**XOGO – THE CARD GAME**

**J COMPONENT PROJECT REPORT**

**Review - I**

Submitted by

**RITIK GUPTA (18BCE0154) MAITRISH JAIN (18BCE0574)**

**AAYUSH SAXENA**

**(18BCE0853)**

**SHASHANK RAJORIA**

**(18BCE2231)**

To

**Prof. Harshita Patel**

Artificial Intelligence(CSE3013)

*in partial fulfillment for the award*of the degree of

**B. Tech**

in

##### Computer Science and Engineering

****

*Vellore-632014, Tamil Nadu, India*

**School of Computer Science and Engineering**

January 2020

**Objective**

UNO is a game totally based on luck. The game becomes boring and can even run for a long time when number of players are less. XOGO is a modified version of UNO which is addictive to play again and again and more you play the tuffer it would become to defeat the BOT as it will learn from its previous mistakes.

**Abstract**

1.XOGO is an exciting card-based game in which you take on the challenge vs computer in a 1 vs 1 scenario.

2.Game includes some exciting power cards, where number of moves are limited, and objective is to finish the game with least number of cards.

3.The game contains basic 1-9 cards with colour coding.

4.The card set also contains power cards. Power cards can be used to reduce number of cards, to sway the game in your favour.

5.Each player starts off with same number of cards and in the end player with least cards is the king of XOGO.

6.The AI part is Computer or bot keeps on learning with each game.

**Hardware and Software Required**

**1.PYTHON-PYCHARM**

To run the program that we have developed for the game.

**2.TENSORFLOW**

For numerical computation using data-flow graph.Mainly will be used for understanding,discovering and prediction.

**3.PANDAS**

Used for data manipulation and analysis.In particular, it offers data structures and operations for manipulating numerical tables and time series.

**4.LAPTOP**

Any laptop with specs>=i3 processor will be sufficient to run this application.

**Literature Riview**

**PHASE 1**

**History of Video Games and Their Use of AI**

The history of artificial intelligence in video games can be dated back to the mid sixties. The earliest real artificial intelligence in gaming was the computer opponent in “Pong” or its variations, which there were many. The incorporation of microprocessors at that time would have allowed better AI which did not happen until much later. Game AI agents for sports games like football and basketball were basically goal-oriented towards scoring points and governed by simple rules that controlled when to pass, shoot, or move.

There was a much better improvement in the development of Game AI after the advent of fighting games such as “Kung Foo” for Nintendo or “Mortal Kombat”. The moves of the computer opponents were determined by what each player was currently doing and where they were standing. In the most basic games, there was simply a lookup table for what was currently happening and the appropriate best action. Enemy movement was primarily based on stored patterns. In the most complex cases, thecomputer would perform a short minimax search of the possible state space would be performed and best action would be returned.

The minimax search had to be of short depth and less time consuming since the game was occurring in real-time. The emergence of new game genres in the 1990s prompted the use of formal AI tools like finite state machines. Games in all genres started exhibiting much better AI after starting to use nondeterministic AI methods. Currently, driving games like “Nascar 2002” have computer controlled drivers with their own personalities and driving styles.

MAIN AREAS WHERE AI APPLIES TO GAMES:

• Non-Player Character AI

• Decision Making and Control

• Machine Learning

• Interactive Storytelling

• Cooperative Behaviours

• Player Control

• Content Creation

**PHASE 2**

**Traditional Methods**

The traditional method for implementing game AI was by mainly using tri-state state machines.

The complexity of the AI that could be implemented was restricted because of the tri-state statemachines. Traditional AI implementation methods also predominantly followed the deterministic implementation method. In the deterministic implementation method the behaviour or the performance of the NPC’s and the games in general could be specified before hand and it was also too predictable.

There was a lack of uncertainty which contributed to the entertaining factor of the game. An example of deterministic behaviour is a simple chasing algorithm.

**Current Methods**

The current methods which are used for implementing Game AI vary from neural network,Bayesian technique, genetic algorithms, finite state machines and Pathfinding. All these methods are feasible but not applicable in every situation given. Game AI is a field where research is still going on and developers are still perfecting the art implementing all these methods in any situation given.

**Pathfinding and Steering** - Pathfinding addresses the problem of finding a good path from the starting point to the goal, avoiding obstacles, avoiding enemies, and minimizing costs in the game.

Movement addresses the problem of taking a path and moving along it. At one extreme, a sophisticated pathfinder coupled with a trivial movement algorithm would find a path when the object begins to move and the object would follow that path, oblivious to everything else. At theother extreme, a movement-only system would not look ahead to find a path (instead, the initial "path" would be a straight line), but instead take one step at a time, considering the local environment at every point. The gaming industry has found out that the best results are achieved by using both pathfinding and movement algorithms.

**Neural Networks**

Neural networks can be used to evolve the gaming AI as the player progresses through the game.Neural networks act as a way to map parameters from one space, an input space, to anotherspace, the output space. The mapping may be highly nonlinear and depends on the structure of the neural network and how it was trained. The best thing with neural networks is that they will continually evolve to suit the player, so even if the player changes his tactics, before long, the network would pick up on it.

Some advantage of neural networks over traditional AI are, using a neural network may allow game developers to simplify the coding of complex state machines or rules-based systems by relegating key decision-making processes to one or more trained neural networks and neural networks offer the potential for the game's AI to adapt as the game is played.

The biggest problem with Neural Networks programming is that no formal definitions of how to construct an architecture for a given problem have be discovered, so producing a network to perfectly suit your needs takes a lot of trial-and-error.

**Genetic Algorithms** - Genetic algorithms offer a way to solve problems that are difficult for traditional game AI techniques. We can use a genetic algorithm to find the best combination of structures to beat the player. So, the player would go through a small level, and at the end, the program would pick the monsters that faired the best against the player, and use those in the next generation. Slowly, after a lot of playing, some reasonable characteristics would be evolved. Genetic algorithms (GAs) are one of a group of random walk techniques. These techniques attempt to solve problems by searching the solution space using some form of guided randomness. Another technique of this type is simulated annealing.

Larger populations and more generations will give us better solutions. This means that Genetic Algorithms are better used offline. One possible way of doing this is by doing all of the Genetic Algorithm work in-house and then releasing an AI tuned by a GA. By having a GA engine to work on the user’s computer while the game is not being played this can be achieved to a certain extent.

**PHASE 3**

**Need for a Learning Ability**

Learning A.I. would allow the game to surprise the player and maintain the suspension of disbelief as the systems remains invisible. Many games companies are currently looking at the possibility of making games that can match the player's ability by altering tactics and strategy, rather than by improving the ability of opponents.

There are few games in the market currently which can uncover a player's tactics and adapt to them. Even on the toughest difficulty settings of most games most players tend to develop a routine, which if they find using successful, will continue using that so that they win more often than not. What would make it interesting at this point is if the AI could work out their favourite hiding places, or uncover their winning tacticsand adapt to them. This is a very important feature as it would prolong game-life considerably.

Central to the process of learning, is the adaptation of behavior in order to improve performance.Fundamentally, there are two methods of achieving this. Either by directly changing the behavior and by testing modifications to it, and indirectly by making alterations to certain aspects of behavior based on observations.

**References and Research Papers**

1.<https://patents.google.com/patent/US6651984B1/en>

2.<https://www.researchgate.net/publication/220344080_Machine_learning_and_games>

3.<https://www.researchgate.net/publication/280308617_Solving_Multiplayer_Games_with_Reinforcement_Learning>

4.Machine Learning in Games: A Survey Johannes F¨urnkranz : <https://pdfs.semanticscholar.org/3e1a/62d3bfc2de07762b114716bb9a1a17b65b3f.pdf>

5.Application of Self-Play Reinforcement Learning to a Four-Player Game of Imperfect Information

Henry Charlesworth: <https://arxiv.org/pdf/1808.10442>

6.The Hanabi challenge: A new frontier for AI research: <https://www.sciencedirect.com/science/article/pii/S0004370219300116>

7.UNO is hard, even for a single player: <https://link.springer.com/chapter/10.1007/978-3-642-13122-6_15>

8.Game refinement theory and multiplayer games: case study using UNO:<https://dspace.jaist.ac.jp/dspace/handle/10119/12881>

9.<https://www.researchgate.net/publication/221024452_Opponent_Modeling_in_Real-Time_Strategy_Games>

10.Opponent modelling for case-based adaptive game AI: <http://www.spronck.net/pubs/bakkes_journalOM.pdf>

**WHAT'S DIFFERENT**

**1.WORKING**

Simple and very easy to use.

**2.EXPERIENCE BASED LEARING**

Bot will keep on learning from each and every game it plays.

**3.ANTICIPATING MOVES**

Our bot will also be capable of anticipating the moves that can lead to its victory.

**4.PREDICTING OUTCOME**

Our bot will also be able to predict the outcome resulting from the particular move.

**FUTURE PROSPECTUS**

**DESIGNING COMPLEX GAMES**

Many complex games can be designed through AI such as shooting games in which bot enemy can learn from the behaviour of the user.

**MULTIPLAYER GAME**

This game currently can be played between 2 people.In future,we can also implement in such a way that it becomes a multiplayer game.