**Extended Project Qualification**

Control of an Inverted Pendulum Environment Through Various Agents (Evaluated Against Internet-Based Standards)

Word count: 3166

Candidate name: Kusalin Thanyakulsajja

Candidate Number: 4100

Centre Number: 75747

June 2018 Examination Series

**Abstract**

An inverted pendulum is simulated in a virtual environment and three different control strategies were used in an attempt to solve the inverted pendulum control problem. The three different control strategies were hard coding, genetic algorithm and artificial neural network. The effectiveness of the control strategies were first evaluated by observation, mostly using sight and logic to deem its extent of control over the environment. If it was deemed to be able to control the environment in real time, it was evaluated against an internet-based standard. Only the artificial neural network control strategy was evaluated to be successful at practically controlling the environment.

Brianno Coller’s “Classic Inverted Pendulum - Equations of Motion” and Richard S. Sutton’s gym cartpole.py were some of the resources which helped me create my virtual environment. Charles W. Anderson’s “Learning To Control An Inverted Pendulum Using Neural Networks” and Sentdex’s “Intro - Training a neural network to play a game with TensorFlow and Open AI” were some of the resources which helped me create the artificial neural network control system. These resources are explained and evaluated in the Literature Review section.

Abstract word count: 184

**Contents**

[**Introduction**](#_lmb1dcbgqgv) **4**

[**Literature Review**](#_dk5tbymzngfa) **5**

[Brianno Coller (Youtube account), “Classic Inverted Pendulum - Equations of Motion”](#_upd8ak9m0mmi) 5

[Charles W. Anderson, “Learning To Control An Inverted Pendulum Using Neural Networks”](#_5v5nm0xbqoms) 5

[Richard S. Sutton, gym cartpole.py](#_l2gj3lhtwx7f) 6

[Sentdex (Youtube account), “Intro - Training a neural network to play a game with TensorFlow and Open AI”](#_r97ns7jcg7qp) 6

[**Process and Execution**](#_w9ahii3k9kcl) **7**

[Hard Coded Agent](#_hlghbdex3t1h) 7

[Genetic Algorithm Agent](#_wd2e09sjco8w) 8

[Artificial Neural Network Agent](#_cy2qjv5bijoq) 9

[Analysis and Application of Primary Sources](#_o9hjdtig266g) 11

[**Conclusion**](#_3jba198e9ljc) **13**

[**Bibliography**](#_nqwog13qq72h) **14**

[**Appendix**](#_d4k4l2y0h0p6) **15**

[Appendix A](#_ezts11jj41v3) 15

[Appendix B](#_vvjn5kt4u366) 15

[Appendix C](#_rarkditvtztn) 15

[Appendix D](#_kfgq42cp2nk2) 16

[Appendix E](#_2v3m71wj1j64) 16

[Appendix F](#_bvp16mqk8y10) 16

[Appendix G](#_8cn5k0tuoox5) 18

[Appendix H](#_3bmeebl3ufex) 20

[Appendix I](#_lp1yxqvpl9i7) 22

# **Introduction**

Two key terminologies used in this report are “environment” and “agent”. When the word “environment” is used, it is referring to the code in the program which is responsible for the simulation of the imaginary inverted pendulum being controlled. This includes calculations of motion and calculations for visuals etc.

The structure of the report is influenced by the structure of the project/software development. The project is divided into three main “phases”: Hard Code, Genetic Algorithm and Artificial Neural Network. During the respective phases, each respective agent was developed (Hard Code agent developed in the Hard Code phase etc.), and the environment is improved throughout each phase. I felt that this kind of structure would allow me to present my findings in a logical and clear manner.

The aim of the project was not just to efficiently or effectively control the environment, but also to allow me to explore physical computation and gain experience by coding in this field. The motivation of the project lies behind my interest in computer science and control systems, thus the skills, knowledge and understanding gained from the project is actually what is valuable to me. This meant doing things such as experimenting with incorrect code or writing my own environment, which is unnecessary to control the environment but is very beneficial to my education.

# **Literature Review**

## **Brianno Coller (Youtube account), “Classic Inverted Pendulum - Equations of Motion”[[1]](#footnote-0)**

This is a video I came across during the development of the genetic algorithm agent. The contents of the video explained the physics and mathematical equations behind the motion of both the cart and the pendulum in an inverted pendulum environment, which is directly related to the production of my virtual environment. Though the resource was from an unreliable source, I tested its reliability by programming the virtual environment using his mathematical equations and observed a more realistic and accurate representation of an inverted pendulum than what I had previously. The content of this video linked to the topic of differentiation, which I had learnt in AS mathematics class.

## **Charles W. Anderson, “Learning To Control An Inverted Pendulum Using Neural Networks”[[2]](#footnote-1)**

This is a report I came across before the development of the neural network agent. The contents covered the use and result of two different types of agents used to control a virtual inverted pendulum environment: two single-layer networks and two-layer networks. The paper had stated that “As shown in later sections, the desired evaluation function for the inverted pendulum task is nonlinear; a single-layer neural network cannot form this map” which helped me decide on the use of a deep neural network for my program.

I expected this source to be reliable as “This work was partially supported by the Air Force Office of Scientific Research and the Avionics Laboratory”(Charles W. Anderson)[[3]](#footnote-2) and was published on June 1988 in the American Control Conference[[4]](#footnote-3).

The content of this report is linked to the topic of artificial neural network, which I was able to understand after some research[[5]](#footnote-4).

## 

## **Richard S. Sutton, gym cartpole.py[[6]](#footnote-5)**

Richard S. Sutton, Currently he is professor of Computer Science and iCORE chair at the University of Alberta, thus I assume a reliable source. This is a code repository which shows the code for gym’s cartpole environment. I have utilised the code for the production of my own virtual environment (using pygame). I have observed the motion of the virtual pendulum when running my code, thus testing the source’s reliability. The logic behind the code is linked to the topic of Object-Oriented Programming, which I had first learnt from an internet resource[[7]](#footnote-6).

## **Sentdex (Youtube account), “Intro - Training a neural network to play a game with TensorFlow and Open AI”[[8]](#footnote-7)**

This is a video I came across during the development of the neural network agent. The video is the start of a series which explains and shows the use of a deep neural network to control a virtual inverted pendulum environment. The reliability of this source was checked by running the code in my own computer and observing the behaviour of the virtual environment. Since the virtual environment used was from a reputable source (Richard S.Sutton/Open AI gym), this is a viable method of testing this source. Much of my code for the neural network agent is copied from him, though I have shown my understanding of the function of the code further in my report. The content of the series of videos is linked to the topic of artificial neural networks, which I was able to understand after some research[[9]](#footnote-8)

# **Process and Execution**

## **Hard Coded Agent**

Hard Coded is a term used for a program which behaves in a way that cannot change without modifying the program (the other two agents will change its behavior depending on random chance or training data). The control method first developed is based on the simple thought of “if the pendulum is to the right of the cart, push the cart to the right” and vice versa (as well as making the magnitude of the force proportional to how far away the pendulum is to the cart). This was done by equating the force exerted on the cart to a constant factor (hence called “kfact”) multiplied by the pendulum’s horizontal distance away from the cart. This creates an effect resembling simple harmonic motion (a type of repetitive motion often seen in regular pendulums), but is actually not. [see Appendix D]

The control method was improved shortly after. This time, the constant factor mentioned decreases constantly over time in an attempt to “slow down” the motion similar to simple-harmonic motion created by the first method. Essentially trying to create motion similar to damped simple harmonic motion (motion seen in regular pendulums). This method has failed to produce the results I wanted, instead the amplitude of the pendulum’s motion keep increasing until the pendulum hits the ground. This variation is referred to as “damp kfact” by me. [see Appendix E]

After this attempt, a new control method was introduced; this variation is referred to as “tinker” by me. The new method works with the idea to tinker the acceleration of the cart in a tricky way in order to get the pendulum to be upright. This method was inspired by the failure of the previous two methods. A core strategy of this method is to not let the pendulum fall over to both sides and create oscillation (repetitive motion): if it falls to the right, then we will accelerate the cart just enough to keep theta constant, then add a little acceleration/force for a single time-step in order to slowly turn the pendulum upright, then counteract that added force with an equal and opposite force when the pendulum is upright to keep theta at zero (or very close). While this method showed some success, the cart moves indefinitely as its velocity is non-zero when zero theta is achieved. The smoothness of the pendulum’s movement also felt unrealistic, as the pendulum hardly moved horizontally as the cart slid under it. This made me realise that the calculations for the virtual environment I derived was inaccurate. [see Appendix F]

Though the environment’s motion calculations derived by me was later found to be inaccurate and the final control method would not be practical in a real-world situation, there were many successes about the “Hard Code” stage of this project. Firstly, deriving the equations myself and experimenting with the visuals helped me understand physical computation and it’s discrete “time-step” nature better. Secondly, the were many instances of using primary sources (observations from older control methods) to improve upon the algorithm. This is evidence that my abilities as a programmer is improving. [see Appendix A to F]

## **Genetic Algorithm Agent**

This agent was developed due the Hard Code agent’s inability to control both the velocity of the cart and angle of the pendulum at the same time. The environment was also redeveloped to better simulate a real pendulum (frictionless and in vacuum).

Due to improvements in programming skills and internet-based researches[[10]](#footnote-9) [[11]](#footnote-10), I was able to create a cleaner program using function definitions and Object-Oriented programming, as well as implementing an original genetic algorithm program (inspired by carykh’s videos[[12]](#footnote-11)). The control system comprises of “Force” objects which contains an array of numbers (“forcecode”) representing the force on the cart for each time-step. The objects are given a score based on how well its forcecode keeps the pendulum stable (controls the pendulum), this is done by the function “give\_score”. The “mutate” function will slightly alter a forcecode. The function “survive” simulates survival of the fittest: two “children” objects are created by a single “parent” forcecode, both of which has slightly “mutated” forcecodes, then the forcecode of the one with the lowest score (better at controlling the environment) is “passed on” to the next generation to be used as the “parent” forcecode. After many generations, we can hope that the forcecode of the last object can effectively control the environment. [see Appendix G]

Table of results (time taken to find solution, in minutes):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4.23 | 4.35 | 4.53 | 4.44 | 4.38 | 4.52 | 4.47 | 4.79 | 4.64 | 4.56 |

Test result average: 4.49 minutes (3 significant figures)

The developments in the environment is mostly based on Brianno Coller’s video[[13]](#footnote-12), which involved having to learn the basics of general relativity and circular motion. This new environment better approximates an ideal inverted pendulum as the natural motion of the cart and ball both affect each other in the correct manner, though the effect of a force on the cart is still yet to be accurately modelled. [see Appendix G]

The biggest problem of the genetic algorithm is its inherent incompatibility with the inverted pendulum control objective due to its nature. It simply takes too much time to arrive at the solution, as many generations has to be calculated in order to obtain a reasonable solution. It does not help that the solution to every single time-step must be calculated right from the start. Imagine the pendulum getting tipped over, then over four minutes later the cart makes a series of movements: the random nature of this algorithm is too time-costly for it to be a feasible solution to the control problem. Thus the imperfect environment is not a cause for concern over the performance of this algorithm.

## **Artificial Neural Network Agent**

(This agent can also be referred to as “Neural Network Agent”)

After the genetic algorithm phase, I needed to find a new control method which can calculate the solution in realtime (a more practical solution was needed). But before I can create a program the new agent, a new environment needed to be used to ensure the agent has the correct environment to control. [see Appendix H]

I have first utilised an open-sourced environment from the internet[[14]](#footnote-13), then wrote my own complete environment file (calculations based on the internet resource). The use of this secondary source helped me develop my own environment file in many ways. Firstly, the order of calculations in this source had taught me the correct logic behind physical computation. The angle and displacement was to be updated before the angular velocity and cart velocity, which seems counterintuitive. Though this makes sense when considering discrete time-steps, as the velocity of the last time-step (not the velocity of the current time-step) should affect the displacement of the current timestep; the velocity of the current time-step simply has no “time” to affect the current displacement (and thus will go on to affect the displacement of the nest time-step). This sort of logic was made understandable to me partly due to the earlier experience of trying to program my own environment (from Hard Code phase), and solves the problem with the environment mentioned earlier (from Genetic Algorithm phase). Secondly, the different functions (or “Methods”) an environment “object” needs to have was made clear to me: step, reset etc. This allowed me to code more efficiently and effectively: more robust code in less time. [see Appendix H]

The control method used was based mostly on Sentdex’s tutorials[[15]](#footnote-14) , and much of the code is identical to his work. Though the understanding of this method can be attributed to many other secondary resources such as Brandon Rohrer[[16]](#footnote-15), which was extremely useful as the learning curve to Artificial Neural Networks is steep. [see Appendix H]

The control method works by “mapping” every input to every output through a series of multipliers or “weights”. The complexity of the algorithm comes from its layered structure of nodes, hence artificial neural networks. Essentially, the neural network is a matrix of numbers which are multiplied or added to each input in an orderly fashion. This matrix of numbers are changed via a method called reinforcement learning, which uses differentiation to lower the “error” of the output (error means the difference between the input and expected output). Reinforcement learning requires a training dataset with known inputs and expected outputs. After the neural network has been trained, it can be used to give outputs to a new inputs which we do not know the expected output of; this is useful for solving the inverted pendulum problem.

Essentially, the function “initial\_population” creates random inputs and feeds it to the environment, then creates the training data out of the input-output pairs of the games where the pendulum happened to survive for a long enough period of time. This training data is the used with reinforcement learning to train a randomly created artificial neural network (random matrix of numbers), resulting in a trained neural network (less random matrix of numbers). After that, the trained neural network is tested against random scenarios in the environment to test how long it can survive (be within 15 degrees of zero theta). [see Appendix H]

Training data’s average (in seconds): 3.34

Testing results for 10 iterations (in seconds):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5.36 | 5.38 | 6.84 | 7.20 | 4.94 | 5.02 | 5.98 | 5.08 | 6.34 | 5.68 |

Testing result average (in seconds): 5.78 (3 significant figures)

I first had doubts on copying other people’s code, but as I became a better/more experienced programmer I realised the point is whether I understand the logic behind the code or not and that copying code is a big part of learning as a programmer[[17]](#footnote-16).

The success of this particular program can be compared to an internet-based standard from Open AI’s CartPole-V0 standard[[18]](#footnote-17), which defines "solving" as getting average reward of 195.0 (3.9 seconds) over 100 consecutive trials. An average of 5.78 seconds is way beyond the 3.9 seconds hurdle, but it was done only in 10 consecutive trials (which means it is more subjected to random chance). Though after thinking about the nature of the algorithm, I decided that it was successful in controlling the environment.

The trained neural networks are a matrix of numbers. This means that this method can be used to first train a neural network, then just upload the successfully trained neural network into the smaller CPU of a physical cart. It would have no problems running real-time as the CPU is simply using the pre-calculated matrix to perform simple calculations on its inputs, and a more powerful CPU can be attached to a physical cart in order to generate the training data. This solves the problems found when testing the genetic algorithm method, as the training of the neural network and the usage of it is separate. Moreover, a successfully trained neural network can be copied to any number of cart devices without having to waste time training it again. Thus, I find the artificial neural network to be a successful controller of the inverted pendulum environment.

## **Analysis and Application of Primary Sources**

|  |  |  |  |
| --- | --- | --- | --- |
| **Primary Source Description** | **Development Phase** | **Changes made due to it** | **Reasons for changes** |
| Observation of motion similar to simple-harmonic motion | Hard Code, 1.0 | Attempt to damp the system | This kind of motion would take up theoretically infinite energy/electricity to keep up, thus not practical |
| Observation of continual increase in the amplitude of the pendulum | Hard Code, 1.0 | Development of “Hard Code, 2.0” which uses a method I refer to as “tinker” | Unable to keep pendulum upright, thus unable to control environment |
| Observation of unrealistic smoothness in the pendulum’s control | Hard Code, 2.0 | Development of a new virtual environment in the genetic algorithm phase | A more accurate virtual environment is desired in order to better evaluate the control system’s effectiveness |
| Observation of a constant non-zero velocity when zero theta is achieved, leading to the cart’s continual movement away from its starting point | Hard Code, 2.0 | Development of the genetic algorithm agent, in order to try to achieve both zero theta and zero velocity at the same time | This is not viable in a real system as the non-zero velocity would affect the theta due to frictional forces or would move the cart indefinitely in an ideal system. |
| Test results of Genetic Algorithm Agent run-time | Genetic Algorithm | Development of Artificial Neural Network Agent | This would not be a viable solution to any system running real-time, as the solution would be achieved too late. The development of the artificial neural network agent aims to solve this issue. |
| Test results of Artificial Neural Network Agent | Artificial Neural Network | Completion of Project | As the tests results show a success when compared to an internet-based standard, I have statistical evidence of the agents success. |

# 

# **Conclusion**

Hard coding was not found to be a viable control strategy after observing its behavior in its development and using judging it to be impractical to control both the pendulum angle and cart velocity at the same time. Genetic algorithm was not found to be a viable control strategy due to it taking too long to arrive at a solution on the first time-step (average of 4.49 minutes from testing data). The flaw in genetic algorithm is judged to be due to its nature against the inverted pendulum problem, as it requires the computer to find the solution for multiple time-steps ahead within the time limit of just one time-step. The judgement led to the development of the neural network control system.

Artificial neural network combined with reinforcement learning is concluded to be a viable control strategy of the inverted pendulum control problem. Firstly, it has averaged 5.78 seconds in run-time compared to the internet-based standard of 3.9 seconds. Secondly, I have observed its efficiency by observing the realistic motion it creates in the virtual environment and judged its computational time for the solution to be less than one time-step (it is fast enough to run realtime). The judgement was aided by the understanding of the matrix-based nature of the neural network. Lastly, the understanding of the matrix-based nature of the neural network allowed me to recognize its ability to reduce memory space needed and its ability to be copied onto any number of devices when applied to a real-life situation.

# 

# **Bibliography**

Brandon Rorher, “How Deep Neural Networks Work”

Available at: <https://www.youtube.com/watch?v=ILsA4nyG7I0&pbjreload=10>

Accessed [20/10/2017]

Brianno Coller (Youtube account), “Classic Inverted Pendulum - Equations of Motion”

[online] Available at: <https://www.youtube.com/watch?v=5qJY-ZaKSic&t=190s> Accessed [1/5/2017]

Carykh (Youtube account), “Evolution Simulator (Part 1/4)”

[online] Available at: <https://www.youtube.com/watch?v=GOFws_hhZs8>

Accessed [10/5/2017]

Charles W. Anderson, “Learning To Control An Inverted Pendulum Using Neural Networks”

[online] Available at: <http://www.cs.colostate.edu/~anderson/newsite/publications/anderson-pole-1989.pdf>

Accessed [17/10/2017]

Corey Schafer, “Python OOP Tutorial 1: Classes and Instances”

[online] Available at: <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&list=PL-osiE80TeTsqhIuOqKhwlXsIBIdSeYt>

Accessed [15/4/2017]

John Sonmez, “I Can't Build an Application With Out Copy Pasting!”

[online] Available at: <https://www.youtube.com/watch?v=mXJTrH69JM0>

Accessed [24/10/2017]

Richard S. Sutton, gym cartpole.py

[online] Available at: <https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py>

Accessed [17/10/2017]

Sentdex (Youtube account), “Intro - Training a neural network to play a game with TensorFlow and Open AI”

[online] Available at: <https://www.youtube.com/watch?v=3zeg7H6cAJw&t=222s>

Accessed [17/10/2017]

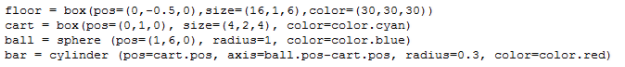
Sentdex (Youtube account), “Python 3 Programming Tutorial - Functions”

[online] Available at: <https://www.youtube.com/watch?v=owglNL1KQf0&index=11&list=PLQVvvaa0QuDe8XSftW-RAxdo6OmaeL85M>

Accessed [15/4/2017]

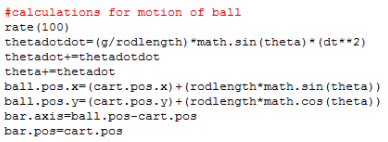
# **Appendix**

## **Appendix A**



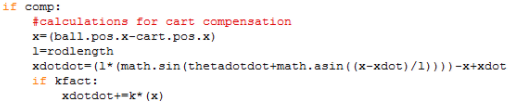
Environment initialisation.

## **Appendix B**



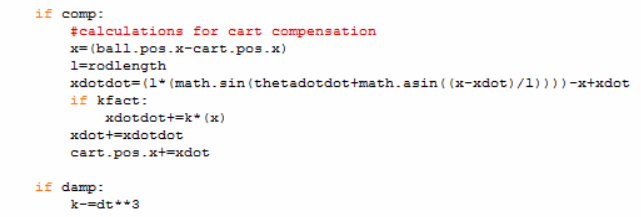
Motion Calculation of the ball/pendulum.

## **Appendix C**



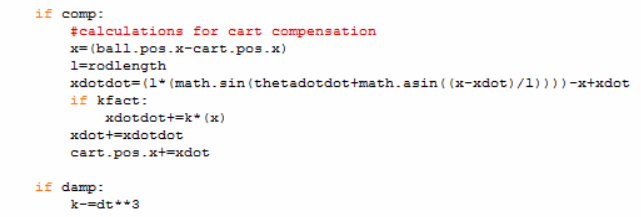
Motion Calculation of the cart.

## **Appendix D**



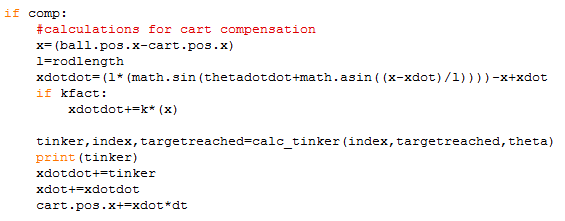
“kfact” control method (Hard Code, version 1.0).

## **Appendix E**

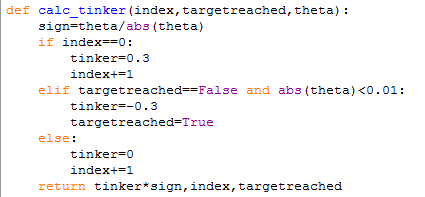


“Damp kfact” control method (version 1.0).

## **Appendix F**

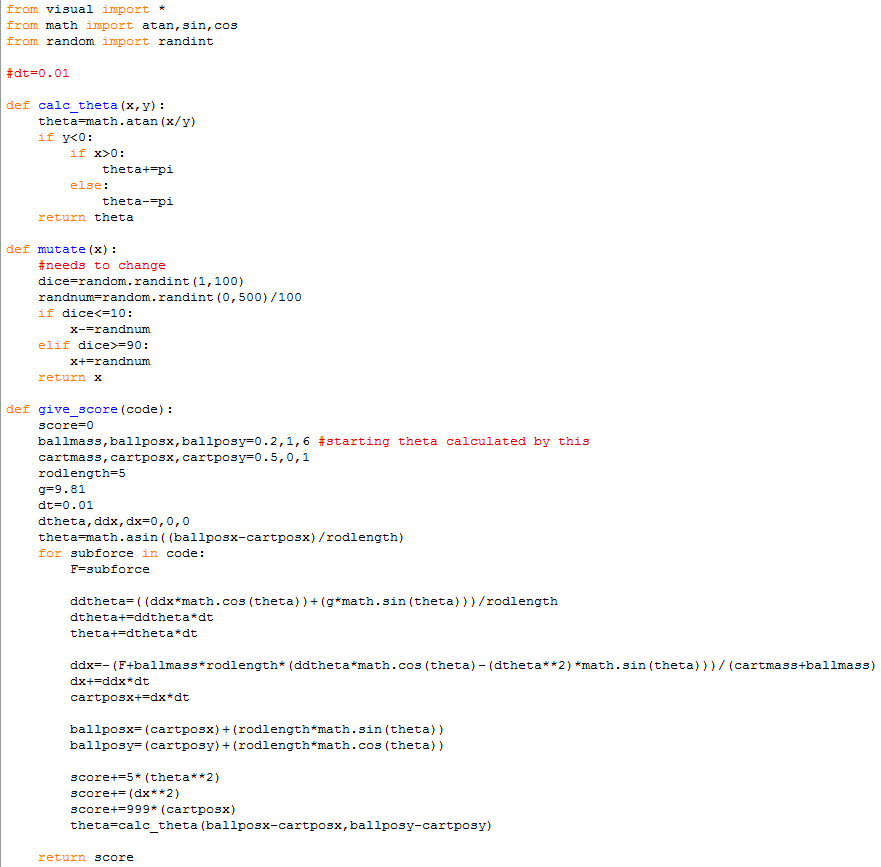


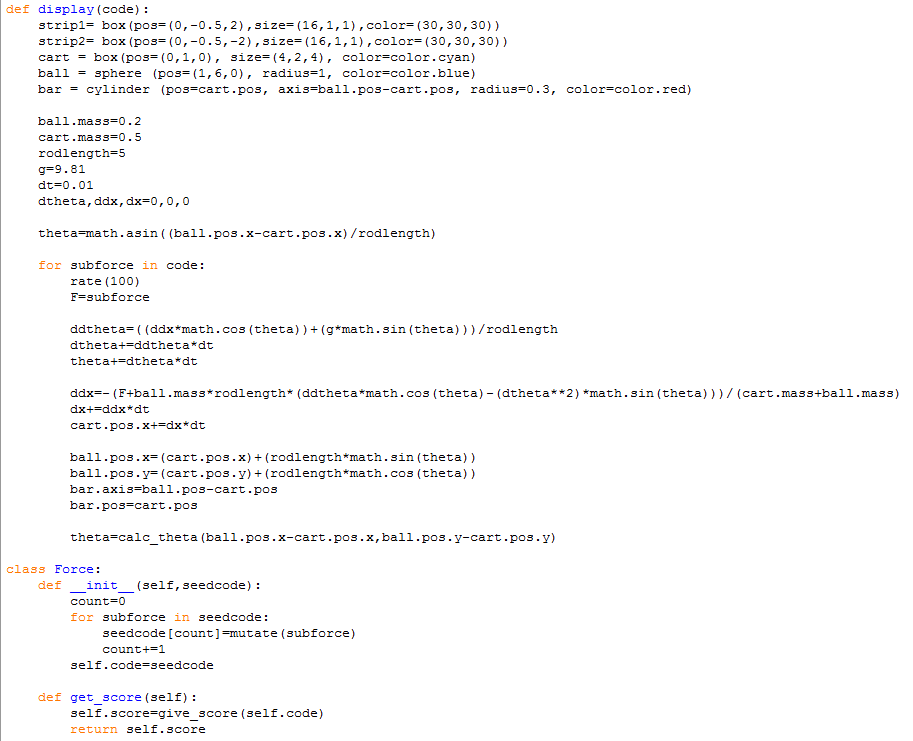
(above) “tinker” control method.



(above) definition of the “calc\_tinker” function.

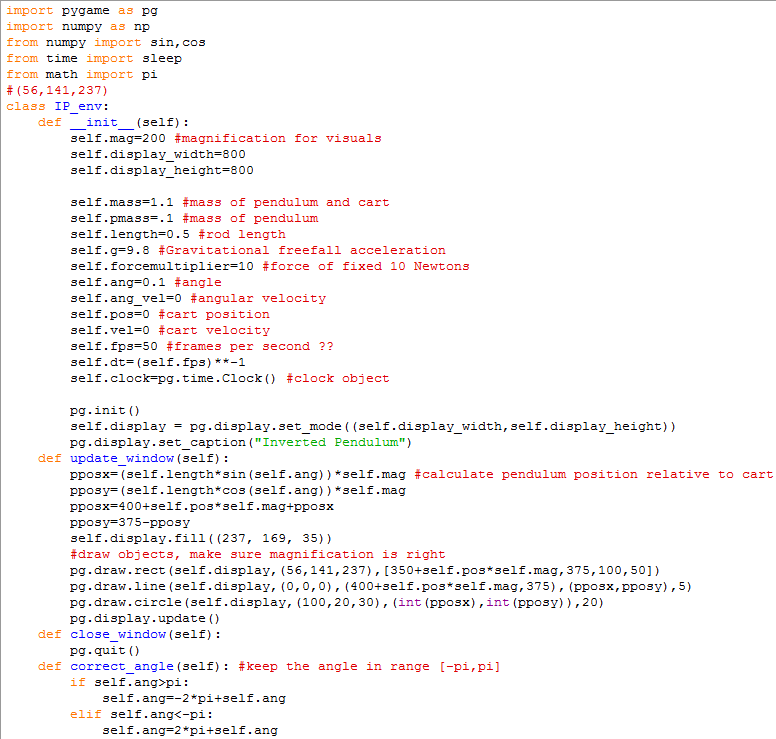
## **Appendix G**

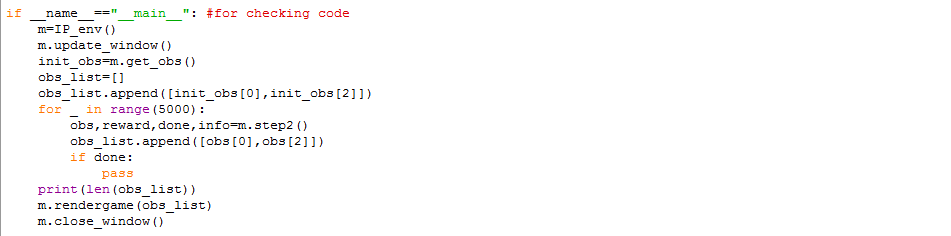
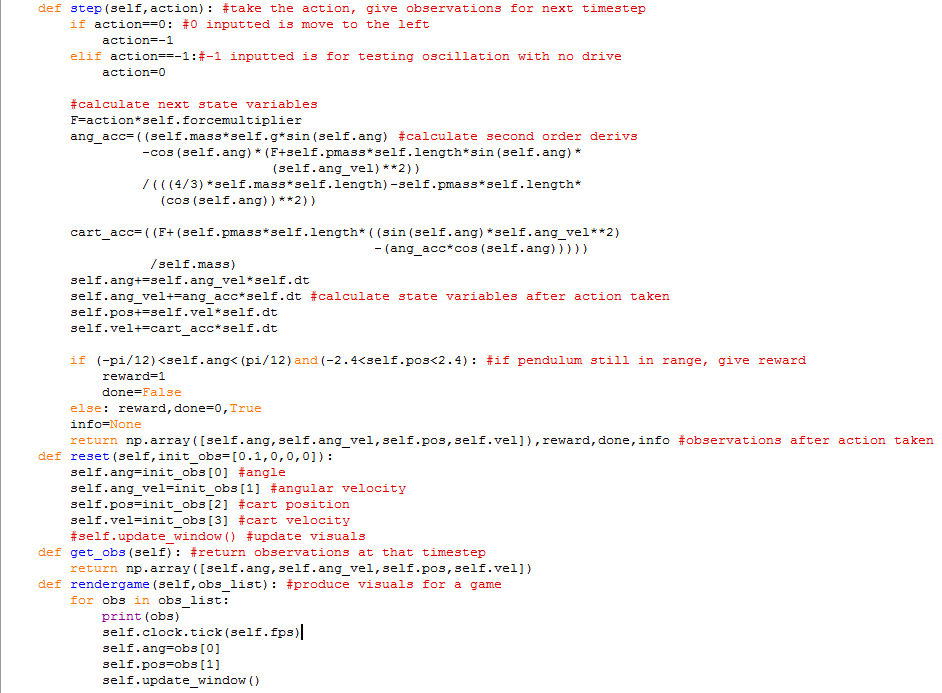




Three snapshots which shows the whole genetic algorithm program.

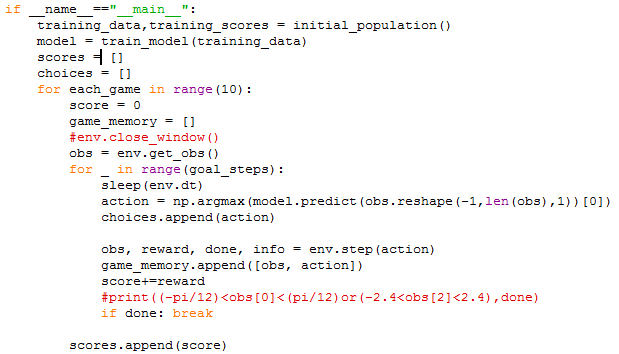
## **Appendix H**





Three snapshots which shows the whole environment program.

## **Appendix I**



Main program which integrates all functions (such as “initial\_population” and “train\_model”) in order to complete the neural network agent.

# 

# 

1. <https://www.youtube.com/watch?v=5qJY-ZaKSic&t=190s> [1/5/2017] [↑](#footnote-ref-0)
2. <http://www.cs.colostate.edu/~anderson/newsite/publications/anderson-pole-1989.pdf> [17/10/2017] [↑](#footnote-ref-1)
3. <http://www.cs.colostate.edu/~anderson/newsite/publications/anderson-pole-1989.pdf> [17/10/2017] [↑](#footnote-ref-2)
4. <http://ieeexplore.ieee.org/document/4790107/> [17/10/2017] [↑](#footnote-ref-3)
5. <https://www.youtube.com/watch?v=ILsA4nyG7I0&pbjreload=10> [20/10/2017] [↑](#footnote-ref-4)
6. <https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py> [17/10/2017] [↑](#footnote-ref-5)
7. <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&list=PL-osiE80TeTsqhIuOqKhwlXsIBIdSeYt> [17/10/2017] [↑](#footnote-ref-6)
8. <https://www.youtube.com/watch?v=3zeg7H6cAJw&t=222s> [17/10/2017] [↑](#footnote-ref-7)
9. <https://www.youtube.com/watch?v=ILsA4nyG7I0&pbjreload=10> [20/10/2017] [↑](#footnote-ref-8)
10. <https://www.youtube.com/watch?v=owglNL1KQf0&index=11&list=PLQVvvaa0QuDe8XSftW-RAxdo6OmaeL85M> [15/4/2017] [↑](#footnote-ref-9)
11. <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&list=PL-osiE80TeTsqhIuOqKhwlXsIBIdSeYt> [15/4/2017] [↑](#footnote-ref-10)
12. <https://www.youtube.com/watch?v=GOFws_hhZs8> [10/5/2017] [↑](#footnote-ref-11)
13. <https://www.youtube.com/watch?v=5qJY-ZaKSic&t=190s> [1/5/2017] [↑](#footnote-ref-12)
14. <https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py> [17/10/2017] [↑](#footnote-ref-13)
15. <https://www.youtube.com/watch?v=3zeg7H6cAJw&t=222s> [17/10/2017] [↑](#footnote-ref-14)
16. <https://www.youtube.com/watch?v=ILsA4nyG7I0&pbjreload=10> [20/10/2017] [↑](#footnote-ref-15)
17. <https://www.youtube.com/watch?v=mXJTrH69JM0> [24/10/2017] [↑](#footnote-ref-16)
18. <https://gym.openai.com/envs/CartPole-v0/> [17/10/2017] [↑](#footnote-ref-17)