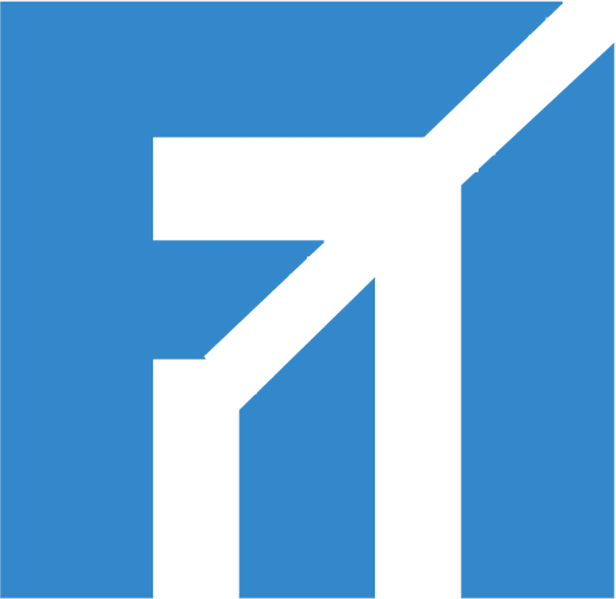
**“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI**

**FACULTY OF COMPUTER SCIENCE**



BATCHELOR THESIS

**Application of K-means Clustering in Video Games**

Proposed by

**Andrei Ghiran**

**Session:** 06.2020

Scientific coordinator

**Prof.lect.dr. Moruz Alex**

**“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI**

**FACULTY OF COMPUTER SCIENCE**

**Application of K-means Clustering in Video Games**

**Andrei Ghiran**

**Session:** 06.2020

Scientific coordinator

**Prof.lect.dr. Moruz Alex**

SUMMARY

Introduction 3

1.Similar Applications 4

1.1. Deep learning agents 4

2.2. Computer vision-based agents 5

2. Implementation 7

2.1. AI adversary 7

2.1.1 The K-means Algorithm 7

3. User Manual

4. Conclusion and Future Goals

INTRODUCTION

From the day I started my studies at the faculty of computer science I knew I wanted to pursue a career in game developing. To be one step closer to my career of choice I decided that for my bachelor thesis I would combine my passion for game developing and the limitless applications of machine learning. Using the Unity engine I developed a simple game that showcases the K-means algorithm by using it to train an Artificial Intelligence adversary for the player.

The game has a simple premise, the player controls a blue square that can only move on a grid in the 4 cardinal directions and must reach the goal, the AI adversary, with the same movement constraints as the player and only moving at the same time as the player, must intercept and bump into them to end the game. The adversary doesn’t know the position of the player at all times, he has a vision radius in witch he can detect the player, acquire his position and save it in a file for later use. The K-means algorithm uses the saved player locations to form clusters and determine an advantageous position from where the adversary can intercept the player, the using the A star algorithm the adversary will find a path to this more advantageous position and follow it.

**1.SIMILAR APPLICATIONS**

Information on the use of machine learning techniques in video games is mostly known known publicly through research projects as most gaming companies don’t publish specific information about their intellectual property. The most common use of clustering algorithms in the domain of video games is the classification of player behaviour in massive multiplayer online games (MMOs). These behaviours range from completing missions to exploring the game wold and even spending time with other payers. The classification is used by the developers to understand what most players enjoy doing the game and this knowledge influences the further development of the game. K-means clustering is also commonly used for procedural content generation.

The most popular application of machine learning in games is the use of deep learning agents that compete with professional human players. Another popular type of agent is the computer vision-based AI player. The most significant application of machine learning has been done on games such as Dota 2, the StarCraft series, Pong and Doom. Games that did not originate as video games such as Chess and Go have also been subject to machine learning research.

**1.1. Deep Learning Agents**

Deep learning focuses heavily on the use of artificial neural networks that learn to solve complex tasks. Deep learning uses multiple layers of artificial neural networks and other techniques to progressively extract information from an input. Due to this complex layered approach, deep learning models often require powerful machines to train and run on.

**Chess** is considered a difficult AI problem due to the computational complexity of its board space, its state space being 10^120 possible board states. Similar strategy games are often solved with some form of a Minimax Tree Search. These AI agents have been able to beat professional human players, such as the historic 1997 Deep Blue versus Garry Kasparov match. Since then, machine learning agents have only shown even greater success.

**Go** poses an even more difficult AI problem then chess. The state space of Go is approximately 10^170 possible board states, much greater then the state space of chess.

Google’s 2015 AlphaGo was the first AI agent to beat a professional Go player. It used a deep learning model to train the weights of a Monte Carlo tree search. The deep learning model consisted of 2 Artificial Neural Networks, one network to predict the probability of potential moves made by the opponent and the other network to predict the win chance of a given state. The combination of the deep learning model and the Monte Carlo tree search allowed the agent to explore the state tree much more efficiently than if it was using just the Monte Carlo tree search.

AlphaGo was initially trained on games against human players and later on games against itself. Later, in 2017, another implementation of AlphaGo, AlphaGoZero, was made public. This implementation was able to entirely train by playing against itself and it would train much faster then its predecessor. In the final stages of its training AlphaGoZero would develop new advanced strategies that, until then, were never used by a Go player.

**The StarCraft series** is a series of real-time strategy video games that have become very popular environments for AI research. Blizzard, the game’s developer, and DeepMind have collaborated to release a public StarCraft 2 environment for AI research.

Alphastar was the first AI agent to beat a professional StarCraft 2 player without any in-game advantages. Initially for its deep learning network the agent used a simplified zoomed out version of the gamestate as imput, but later it was updated to play using a camera like other human players. Up to the date of writing this document the code or architecture of the model have not been publicly released by the developers of Alphastar, but they have listed several machine learning techniques that they used, such as relational deep reinforcement learning and long short-term memory.

Initially Alphastar was trained with supervised learning, it watched replays of many human games in order to learn basic strategies, later it trained against different versions of itself an was improved trough reinforcement learning. The final version was successful but it was only trained to play on a specific map and matchup.

**Dota 2** is a multiplayer online battle arena (MOBA) game. Because of its complexity traditional AI agents have not been able to play at the same level as professional human players. The only widely published information on AI agents attempted on Dota 2 is OpenAI’s deep learning Five agent.

OpenAI Five utilized separate Long Shor-Term Memory networks to learn each hero in the game. It trained using a reinforcement learning technique known as Proximal Policy Learning. It trained against itself on a system containing 256 GPUs and 128,000 CPU cores accumulating 180 years of game experience each day. OpenAI Five's first public appearance occurred in 2017, it won a one-on-one game against a professional Dota 2 player known as Dendi. The following year the agent advanced so much that it could performe as a team and in a 2019 series of games beat the 2018 Dota 2 champion team.

**1.2. Computer vision-based agents**

Computer vision focuses on training agents to gain a high-level understanding on digital images and videos. Many computer vison techniques incorporate forms of machine learning and have been applied to various video games. This technique focuses on interpreting game events using visual data, in some cases AI agents, using model-free techniques, have learned to play games without any direct connection to internal game logic, solely using video data as input.

**Pong** is a table tennis sports game featuring simple two-dimensional graphics, manufactured by Atari and originally released in 1972. Andrej Karpathy, Tesla's director of artificial intelligence, demonstrated that a neural network with just one hidden layer is capable of being trained to play Pong based solely on screen data.

**Doom** (1993) is a first person shooter (FPS) game. A group of student researchers from Carnegie Mellon University used computer vision techniques to create an agent that could play the game using only image pixel input from the game. The students used convolutional neural network layers to interpret image data and output valid information to a recurrent neural network which was responsible for outputting game moves.

Other uses of machine learning in video games focus more on generating procedural content rather than AI agents that interact with the player. Procedural content generation is the process of generating data algorithmically rather than manually. Examples of procedurally generated content can be found in the weapons of Borderlands 2, the map layouts of Minecraft and the entire universes in No Man’s Sky. Common approaches to procedural generation include techniques that involve grammars, search-based algorithms, and logic programing. These approaches require a human to manually the range of content possible, but machine learning is theoretically capable of learning the features of possible content when given examples to train on, thus reducing the time spent on defining the range of desired content. Machine learning techniques used for content generation include Long Short-Term Memory, Recurrent Neural Networks, Generative Adversarial Networks and K-means clustering.

**Galactic Arms Race** is a space shooter video game that uses neuro-evolution powered procedural content generation to generate unique weapons for the player, based on his personal preferences. This game was a finalist in the 2010 Indie Game Challenge and its related research paper won the Best Paper Award at the 2009 IEEE Conference on Computational Intelligence and Games. The content was generated using a form of neuro-evolution called cgNEAT.

**2. IMPLEMENTATION**

The project had been developed using the Unity engine, a cross-platform platform game engine developed by Unity Technologies. The engine can be used to create three-dimensional, two-dimensional, virtual reality and augmented reality games as well as simulations, animations and other experiences. Several major versions of unity have been released since its launch. As of writing this document the latest stable version in 2019.4.0, the project has been developed with version 2019.3.4f1.

The project can be divided in three integral components: the AI adversary, the player characters and the environment.

**2.1. AI adversary**

The core components of the AI are the K-means clustering algorithm for determining an advantageous position and the A star algorithm for calculating a path to that position.

**2.1.1. The K-means clustering algorithm**

Clustering is used for partitioning a set of objects into groups, called clusters. Objects that are similar, according to a certain similarity measure, are places in the same cluster, while dissimilar objects are placed in different groups.

The K-means algorithm aims to partition n objects , in k () clusters , in witch every object belongs to the nearest cluster center. The most common algorithm uses an iterative refinement technique.

Given an initial set of k means , represented by centroids corresponding to each clusters, the algorithm alternates between two steps:

**Assignment:** The algorithm assigns each object to the cluster with the nearest mean, that with the minimum Euclidean distance between the object and the centroid corresponding to that cluster. If an object has the same minimum Euclidean distance to multiple centroids, it will be randomly assigned to one of the cluster that correspond to those centroids.

, where t represents the current iteration.

**Update:** Recalculate the means (the position of the centroids) for the new clusters.

The algorithm stops when, after 2 consecutive iterations the assignments no longer change. The algorithm does not guarantee to find the optimum partitioning, but it can find a local optimum.

The algorithm is most commonly used to assign objects to the nearest cluster by Euclidean distance. Using a different distance function may result in the algorithm never stopping.