*Lachlan Dauth*

*FOP COMP1005/5005*

*FOP Assignment for 2022 -*

WHAT IS THE TIME MR WOOLF?

# The Task

The task is to create a simulation of a game or scenario using object-oriented programming and other techniques learnt in this unit. I will need to simulate the movement of classes of beings in an environment. There will need to be multiple classes of characters and different modes or states for the simulation to be in.

## A list of requirements for the simulation are as follows:

1. Movement should change and adapt to the mode of the simulation.
2. There show be a file showing how the terrain is formed. This should impact the characters motion.
3. The characters should be able to interact in the simulation. Eg seeing other being and having a function call when they interact.
4. The program should use command line arguments rather than fixed variables.
5. The program must have zones that affect the players and initiate a function.
6. Made the data and simulation easy to see and interpret.
7. The program should have 3 or more different scenarios.

# What am I Doing

I have decided to do a simulation of the game ‘what’s the time mr Woolf.’ This game is easy to understand and will be a great project to model. The simulation will have two classes the pig and the wolf. The wolf will pick a ‘time’ either an integer or ‘dinnertime’. If it is ‘dinnertime’ the wolf will chase the pigs to the left of the screen. This scenario ends if the wolf reaches a pig or if all the pigs reach the far left of the screen. If the ‘time’ is an integer the pigs will move to the right of the screen a distance based on the ‘time’ and other factors. If a pig makes it to the far right of the screen that pig wins, if not another ‘time’ will be picked.

To calculate how fast a pig or wolf should be they will have a speed variable. They will also be affected by the terrain. When the wolf is chasing the pigs, it will travel to the closest pig.

To create the terrain, I will read in terrain data and use this as both a background image and as an input to the objects move functions.

After the simulation is made, I will have another version that outputs if the fox won to a file. This version will come without the graphing to speed up processing. I will also have a program similar to dosage\_sweep.sh from prac08 to test a range of values for variables like the chance a ‘time’ will be ‘dinner time’.

## How my program fits the list of requirements:

1. My simulation has 2 moving modes for the pigs and one for the wolf. My movement is not based on Moore or Von Neumann neighbourhoods. Rather my movement is based on Euclidean geometry. This has the advantage of having a greater connection to the real world and makes more intuitional sense. The drawback is the movement functions can become messy and the creatures don’t fit into boxes having positions of a float rather than an integer.
2. My game has terrain built into the simulation. It will be read in through a csv, this will help multiple different scenarios to be easier to implement. The terrain will affect how fast the creature’s travel. The types of terrain will be represented with the shade of green.
3. The pigs and wolves interact extensively in this simulation. The wolf can ‘see’ all of the pigs and catch them. Wolves can catch pigs when they are less than one step away.
4. The non-moving targets in this simulation are the far left and right sides of the screen. When a pig reaches the far right it wins the game. If all the pigs reach the far left of the screen they win.
5. The program will output data in multiple ways using matplotlib and print statements. The sweeper program will output the data, a rolling average and a distribution that best fits the data. All these sources should result in easy data interpretation.
6. My program will use 6 command line arguments to control simple variables.
7. For the multiple scenarios I will change factors like the terrain, the size of the board, the number of pigs, the position of the pigs or even having multiple wolves.

# How to use the programs

The first program called game.py is used to simulate individual games:

1. A screen shot of numbers

   Description automatically generated with low confidenceFirst set board.csv like on the right
   1. Chart

      Description automatically generatedThe value of a cell is the relative difficulty to traverse the square in the plot this is represented by a darker square as you can see from the plot
   2. Squares with a value of p will spawn a pig
   3. Squares with a value of w will spawn a wolf
   4. Squares with animals will have the relative difficulty of the square set to the default terrain value
   5. Set terrain values to above 0 and values above 1 are best
2. Put in command line arguments
   1. If not command line arguments are put in the variables have default
   2. If one argument is given it will control the dinner chance. This is the chance that it will be dinnertime on any given turn. A value of 0 will have it never be called and 1 will have it always be called
   3. If 6 are given it will control in this order:
      1. Dinner chance
      2. The pig speed
      3. The wolf speed
      4. The lowest time the wolf will pick
      5. The highest time the wolf will pick
      6. The amount of inter ‘hour’ calculation e.g. if the time is 3 and this value is 10 the program will 30 steps of a tenth of the normal size

Another program called game2.py is the same program with 3 changes:

1. No plots to speed up processing
2. No print statements
3. Output to a csv file

dinChanceSweep.sh sweeps through values of dinner chance. To use it:

1. Input command line arguments
   1. The lowest value for dinner chance (inclusive) (I recommend 0)
   2. The highest value for dinner chance (exclusive) (I recommend 1)
   3. The steps for dinner chance
   4. The number of trials per step (it will get rounded up to the nearest multiple of 8 due to a crude but successful attempt of parallel processing)
   5. The pig speed
   6. The wolf speed
   7. The lowest time the wolf will pick
   8. The highest time the wolf will pick
   9. The amount of inter ‘hour’ calculation
2. What the program does
   1. Creates and copies files into a folder
   2. Echoes the parameters
   3. It starts a timer
   4. It loops through the program
   5. Echoes the time taken
   6. Runs dataPro.py
      1. Plots the win chance on dinner chance
      2. Plots a rolling average with a window of 2%
      3. Finds the best distribution to fit the data
      4. Plots the distribution

# Functions:

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Location | Description | Testing |
| Players – dist2points | Players.py line 10 | Returns the distance between to points given their x and y coordinates. This is done using Pythagoras. This function is used for the ‘closest’ function. | I tested this program by feeding the function Pythagoras triples and seeing if the output was correct. |
| Players – closest | Players.py line 19 | Finds the closest object to a main point, it returns the coordinates and the distance to the object and the name of the object. The function works by first simplifying the function into just a list of points. It then loops through all the points. It applies the ‘dist2points’ function to the point and the main point. If it is the closest point so far the function will remember the name, point and distance. At the end of the loop it will return the name, point and distance. This is useful for the wolfs chasing code. | I tested this program by printing the closed pig and seeing if it was correct. |
| Players – wolf | Players.py line 36 | The wolf class for the simulation. It sets the time, and chases the pigs. The wolf has 2 functions, chase, and time. The time function picks a new time based on the min time, max time, and dinner chance. The chase function calls the closest function with the pigs as the list of objects. If the wolf can move further than the distance to the pig the pig is caught. Else the wolf moves. | I check to see if he is calling the correct times by running the program many times. I will check his chase function the same way. |
| Players – pig | Players.py line 100 | The pig class for the simulation. The pig has two simple functions, move towards, and run away. Move towards moves to the right of the screen, winning the game if it can go all of the way. Run away moves to the left of the screen, the pig is safe if he gets all of the way. | I checked that move towards moves right and wins if it gets to the end by setting dinner chance to 0. I checked that run away works by setting dinner chance to a high value and seeing if it worked |
| Game – plot players | Game.py line 12 | Plots a list of objects on the screen with annotations, works with pigs and wolfs. | I plot the initial values at the start of the sim to check if the function is written correctly |
| Game – main | Game.py line 32 | The main function for the simulation.  It does these things:   1. Reads the board.csv file 2. Creates variables with default values 3. Processes command line args 4. Uses the board.csv values to create pigs and wolfs 5. Uses the board.csv values to make a background image 6. Displays the initial setup 7. Creates a loop 8. If it’s not dinnertime a new time is picked 9. If that’s not dinnertime the pigs move 10. If it is dinnertime the pigs run away and the wolf chases | I tested a range of values and fix bugs until the output was reasonable. |
| DataPro | DataPro.py | The data pro script processes the out.csv file. It generates the standard plot, a rolling average, and a distribution. | I tested this on a range of out.csv files making sure the graph always fits the data. |

# My results:

I am going to explore different starting scenarios for this project. I will run the sweeper program for each of them and find how they affect the distribution of results.

The scenarios are:

1. One wolf, six pigs, and a long board with hard terrain

Chart, waterfall chart

Description automatically generated

1. One wolf, six pigs, and a long board with no hard terrain

Chart

Description automatically generated with medium confidence

1. One very fast wolf, six pigs, and a long board with hard terrain

Chart, waterfall chart

Description automatically generated

1. Three wolves, six pigs, and a long board with hard terrain

Chart, waterfall chart

Description automatically generated

1. One wolf, six pigs, and a short board with hard terrain

Chart

Description automatically generated

I will test these scenarios and find both the best amount of time for the wolf to call dinner time for the wolf to win and what his chances are of winning. I picked these scenarios as they cover a wide variety of factors. They also only have one factor different from the first scenario to isolate the effects of each factor. I will now show the code and values for board.csv I used to create the results. I will also share a graph of the output.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Board.csv value (this controls the location of the pigs the board size and terrain features) | The command to run the sweeper | The output (graph of the relationship between dinner chance and the rate the wolf wins) | Analysis of the output |
| One wolf, six pigs, and a long board with hard terrain | p, 2, 1, 1, 8, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 6, 1, 5, 1, 1, 1, 1, 1, 1, 1  p, 2, 1, 1, 7, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 9, 1, 5, 4, 1, 9, 1, 1, 1, 1  p, 2, 1, 3, 1, 1, 2, 1, 2, 1, 4, 3, 1, 3, 1, 1, 1, 5, 1, 6, 1, 8, 1, 8, 1  p, 1, 1, 8, 1, 1, 2, 1, 1, 1, 1, 3, 1, 3, 4, 1, 1, 4, 1, 6, 1, 8, 1, 1, w  p, 1, 1, 3, 4, 1, 2, 3, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 6, 1, 8, 1, 1, 1  p, 5, 1, 3, 1, 5, 1, 7, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 8, 1, 1, 9, 9, 1 | sh dinChanceSweep.sh 0 1 0.0005 32 1 1.5 3 6 5 | Chart, histogram  Description automatically generated | In this graph the blue line, the acutual data, is indecipherable. This is due to the low amount of samples. The rolling average, the oragne line, gives a far better visualisation of the data. We can see the graph follows a disibution and has a large right skew. The best value for dinner chance is around 0.15 and results in a win rate of 0.12. The fitted distribution in green fits the data quite well. |
| One wolf, six pigs, and a long board with no hard terrain | p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1  p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1  p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1  p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, w  p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1  p, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 | sh dinChanceSweep.sh 0 1 0.0005 32 1 1.5 3 6 5 | Chart, histogram  Description automatically generated | The data in this graph is also right skewed but has many differences. It does not as cleanly fit into a skewed distibution as seen by the difference between the rolling average and the distibution. The best value for dinner chance is far larger than in the first scenario at 0.22 and results in a win rate of 0.12. This means this scenario favours the wolves and the wolves should call dinner time later. |
| One very fast wolf, six pigs, and a long board with hard terrain | p, 2, 1, 1, 8, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 6, 1, 5, 1, 1, 1, 1, 1, 1, 1  p, 2, 1, 1, 7, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 9, 1, 5, 4, 1, 9, 1, 1, 1, 1  p, 2, 1, 3, 1, 1, 2, 1, 2, 1, 4, 3, 1, 3, 1, 1, 1, 5, 1, 6, 1, 8, 1, 8, 1  p, 1, 1, 8, 1, 1, 2, 1, 1, 1, 1, 3, 1, 3, 4, 1, 1, 4, 1, 6, 1, 8, 1, 1, w  p, 1, 1, 3, 4, 1, 2, 3, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 6, 1, 8, 1, 1, 1  p, 5, 1, 3, 1, 5, 1, 7, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 8, 1, 1, 9, 9, 1 | sh dinChanceSweep.sh 0 1 0.0005 32 1 3 3 6 5 | Chart, histogram  Description automatically generated | This graph is right skewed and has a peak at 0.16 with a value of 0.33. It has a simular shape to the first senario. The faster speed of the did not siginifgantly change the best value of dinner chance. It only effected the chance the wolf won over doubling the win rate. |
| Three wolf, six pigs, and a long board with hard terrain | p, 2, 1, 1, 8, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 6, 1, 5, 1, 1, w, 1, 1, 1, 1  p, 2, 1, 1, 7, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 9, 1, 5, 4, 1, 9, 1, 1, 1, 1  p, 2, 1, 3, 1, 1, 2, 1, 2, 1, 4, 3, 1, 3, 1, 1, 1, 5, 1, 6, 1, 8, 1, 8, 1  p, 1, 1, 8, 1, 1, 2, 1, 1, 1, 1, 3, 1, 3, 4, 1, 1, 4, 1, 6, 1, 8, 1, 1, w  p, 1, 1, 3, 4, 1, 2, 3, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 6, 1, 8, 1, 1, 1  p, 5, 1, 3, 1, 5, 1, 7, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 8, w, 1, 9, 9, 1 | sh dinChanceSweep.sh 0 1 0.0005 32 1 1.5 3 6 5 | Chart, histogram  Description automatically generated | This graph is right skewed and has a peak at 0.13 with a value of 0.17. It has a simular shape to the first senario and peak. This shows how little the extra wolves effected the result. I believe this is because the wolves started so close together. If a wolf was allowed to start at the beginning of the track or the middle, I believe that they would have a greater effect. |
| One wolf, six pigs, and a short board with hard terrain | p, 2, 1, 8, 1, 1, 1, 1, 1  p, 2, 1, 7, 9, 1, 1, 1, 1  p, 2, 3, 1, 1, 8, 1, 8, 1  p, 1, 8, 1, 1, 8, 1, 1, w  p, 1, 3, 4, 1, 8, 1, 1, 1  p, 5, 3, 1, 1, 1, 9, 9, 1 | sh dinChanceSweep.sh 0 1 0.0005 32 1 1.5 3 6 5 | Chart, histogram  Description automatically generated | This graph is right skewed and has a peak at 0.19 with a value of 0.1. This graph follows a simular shape to the scecond scenario as they have a lot in common. It has the same speed variables along with the only change being the terrain. Even the terrain that seems completely different is simular. In this scenario the pigs have to travel 23.7 terrain squres on average and 25 on the second scenario. |

# Conclusions and further investigation

I have learned a lot during this investigation and have worked on an interesting project. I have found the speed of the animals to have the greatest impact on whether the wolf wins, while the amount of distance the pig has to travel has the greatest impact on the best value for dinner chance. Some further projects I could expand this into are, making a pathfinding algorithm to find the best path for both the wolf and pigs, and making a sweeper program for other variables. I would also like to experiment with a sweeper program going through 2 variables at a time. This data could be represented with a variable on each axis and a heat map to show the win rate.

# Bibliography

ely, lothario, & Paul. (2011, Febuary 22). *How do you directly overlay a scatter plot on top of a jpg image in matplotlib / Python?* Retrieved from Stack Overflow: https://stackoverflow.com/questions/5073386/how-do-you-directly-overlay-a-scatter-plot-on-top-of-a-jpg-image-in-matplotlib

Geeks for Geeks. (2021, November 28). *How to Calculate Moving Averages in Python?* Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/how-to-calculate-moving-averages-in-python/

Rahimi, A., & Paul, P. (2018, May 2). SciPy skewnormal fitting. Retrieved from Stack Overflow: https://stackoverflow.com/questions/50140371/scipy-skewnormal-fitting

SciPy. (n.d.). scipy.stats.skewnorm. Retrieved from SciPy documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skewnorm.html

COMP1005. 2021. “shrimp.py” Assignment sample code, COMP1005 Fundamentals of Programming, Semester 1, 2022.

COMP1005 (2021) Practical 8 Walkthrough Video, iLecture, Semester 1, 2022