Name: Danny Ou

ICS4UI: 2D Simulation of Neural Network Evolution

*Documentation*

**What was built?**

My aim for ICS4UI is to create a very generic but an unique program, which consists of many complex and complicated algorithm to simulate real life. Initially, my target was to build a biological modelling program that draws and shows all of the interaction between our human systems. However, due to the complexity of math and 3D programming experience, it was very hard to build and code; but in researching for a topic like this, it inspired me to create a simple but unique 2D genetics simulator to symbolize the process of evolution. My final project consisted of a three main components, which included:

* The Evolution Process - Ultimately, it was divided into three section of evolution to symbolize the life and death of individuals as well as the importance of genetics
* The Neural Network - By using genetics and traits, neural network allowed me to create and apply the brain-like system into many individuals.
* The 2D Vectors - Allowed me to prove evolution, as well as being able to carry out small actions made in the neural network as well as see a greater impact of evolution through different generation.

What I want my program to demonstrate is the uniqueness of evolution throughout time to see how lifeforms change by applying computer science knowledge. The purpose of the genetic simulator is to provide a change in the interaction for animals (or any lifethings). However, in the bigger picture, what I want my reader to understand about my skills is my ability to combine and as well as to intertwine different and complex subject of sciences into creating a finalized idea to prove evolution.

**How it was built?**

Creating and planning out how the structure of the program would layout would be the hardest part of this project. At the beginning, I stumbled upon on the diversity of genetics and how genetics should be portrayed.

I leant and researched about DNA and the process of carrying them into RNA and further more to MRNA. However, being able to carry out and analyzing 2 million DNA base pairs was tough and as well, it needed excessive amount of processing power. Therefore, in order to make this simple but keep its main component of evolution, I used binary bits to pretend as DNA and each 1’s and 0’s combinations makes up for a simple allele.

Later on, in Sprint 1, I started building the actual genetic algorithm that showed how evolution process would start and end. However, the algorithm that I used to build was very limited in terms of its fitness calculation. As well, the natural selection process did not include many of the important factors on how the certain individuals were chosen such as using evenly shared of distribution. During the research process, I would like to thank Ms. Trainor for helping me understand and clarify confusion regarding evolution. With the help of Ms. Trainor and *HandBook of Genetic Algorithms,* the process of evolution was much clearer and it had provided me a narrow but unique approach to this project. In the end of sprint 1, I programmed the binary decoder to extract its gene to correlates with the fitness.

In Sprint 2, I learnt about the characteristics of what makes a simulator unique and how does different simulator differs in terms of their properties. During the building process, I followed a lot of these characteristics to make my program followable:

* Static/**Dynamics**: To allow for realistic changes and unpredictable outcomes, more interactive.
* Deterministic/**Stochastic**/Chaotic: I want the individual to change according to the DNA bits within their genome, therefore, its behavior will be very unpredictable
* **Discrete**/Continuous: Evolution allows variables to change at a sudden rate, therefore, no gradient of change. When the DNA exchange, individual will have different characteristics (alleles)
* Aggregate/**Individual**: Allow to see the change of individuals as well as their growth over time.

Continuing with Sprint 2, I also leant neural network and try to incorporate the neural network with the genetic algorithm into my program. However, one of the greatest challenge that really slows down my time and schedule was the idea to make neural network compatible with the genetics algorithm. While doing research online, there was not as much resource provided that connected the two algorithms together. As well, creating a 100% genetics algorithms that can connect well with neural network would be impossible due to the wide variety of unique in the combination of alleles as well as different randomization during the Crossing Over process. However, a way that I tried to approach this is to implement gray code for small change in between the numbers. This made the program more flexible in terms of numbers but as well as avoid any dramatics and huge changes in between that may disrupt the process of evolution. As well by converting binary bits to gray code, I was able to encode the weights to fit and match up certain genes of alleles in the individuals.

Finally, my 3rd Sprint is mostly creating the Sprite Kit program itself, incorporating the physics content of the sprite Kit into my program, and implementing the neural network with player’s AI system.

Learning Sprite Kit is very different than Unity, which is a program that I have experience before. SpriteKit, I would say is more of a direct approach to programming. As I didn't use any gameScene item and UIKit items, most of my image and nodes are generated from programming codes, therefore, the learning process for SpriteKit takes a while for me to learn as well as to play around since this is my first time.

Learning the Physics and 2D vectors in SpriteKit would be the second hardest part within this project. Since I was in grade 11, my math was not experience enough for me to learn Calculus and 2D vectors because of this, I had to learn from tutorials online from Raywenderlich and from Mrs. Suresh and Mr. Fare to approve some of my logics and math regarding with vectors and radar detection. However, in the end, I was able to create a detect-on-hitbox collision, which collects and sends signal whenever an individual made a contact with the object nearby. During the contact of the object, the system detects what object was contacted as well as provide the interaction needed to happen.

Along with creating Physics, one of the main task that I had to do was to connect the network network algorithm to the player’s detection field of view. During the building of this function, I used a contact-on-detection function that records down the objects collided with the field of view body of each individual player. In terms of processing, it locates the object and its position and carry out the Field of View Check. If it succeed, it will be passed on to one of the 12 sectors that is divided equally to 15 degs. Each sector consists of an individual's neuron section. The implementation of this process allow the neural network to be specialized as well as to receive input from the player.

**Reflection:**

* What worked?

At the end, I was really proud of my idea and the way the program was created after through the 4 months process of studying but as well as applying and integrating those concepts into programming and computer science in general. The creation of this idea and the connection between different algorithms to increase the diversity of evolution was well planned out and the project’s building process required tons of studying to ensure that I was following the right path. My genetics algorithms worked well to mimic the process of evolution and was done with precision. As well, my creation and experience with SpriteKit was pretty amazing and beneficial as it develops my skills of game programming in terms of learning new techniques to build scene. As far as for the physics and math part of the SpriteKit, I believe that I could improve on engineering a better path behavior however, learning 2D vectors and applying it to graphics programming according to my own understanding is something that I am really proud of that really somewhat work to my liking.

* What didn’t work out so well?

The idea and the creation of the program was well-done; however, I do feel that the genetics algorithms and the neural network can be much improved. At the end, the genetics evolution of individuals was much slower than what I have expected. As well, the neural network training process were varied in skills in many stages, thus the learning curve of the individual was hard to implement successfully (during the beginning, neural network mostly contain nan and inf, which is undefined and infinity for unmeasurable values). This suggested that my integration as well was the learning curve in the training process was too high. I wished that I would have more time to integrate the two complex algorithms and make them more compatible with each other. Also, in the previous question, I was pretty happy that the 2D vectors and the collision worked however; I do think that I could improve on the integration and may use different methods of motion to coordinate the movement of the individuals in my program more better.

Another systematic problem, is when the game hit to ~ 65th Generation, my simulator froze, in which suggested that I need to take care of optimization and memory capacity usage.

* How you would change for the next time?
  + Raycasting to use for the field of view detection was a much better idea than using contact-on-hit method and the reason is that the contact-on-hit method does not know when to keep track or remove the contacted object. So therefore, the program will take up more memory capacity as well as more object to keep track, which can be further optimized to make it better.
  + Neural network implementation was done correctly in terms of creating a normal network however, for genetics neural network, I am missing major components which I need to revisit if I have available time. As a result, I need to do more research and try to lower the learning curve rate in between each generation.
  + Memory Consumption and Usage were not taken much consideration and I would hope to visit and optimize the program to make it more user-friendly and avoid crashes in between. For next time, I would look on how to optimize the collision process as well as the neural network to once again limit the learning process curve and provide more accurate and less memory heavy implementation.
  + User Friendly Code - Ultimately, my code format and the coding order was done decent, however; for anyone who is new looking at it, they will have a harder time to keep track and harder to follow along with, which is a fault in my programming style / technique and can be further improved for next time to make the code more understandable.

Reference:

<https://www.raywenderlich.com/155780/gameplaykit-tutorial-entity-component-system-agents-goals-behaviors-2>

<https://www.raywenderlich.com/145318/spritekit-swift-3-tutorial-beginners>

<https://www.raywenderlich.com/90520/trigonometry-games-sprite-kit-swift-tutorial-part-1>

https://www.raywenderlich.com/90521/trigonometry-games-sprite-kit-swift-tutorial-part-2

<https://www.raywenderlich.com/57368/trigonometry-game-programming-sprite-kit-version-part-1>

*Sprint 2:*

[*http://homepages.inf.ed.ac.uk/pkoehn/publications/gann94.pdf*](http://homepages.inf.ed.ac.uk/pkoehn/publications/gann94.pdf)

[*https://en.wikipedia.org/wiki/Cellular\_automaton*](https://en.wikipedia.org/wiki/Cellular_automaton)

*Neural Network*

[*https://www.youtube.com/watch?v=bxe2T-V8XRs*](https://www.youtube.com/watch?v=bxe2T-V8XRs)

Sprint 1:

*HandBook of Genetic Algorithms, Lawrence Davies*

*Special Thanks to Mrs.Suresh, Mr.Fare, and Ms. Trainor!*