it an additional 15% of the coursework mark.
To mark this exercise, the following questions will be used:

a) Does the program allow the artificial selection of organisms? [4 marks]
b) Is the reproduction function working correctly? [1 marks]
c) Is the mutation function working correctly? [4 marks]
d) Is the crossover function correctly working? [3 marks]
e) Does the interface provide enough information on the current status of the genetic algorithm? [3 marks]