ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

ARTIFICIAL INTELLIGENCE

ARTIFICIAL INTELLIGENCE
HND
Gustavo Aranda
2021/2022
Borja Serrano García

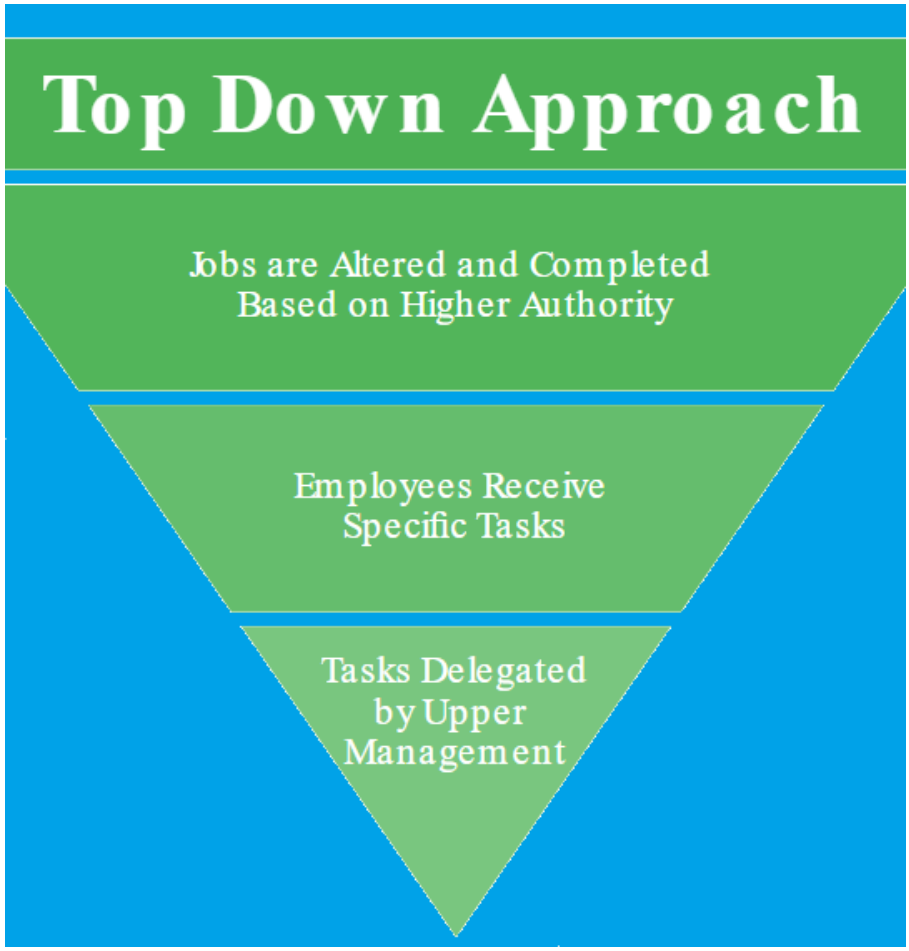
Index

1.- AI top-down technique

1.1 What is it?

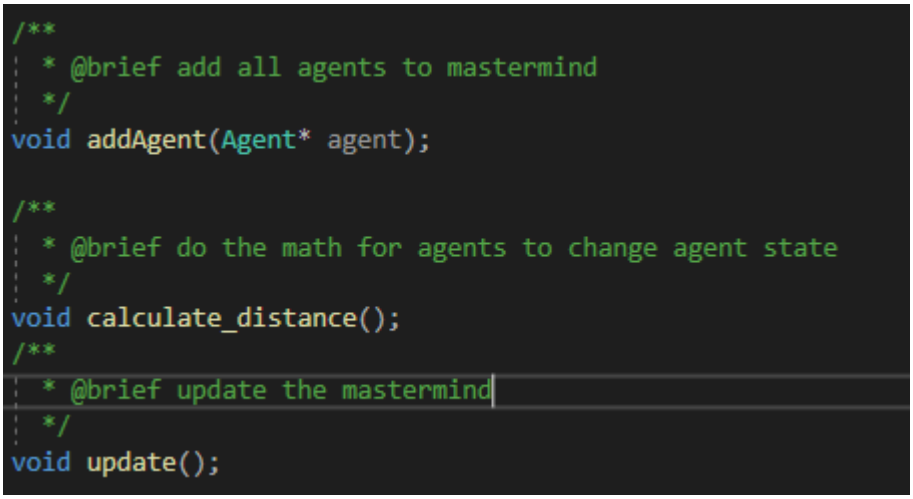
In essence the top down approach to developing AI consists of a plethora of preset methods and goals that they want the automated intelligence to achieve. Essentially the top down approach will develop a system that has pre programmed definitions on cognitive responses analytical intelligence.

The system of the approach would be to go from the top down and essentially add detail into every layer of abstraction.

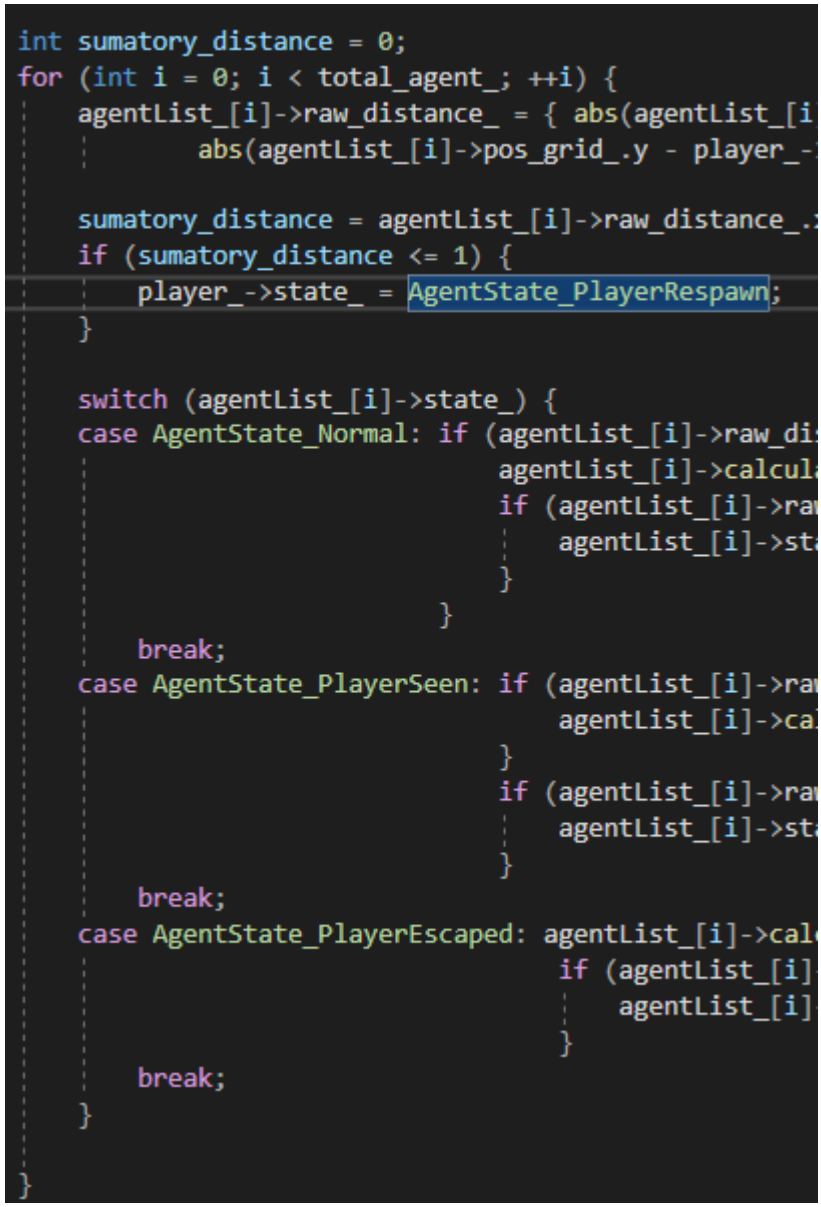


1.2 Mastermind and behavior tree

As a top down technique, I make use of a mastermind with behavior trees. The mastermind receives all the necessary information from all the elements of the scene, in this case the agents. It knows all the positions of each agent and the current state and, making use of a behavior tree like the one in the picture but simpler, it changes the states of the agents according to their programming. For example, the enemy agents are programmed so that when they detect the player at a first distance less than X they detect how many real squares of movement they have to make to reach the player. If those squares are less than Y then it changes the state from Normal to player seen.



When the player leaves the agent's range, first measuring the previous distance and then the exact number of squares. If it is less than X it changes the state back to restart and goes to its exit square of the program and, upon arrival, changes the state to Normal again.



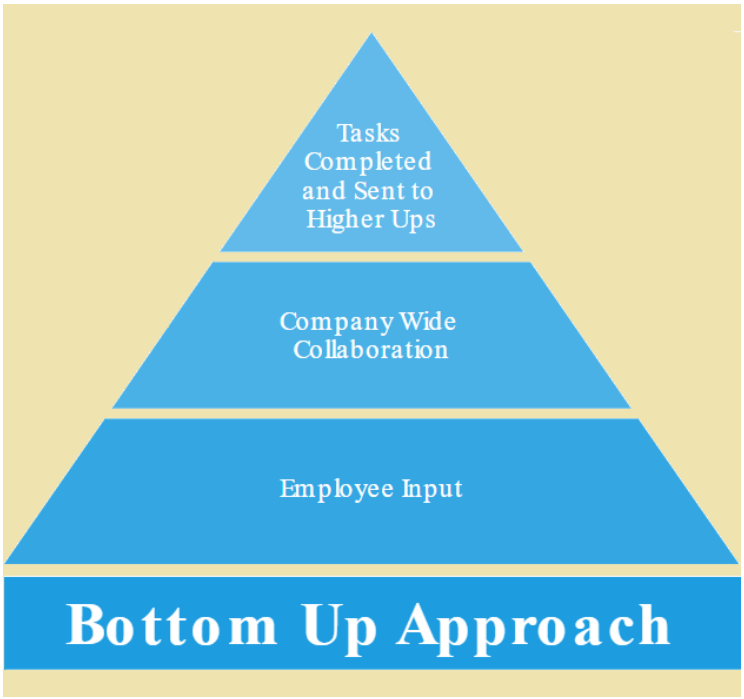
```
int sumatory_distance = 0;
for (int i = 0; i < total_agent_; ++i) {
    agentList_[i]->raw_distance_ = { abs(agentList_[i
            abs(agentList_[i]->pos_grid_.y - player_-
    sumatory_distance = agentList_[i]->raw_distance_.
    if (sumatory_distance <= 1) {
        player_->state_ = AgentState_PlayerRespawn;
    }

    switch (agentList_[i]->state_) {
    case AgentState_Normal: if (agentList_[i]->raw_di
                                agentList_[i]->calcul
                                if (agentList_[i]->ra
                                    agentList_[i]->st
                                }
                            }
        break;
    case AgentState_PlayerSeen: if (agentList_[i]->ra
                                    agentList_[i]->ca
                                }
                                if (agentList_[i]->ra
                                    agentList_[i]->st
                                }
        break;
    case AgentState_PlayerEscaped: agentList_[i]->cal
                                   if (agentList_[i]
                                       agentList_[i]
                                   }
        break;
    }

}
```

All these state changes and distance detections between agents are handled by the mastermind as it is a superior entity with the necessary knowledge to perform

2.- AI bottom-down technique

2.1 What is it?

Essentially the bottom up approach consisting of the construction of neural networks is the complete opposite of the top-down model. The bottom up approach will consist of simple methods and systems that will grow and slowly become more complex. Most of the simple systems are linked to a larger subsystem that encompasses all learned knowledge.

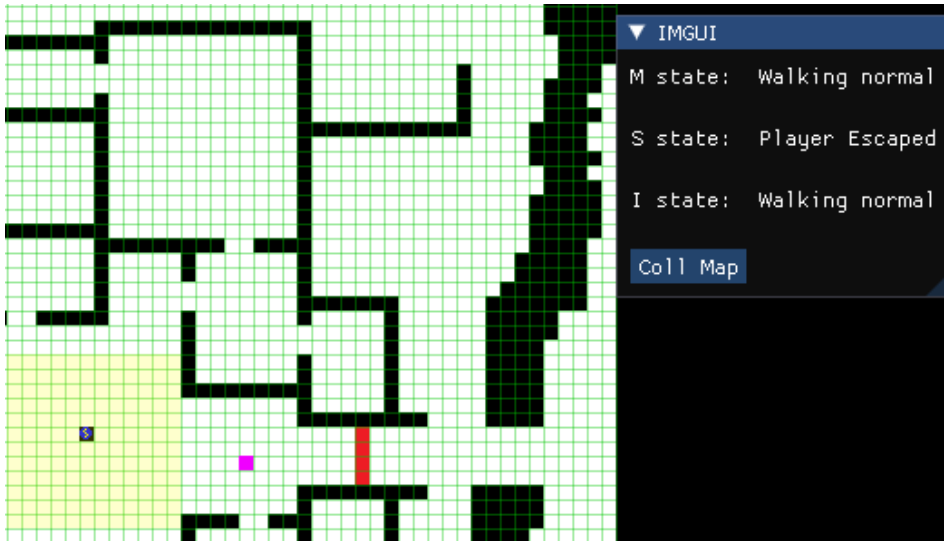


2.2 Pathfinding and tracking

As a bottom-up technique I make use of a pathfinding algorithm for the demo agents. The agents have a limited information of the environment, they are the simplest entities of the program. Their natural state is that of random movement on the map, it only detects when it moves if it can or not, if it is a suitable square or not.

When its state changes, by the mastermind, it starts with a new behavior. When that behavior is player seen, it then performs pathfinding to find the fastest path to the player and performs it. The pathfinding algorithm tries to calculate the distance from each possible movement box of the agent to the player. The agent checks a large number of squares in order to reach the player's position.

If the player disappears from the agent's environment, it changes its state and performs the algorithm again to find the way back to its initial position.


IMGUI
M state: Walking normal
S state: Player Escaped
I state: Walking normal
Coll Map

3.- Emergent AI technology: Machine Learning

3.1 What is it?

Machine learning is a branch of AI that deals with the study, research and development of self-learning machines that are capable of self-programming, learning from their own experience by combining input data and real-world situations.

The concept of machine learning (ML) refers to automatic learning or machine learning and, as such, falls within the field of artificial intelligence. The theory of how ML works is simple, machines take data and learn for themselves.

Machine learning systems allow a system to learn to recognize patterns on its own and make predictions, unlike other older techniques where manual coding of a software program with specific instructions is required to complete a task.
Using this technique, a system has the ability to quickly apply the knowledge obtained from large data sets to achieve different objectives in a short time, such as facial or voice recognition.

3.2 Types of machine learning.

The different types of machine learning are divided into three main categories: supervised learning, unsupervised learning and reinforcement learning.

Supervised learning:

In this type of learning, algorithms use data that have been previously labeled (labeled data), seeking to find a function that, depending on the input data, assigns the appropriate output label.

In addition, the algorithm is able to learn from its own previous decisions, i.e., it uses a data history where all the data used in previous occasions are found and in this way learns to assign the appropriate output label to a new value, i.e., it is able to predict the output value.

Among the main supervised learning methods we find the following:

- Decision tree: it consists of a tool that is responsible for making predictions based on a set of data, building for this purpose a series of logical diagrams that serve to represent the conditions that progressively follow one another, to finally provide a solution to a given problem. These decision trees are made up of nodes, number vectors, arrows and labels.

- Naive Bayes Classifier: this is a probabilistic classifier based on Bayes' theorem, which links the probability of event 'A' given 'B' with the probability of 'B' given 'A', together with other types of simplifying hypotheses. Briefly, one could say that this algorithm assumes that the presence or absence of a feature does not depend on the presence or absence of any other feature, since it is a variable class.

- Ordinary Least Squares (OLS): also known as linear least squares, this is a method for finding the parameters and population statistics in a linear regression model, which serves to approximate the dependence relationship between a dependent variable, another independent variable, and a random term.

- Logistic regression: this is a type of regression analysis used to predict the solution of a categorical variable based on other independent or predictor variables. The probabilities representing the possible outcome of a particular trial are modeled, as a formula of explanatory variables, using a logistic function. This algorithm is useful for analyzing binomially distributed data.

- Support Vector Machines (SVM): this is a set of algorithms related to classification and regression problems. Its operation consists of training an SVM by labeling the classes of a series of assumptions as an example, so that the machine, when faced with a new assumption, is capable of predicting the class to which it corresponds. The SVM represents, by means of a hyperplane, the different sample points in the space, making a wide separation between two

spaces, and thus making it possible to define the class in question according to the greater or lesser proximity it has with each of the classes.

- Ensemble Method: Ensemble methods or ensemble methods use multiple learning algorithms in combination to obtain better predictive performance than could be obtained from any of the learning algorithms individually. Evaluating the prediction of an ensemble of algorithms generally requires more computation than evaluating the prediction by a single algorithm, so ensembles can be seen as a way to compensate for poor learning algorithms by performing a large amount of additional computation. Fast algorithms, such as decision trees, are commonly used in ensemble methods, although slower algorithms can also benefit from ensemble techniques.

Unsupervised learning

The main feature of the unsupervised learning method is that learning is tuned and achieved through mere observations, without the need for training with data.

Thus, it differs from supervised learning in that in this case there is no a priori knowledge, but one must learn directly from what is observed. Thus, unsupervised learning generally treats the input object data as a set of random variables, thus building a density model for the data set.

Among the main unsupervised learning methods we find the following:

- Clustering Algorithms: the term clustering refers to the grouping of a set of elements in a certain way, making the elements that are in the same group (called cluster) more similar to each other (in some sense) than those elements that are in a different group. Element analysis itself is not a specific algorithm, but rather the general task to be solved. It can be accomplished by several algorithms that differ significantly from what constitutes a cluster.

- Principal Component Analysis (PCA): This technique describes a group of data in terms of new, uncorrelated variables (components). These components are ordered by the amount of original variance they describe, making the technique very useful for reducing the dimensionality of a data set. PCA is mostly used in exploratory data analysis and to build predictive models. This technique involves calculating the auto-value decomposition of the covariance matrix, usually after centering the data on the mean of each attribute.

- Singular Value Decomposition (SVD): the SVD of a real or complex matrix is a factorization of the matrix with many applications in the fields of statistics, signal processing and other disciplines.

- Independent Component Analysis (ICA): consists of a computational method used to separate a multivariate signal into its additive components assuming that the source signal has statistical independence and is non-Gaussian. This is a special case of "blind signal separation". ICA is a generalization of principal component analysis (PCA), since in both cases a linear transformation of the original data is performed, although the basic difference is that ICA does not require the original variables to have a Gaussian distribution.

Reinforcement learning

Reinforcement learning is an area that deals with machine learning aimed at solving certain sequential decision problems. It generally deals with problems related to applications in a wide variety of fields such as automatic control, medicine, operations research or economics.

The classical algorithms of this type of machine learning are based on the mathematical theory of dynamic programming, where it is assumed that the state space is discrete and that it is composed of a manageable number of states.

However, in most applications of practical interest, the space between states is continuous, which makes such classical

algorithms less useful. In order to apply this technique to continuous spaces, two requirements must be met: generalizing the behavior learned from a limited set of experiences to cases that have not been previously experienced and representing the policies in a compact form.

The combination of reinforcement learning algorithms with function approximation techniques is currently an active area of research. However, despite the progress made in recent years in this field, there are still aspects that limit the capacity of reinforcement learning in complex problems.

- Dynamic programming: this is a technique for reducing the execution time of an algorithm by using overlapping subproblems and optimal substructures.

- Q-learning: the objective of this method is to learn a policy that tells an agent what action to take under what circumstances. It does not require a model of the environment and can handle problems with stochastic transitions and rewards, without requiring adaptations. For any finite Markov decision process (FMDP), Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward over all successive steps, starting from the current state, is the maximum achievable.

- State-Action-Reward-State-Action (SARSA): this is an algorithm used to learn a Markov decision process policy. The name SARSA is given because it reflects the fact that the main function for updating the value of an element depends on the agent's current state, the action the agent chooses, the reward the agent obtains for choosing that action, the state the agent enters after taking that action and, finally, the next action the agent chooses in its new state.

4.2 How should it work in my project

Using machine learning in my project, the result that I would like to implement would be a small learning of the enemy agents on the player's behavior and another on the agent's own movement to explore certain areas of the scene.

As a simple learning of the agent about the player's behavior, the machine learning that I would apply would work when elaborating and tracking with each encounter with the player an escape movement pattern of the player to save movements in chasing him going to where I think he will escape. If the player when he sees me from his left goes to the right and up the agent goes ahead and chases him directly to the right and up, so with all the directions in which the event in which they are... can occur.

For the agent's own movement, in search of the player through the scenario, machine learning would work by determining the available zones of the scene and learning both to change the zone when it is already traveled and to travel efficiently through the area taking into account the field of view that the agent has to find the player, so it would save movement squares both to travel the area and not repeat squares and to be efficient with the movements to always be in good squares to see the player if he appears at any entrance or to be in range of vision as long as possible. For example, not to be close to the wall if the room is square because from the center he covers the whole room with his vision.

4.- Emerging AI technologies

4.1 Machine Learning as Recommendation

Currently there are many of the shopping pages that we visit today, where we can find the so-called recommendation systems within the ecosystem of video games. These systems offer us, based on the information that the system has stored about us, items that could potentially interest us. The use of these recommendation systems can already be seen on the one hand in virtual game stores such as Steam, offering users other games that might interest them based on those they have already purchased previously.

In the future, I'm sure it will refine the recommendations to a point where it can detect when you are tired of a certain genre, such as by the hours accumulated in recent weeks compared to previous ones, and it will recommend, in an accurate way, games of different genres that are accurate at that time. I also think it could get you to change genre or style of game before you are aware of saturation.

As a final input into the concept of recommendation, I would take it to the point of inclusion in the video games themselves. Games that have different classes to choose from, different skills, upgrade tree for those skills, the system itself is going to be able to recommend or activate a decision option that makes those decisions automatically.

4.2 Machine Learning as Playable experience

This is a field that, if well developed, occupies a lot of work for developers and can generate rejection or engagement in the player. In the future, the difficulty will change according to the skills, the style of play and the moment in which it is used (example sports simulator, there will not be the same difficulty in the preliminary rounds of a tournament as in the final rounds).

A good player will find a difficulty according to his ability and that difficulty will vary, not only by the player's own ability, but

also by his style of play. A game in which the player kills enemies stealthily and without being seen, the AI of the NPCs will be gradually modified to be more careful with places that are out of sight, he will join with other NPCs forming small groups so that it is not so easy to kill him. With players with a more aggressive play style, the NPCs will have a better aim, they will cover themselves by taking cover in a smart way.

In a more distant future and with the evolution and improvement of neural networks, Artificial Intelligence will be able to adapt to any scenario not contemplated by developers, which will allow video games that implement a system for creating or modifying scenarios or game rules to go a step further. In speed games, users will be able to create or modify new playable scenarios and the AI itself will be able to create NPCs that create a difficulty for the player, with which the player will be able to test his circuit and measure his capabilities. This can be taken to games like Mario Maker in which each player can upload his created level and the machine would add enemies completely adapted to the environment. This would allow the content of the games to be much greater, as the machine would not only be able to create enemies, but also companions or group members to cooperate with the player. If we add this advance to the procedural generation of scenarios, with parameters predefined by the player or developer, a customized and infinite game could be created almost as a professional designer would do.

4.3 Machine Learning as Neural Rendering

As our world becomes increasingly digitized, the methods by which we render these virtual worlds are rapidly changing. Neural rendering has huge potential in improving many aspects of the rendering pipeline by leveraging generative machine learning techniques.

Creating 3D virtual worlds today is a complicated and involved process. Each item, or asset, in a virtual scene is represented by a polygon mesh that is created manually. The more detailed we want this specific asset to be, the more polygons the mesh will have. Polygon mesh is only the beginning. Each surface in this 3D world also has a corresponding material,

which determines the appearance of the mesh. At runtime, the material and mesh of the object are used as inputs to shader programs, which calculate the appearance of the object under given lighting conditions and a specific camera angle. Over the years, many different shader programs have been developed, though the fundamental principle is the same: use the laws of physics to calculate the appearance of an object. This is most evident in the approach known as Ray Tracing, where every light ray is traced from its source down to every surface it bounces on.

With neural rendering, we no longer need to physically model the scene and simulate the light transport, as this knowledge is now stored implicitly inside the weights of a neural network. This means that it will be possible to render your face, while it is inside a VR headset without ever having to store or distort a 3D polygon mesh of your face. With neural rendering, the compute required to render an image is also no longer tied to the complexity of the scene (the number of objects, lights, and materials), but rather the size of the neural network (time required to perform a forward pass). This opens up the door for the possibility of really high quality rendering at a blazingly fast frame rate.

5.- Project Memory

5.1 Controls

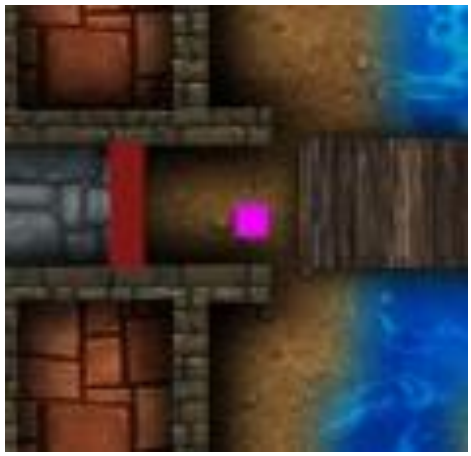
- Right mouse click for player movement

5.2 How it works

It is a demo in which AI techniques such as mastermind, decision trees, pathfinding, tracking, etc. are used.

The blue sprites are the agents moved by the AI. The agents move randomly, they chase the player if he approaches and return to their initial square if he loses sight of them. Each agent has a vision radius and when the player enters it, the agent chases the player. If the player leaves the agent's view, the agent returns to its initial position and returns to its normal state.

The player is the pink cube and is controlled with the right mouse click. The control is simple, right click on the position you want when you want and the player will detect the shortest way to get there.



If the player is caught by the enemy, the player teleports to the initial position.

In the ImGui menu, the "Coll Map" button switches the scene view between normal view and collision map view. In the normal map a map sprite is loaded. In the collision map, the squares into which the map is divided and the radius of view of the agents are displayed.

5.3 Project improvements

As a future implementation would be that the player himself moves automatically in search of an object or escape area and also analyzes the behavior of the enemy because if range of vision is greater, if the enemy goes to one side or has already traveled an area, he knows that this area for an undetermined time will be safer than a different area.

6. Bibliography

1. https://www.ui1.es/blog-ui1/machine-learning-y-videojuegos-futuro-o-presente

2. https://www.aecoc.es/innovation-hub-noticias/5-avances-de-la-inteligencia-artificial-que-probablemente-veremos-en-los-proximos-5-anos/

3. https://www.linkedin.com/pulse/deep-learning-ai-bubble-bursting-samer-l-hijazi/

4. https://spartanhack.com/machine-learning-y-deep-learning-todo-lo-que-necesitas-saber/

5. https://honorat79.wordpress.com/2019/05/30/redes-neuronales- artificiales-y-la-jugabilidad-emergente-el-futuro-de-los-videojuegos/

6. https://www.meneame.net/m/tecnolog%C3%ADa/red-neuronal-crea-render-3d-cara-partir-fotografia-2d-ing

7. https://analyticsindiamag.com/neural-rendering-computer-graphics-machine-learning/

8. https://www.rankia.cl/blog/analisis-ipsa/3617599-analisis-top-down-bottom-up-metodologia-ejemplos