A Markov chain I a system the transitions from one state to another state using a random process that is categorized as memoryless. Andrey Markov created Markov chain. Although I am not able to understand how to program one of these for self-use, I do understand where it is implemented in games and in some other uses outside of games.

Markov chain is used in queuing theory back in the early 1900’s. It was used for optimizing the performance of telecommunications networks. It is also used in statistics using random numbers to reflect desired probability distributions. One more use is for finances. Markov chains to model different phenomena’s like price models and market crashes. These are just a small selection of how Markov chains are used in real world applications. Next we will look at how this system can be used in a gaming environment using three case examples.

First example of using Markov chains in games is your traditional board game like monopoly, sorry or even checkers. In all of these games, your pawn starts off at a certain state, and any given variable like the dice rolling or it being your turn in checkers can change the state of that pawn. Then the next move is based on that state and not the one that you began the game with. Where this is memoryless in game design is you would react off your current location and not program it to count how far you’ve gone away from the home location and retain that value.

The second example is very prominent in open-ended games of today such as tell tales the walking dead. In the game, the state of your character and his or her surrounding cast its based on how you handle certain decisions in normal dialogue. A conversation that went wrong with your best friend can move you both into a different state that will be maintained until an opportunity presents itself to change that state.

The third example, I believe, would be used in a turn based action style game. A game like Pokémon would use this with certain moves like rage where the state of your Pokémon will change with every turn you get. The Pokémon will get stronger by a random amount, and will deal damage based on that amount. The longer that Pokémon stays alive, the stronger it is. On the internals, programming the random stat boost is similar to the dice rolling in a game of monopoly. You move to a different state of strength and only build on that and not from your original starting position.

Markov chains have plenty more uses in some of the most natural use cases that aren’t noticeable to a normal person. Implementing them can be a bit troublesome but once you have the right set up, you’ll have a free flowing, memory conscious system any programmer would be proud of.