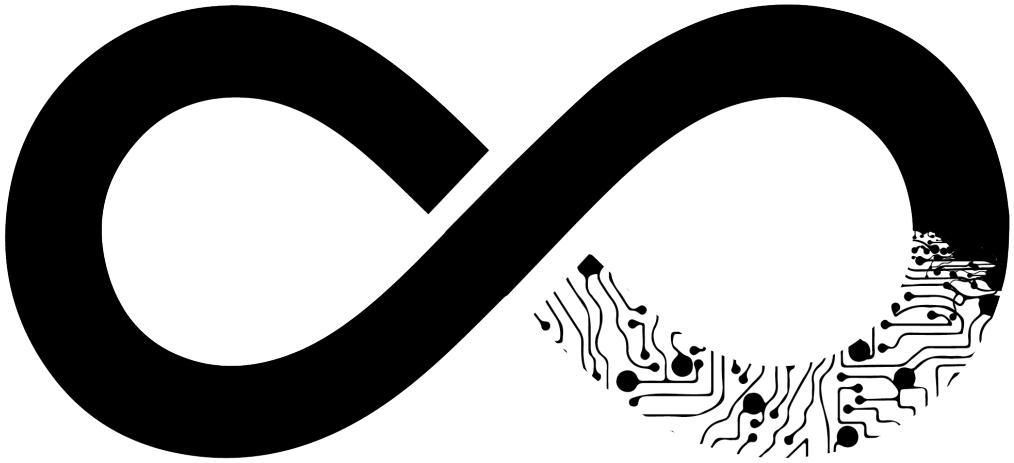
ynov CAMPUS



AYRTON

Machine Learning

Paul Bagnis
Owain Charlon
Dylan Levraud
Aaron Saksik

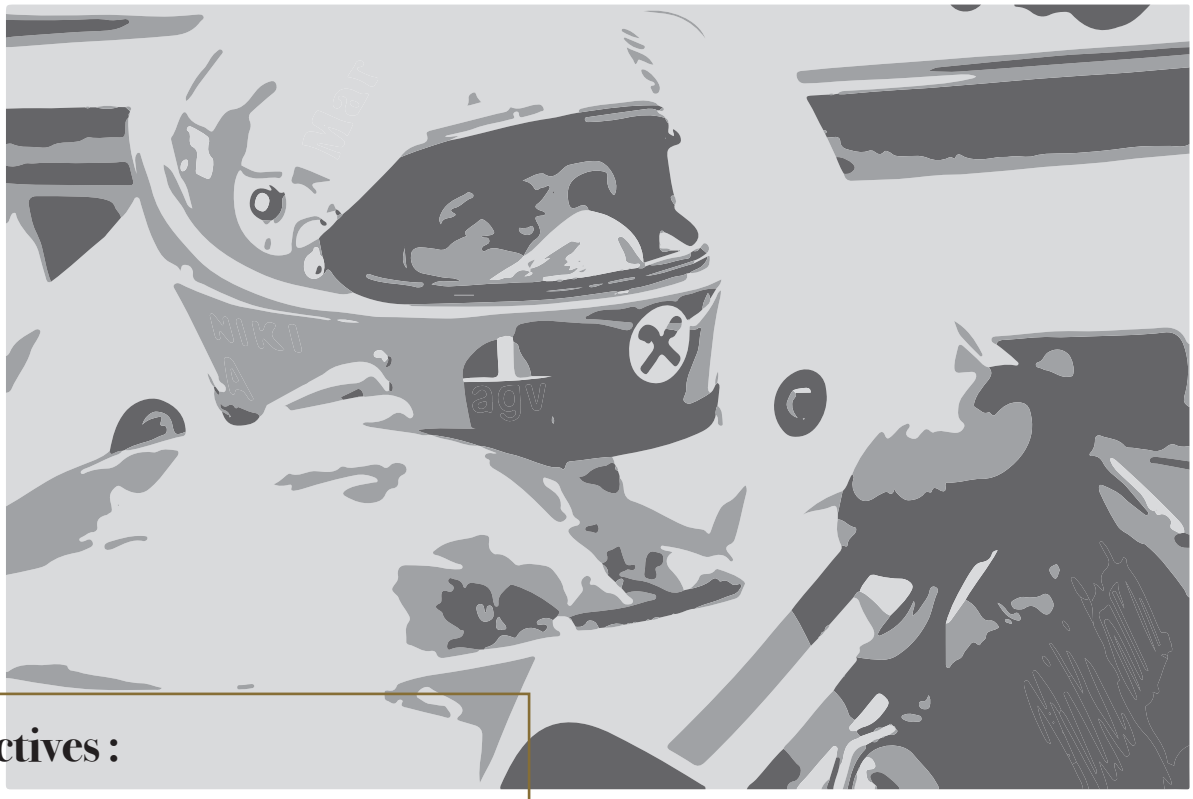
SOMMAIRE

AYRTON

PRESENTATION

DESCRIPTION

The project is an artificial intelligence named "Ayrton" capable of driving a car in a circuit. She will learn to ride the circuit following a "supervised training" by imitating the actions of a human "driver".



Objectives :

- The creation of the neural network of Ayrton.
- Creating a program is a smart car that can learn how to ride a circuit and cross the finish line.
- Creation of a 3D graphic environment (circuit, and car)
- Learn about artificial intelligence and develop knowledge in related languages. - Successfully communicate AI with a third-party development system (Unity)

Generate a database and use it for Ayrton's training.

PRESENTATION

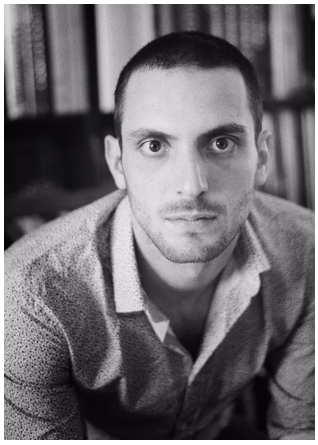
TEAM

MACHINE LEARNING SECTION

The members of this working group will work to develop and train the neural system of artificial intelligence. It will also have to interpret the results reported by the UNITY development section.

UNITY SECTION

This section is in charge of the graphical environment, designing objects and all associated C# scripts. It must try to return the information collect by car sensors in the Unity section.



PAUL BAGNIS

Passionate about artificial intelligence and its promise, he was keen to develop the neuronal part of the project.

Its strengths
Software development, patience, and calm.

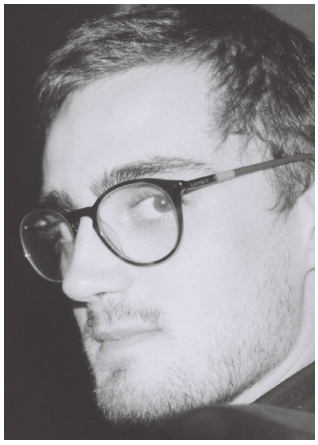
Weaknesses
Out of the school system for 6 years.

OWAIN CHARLON

Passionate about object-oriented programming, he wants to put his knowledge to use in a language that he greatly appreciates: the Python.

Its strengths
Object-oriented programming, seriousness and perseverance.

Its weaknesses
Annoyance, it still seeks itself through multiple experiences.



DYLAN LEVRAUD

Intrigued by video game programming, he wants to develop his computer culture by learning a new language: the C# and the Unity environment.

Its strengths
Management spirit, adaptability and assiduousness.

Its weaknesses
Takes things too much to heart.

AARON SAKSIK

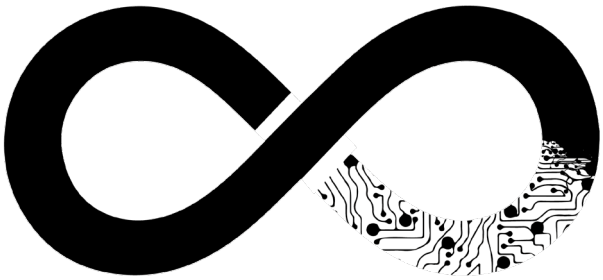
Driven by the desire to learn new object languages, Unity seemed perfect to respond to his ambitions.

Its strengths
calm, persistent.

Its weaknesses
slightly susceptible.

GRAPHIC CHARTER

LOGO



AYRTON

We wanted to represent a race track by an overturned eight. This device also recalls the idea of loops that run through our program. We've added a neural network to it to refer to artificial intelligence. Thus our logo symbolizes an infinite learning loop and reminds us of our concept of a smart car that has to cross the finish line of a race track.

FONTS

ADAM NORMAL

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
1234567890

We chose a typography named : Adam.
This typrograph has fine andclean lines. It conveys an impression of aerodynamics and lightness.

GRAPHIC CHARTER

COLORS

Our colors are based on the last car driven by Ayrton Senna. A blue, gold and black Williams.

R = 255	R = 134	R = 37	R = 33	R = 0
V = 255	V = 110	V = 53	V = 40	V = 0
B = 255	B = 54	B = 114	B = 76	B = 0
# = FFFFFF	# = 866E36	# = 253572	# = 21284C	# = 000000

AYRTON

BUSINESS PLAN

MARKET RESEARCH

First of all, we need to specify the scope of our market analysis, (what are markets, products range or brands that we are aiming) and our strategic business area.

The project

We are positioning ourselves in the video game market to offer companies in this sector the opportunity to outsource us in order to develop an artificial intelligence specific to their needs. Our product is based on Machine Learning, which allows a system to develop its «intelligence» and learn through a database used in supervised training. Knowing that this method is very complete and used for the creation of high-level artificial intelligence, we bring a reliable and innovative solution to this booming sector.

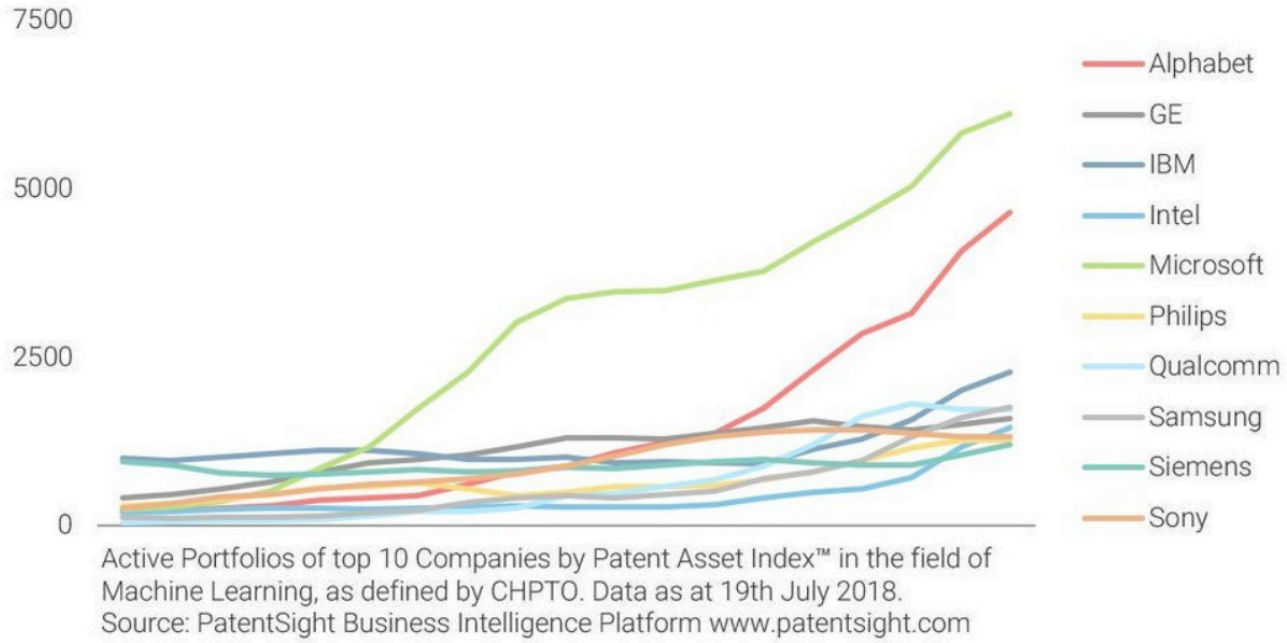
Competition

«Competition» is the rivalry between companies that operate on a market to satisfy the same needs of customers / consumers. The study of competition is at the heart of the market study. It influences the choice of the marketing and commercial positioning of the company. Direct competition is generally differentiated from indirect competition, even if this differentiation is entirely relative.

Direct competition

A direct competitor is a company or organization that offers a product or service that is similar or comparable to that of the company concerned.

Our direct competitors are therefore all companies that develop artificial intelligence based on a neural network for deep learning or machine learning.



Active Portfolios of top 10 Companies by Patent Asset Index™ in the field of Machine Learning, as defined by CHPTO. Data as at 19th July 2018.
Source: PatentSight Business Intelligence Platform www.patentsight.com

(https://www.houseofbots.com/news-detail/4231-1-top-25-machine-learning-startups-to-watch-in-2018-19).

Here we have all the most active companies in terms of machine learning that are listed. Most of these are companies that have some notoriety in a business sector other than AI but have decided to develop it.
Thus, many patents relating to this branch are regularly filed (we speak about 7000 patents filed by Microsoft), and this despite the youth of the market. This gives us a good indicator of the high level of competition present on it, since the main purpose of patents is to preserve an innovation and thus a certain competitive advantage.

Moreover, an article clearly submits that olipoly is almost inevitable if we look current market, to quote the article

"Getting an AI startup to scale for an IPO is currently elusive. Several different strategies are being discussed around the industry and here we talk about the horizontal strategy and the increasingly favored vertical strategy".
"In 2017 CB Insights reports that of the 120 AI companies that exited the market, 115 did so by acquisition. And the majority of those acquisitions were made by just 9 companies. Guess who."
"Thanks to the shortage of AI talent, most of these startups ended up being nothing more than acquihires. We hope that at least some made that life-changing money, but all those guppies ended up swallowed by whales and are now just features or products, not world changing businesses."

BUSINESS PLAN

MARKET RESEARCH

"The core concept of horizontally strategy is to make an AI product or platform that can be used by many industries to solve problems more efficiently than we could before AI."
"Then came what VCs refer to as the monoliths: Google, Amazon, IBM, and Microsoft. As we now know, their dominance in their respective areas of data gave them an almost insurmountable."
"There's a second tranche of horizontal competitors below the monoliths including startups. These reach all the way down to some of the newer automated machine learning (AML) companies like OneClick.AI who integrates deep learning with the standard assortment of ML altos."
"The vertical strategy isn't the only alternative but it's certainly the most widely talked about these days. It shares the concern over data dominance with that strategy but goes further in specifying other aspects required for success."
That kind of strategy also impacts on the rise the competition in the market, increasing the weight of compagnies which are already well inserted.
Beyond the oligopoly situation, previously developed the creation of startups impacts completion as well between compagnies on the IA market.
Indeed, CrunchBase lists **over 5,000 startups** who are relying on machine learning for their main and ancillary applications, products and services today.

(https://www.cbinsights.com/research/artificial-intelligence-top-startups/)

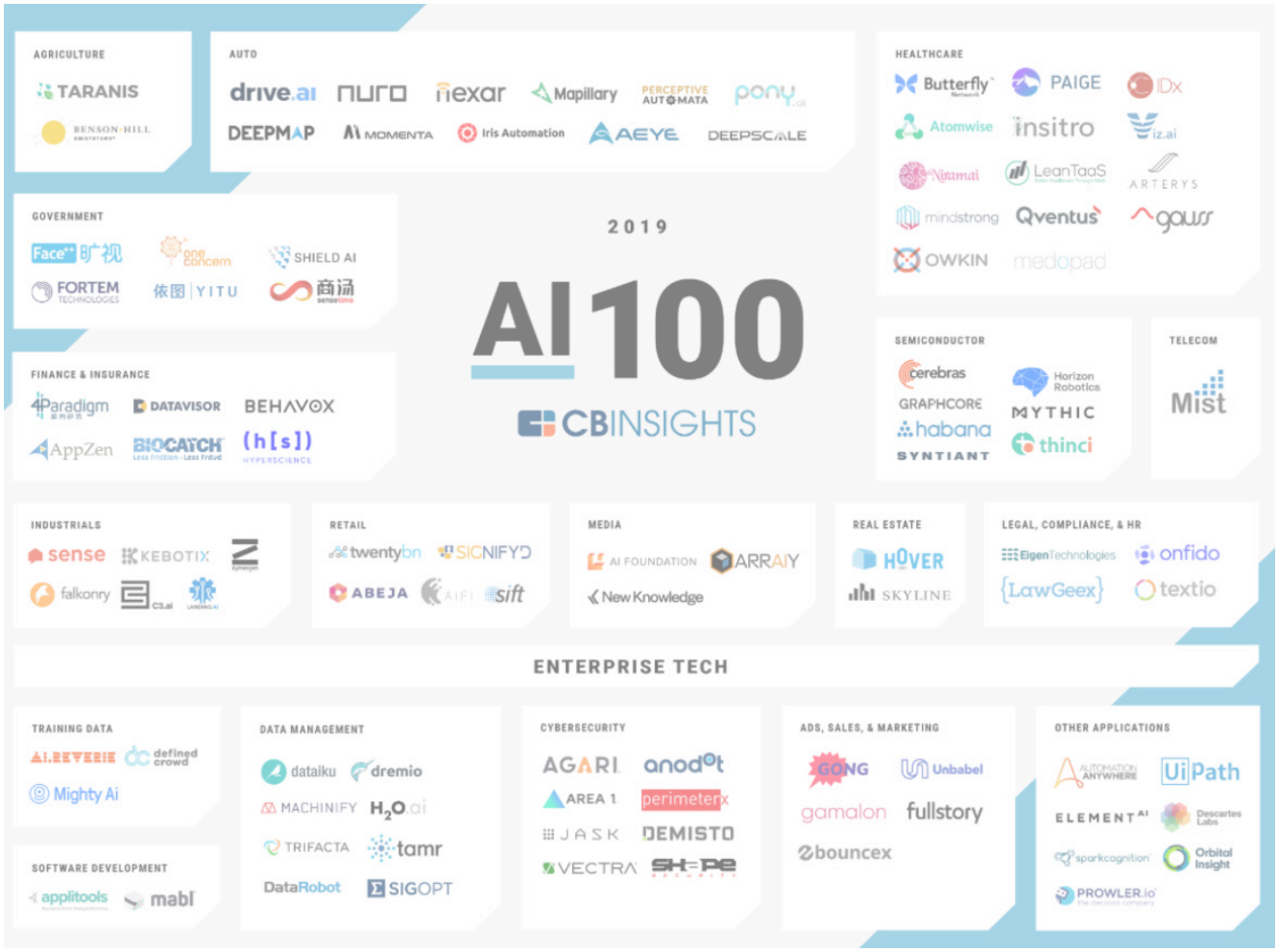
Indirect competition

An indirect competitor is a company or organization that offers a different product or service, but that is likely to meet the same consumer's need.

Our direct competitor includes all companies which develop AI, in the branch of Deep learning/machine learning or else.

The diagram below refers to many companies created in 2019 in the AI world. We can easily detect that many companies are emerging in this area and those within multiple sectors.

This brings a new element as to the level of global competition in the artificial intelligence market, and therefore to the indirect competition to which our product could be subject.



https://s3.amazonaws.com/cbi-research-portal-uploads/2019/02/01125325/AI-100-2019-2.png

AYRTON

BUSINESS PLAN

MARKET RESEARCH

The scope of our market analysis (what are markets that we are aiming)

There is a lot of business based on AI (artificial intelligence), including Google, Amazon or Tesla ...
This domain is very extensive and without limit whatever the utility or the reason of its use.
All these companies approach it in different aspects for different designs, take as an example Google or Amazon who developed an AI for a voice assistant (Google Home, Alexa).
AI is simply the future at all levels, for all people who are likely to use new technologies and around the world.
This technology has been around for quite some time but is still growing every day by offering better realizations for customer demand.
Our company is the perfect link between these two sectors: a specific AI for video games, specifically an AI on video game vehicles.
We are binding these two world economic powers to develop and commercialize this project.

That is why we decided to base our analysis on the IA market and the Video games market. We will start by focusing on the French market, our starting point.

The IA market

We can define the AI Market as all products that are primarily related to

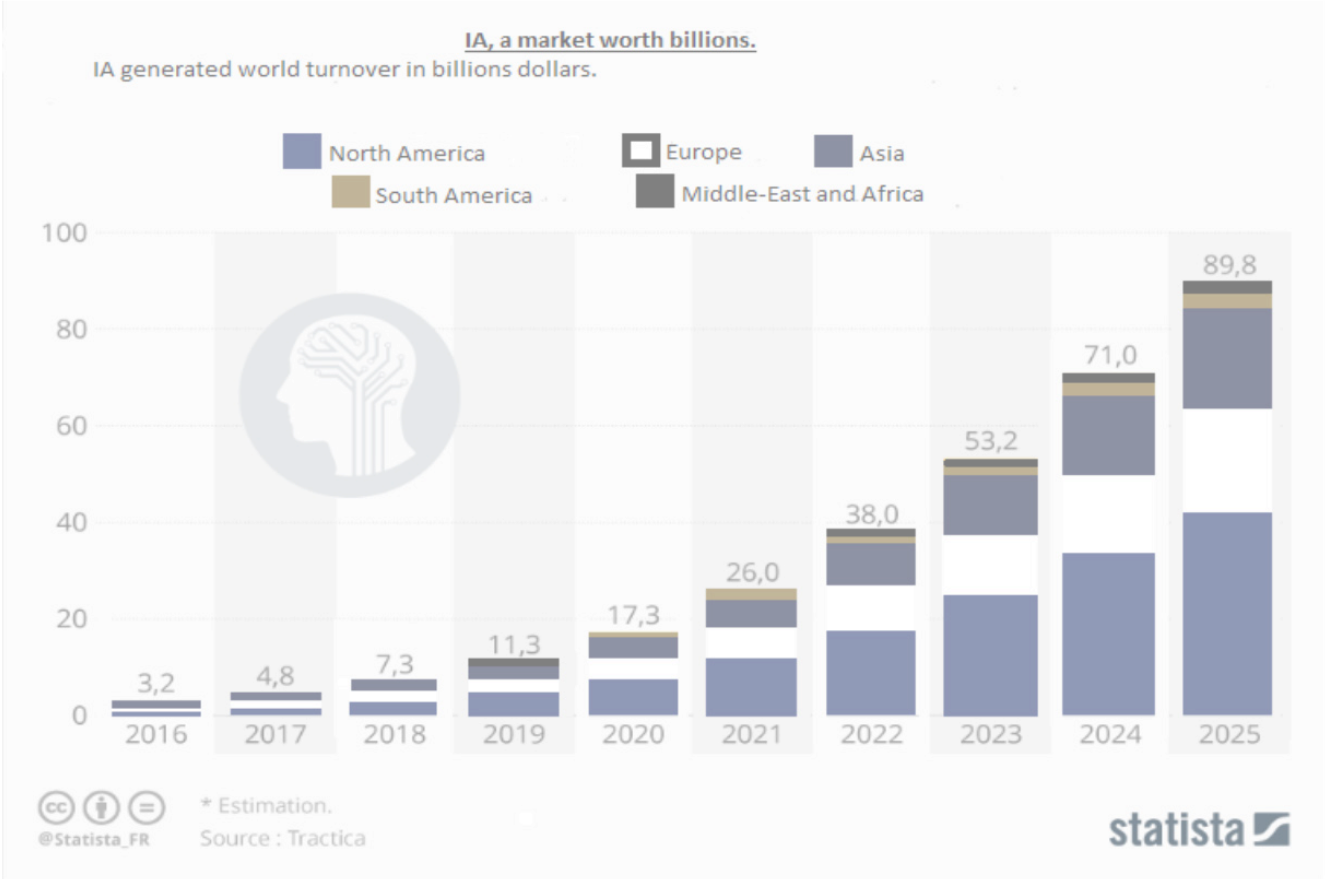
- Big Data which is about harvesting and processing large amounts of data.
- Machine learning/Deep learning two thrust and widely used data analysis techniques

On French's level

- The French IA market is estimated at 11 billion euros in 2024 while it weighed 200 million in 2015.
- 950 start-ups worldwide are active in the field of AI (including more than 270 in France -> 29%).
- More than half (52%) of French companies are sensitive to AI with an average annual budget of 576 000 € (TCS study).
- The growth of this market (platforms, software and related services) was about of 40% between 2016 and 2017.And it has been more marked in 2018, with growth of 49%. (IDC study).

On world's level

- The market will continue to see significant investment over the coming years, reaching an annual growth rate of 54.4% through 2020, for a market that will then represent more than 46 billion of dollars in revenue.
- In the first quarter of 2018, $ 5.42 billion was invested in AI (knowing that 78% of investments in this nascent market are in the United States).
- In terms of investment location, the EMEA region (including Europe, the Middle East and Africa) is currently in second place, but it should be supplanted by Asia-Pacific by 2020, especially due to the explosive growth of investments in Japan (+ 107% average annual increase over the period) ...



(https://fr.statista.com/)

AYRTON

BUSINESS PLAN

MARKET RESEARCH

Here we can see that the global turnover for the AI market is in full bang. If we take the numbers, the latter has been multiplied by 30 in less than 10 years. The global market is very profitable especially in countries or innovative geographic area (such as North America or Europe).

There is therefore a market attracting investors for its profitability, allowing a large fluctuation of capital, promoting in turn innovation and the development of new technology, a very successful environment. We are talking here about a real economic El-Dorado.

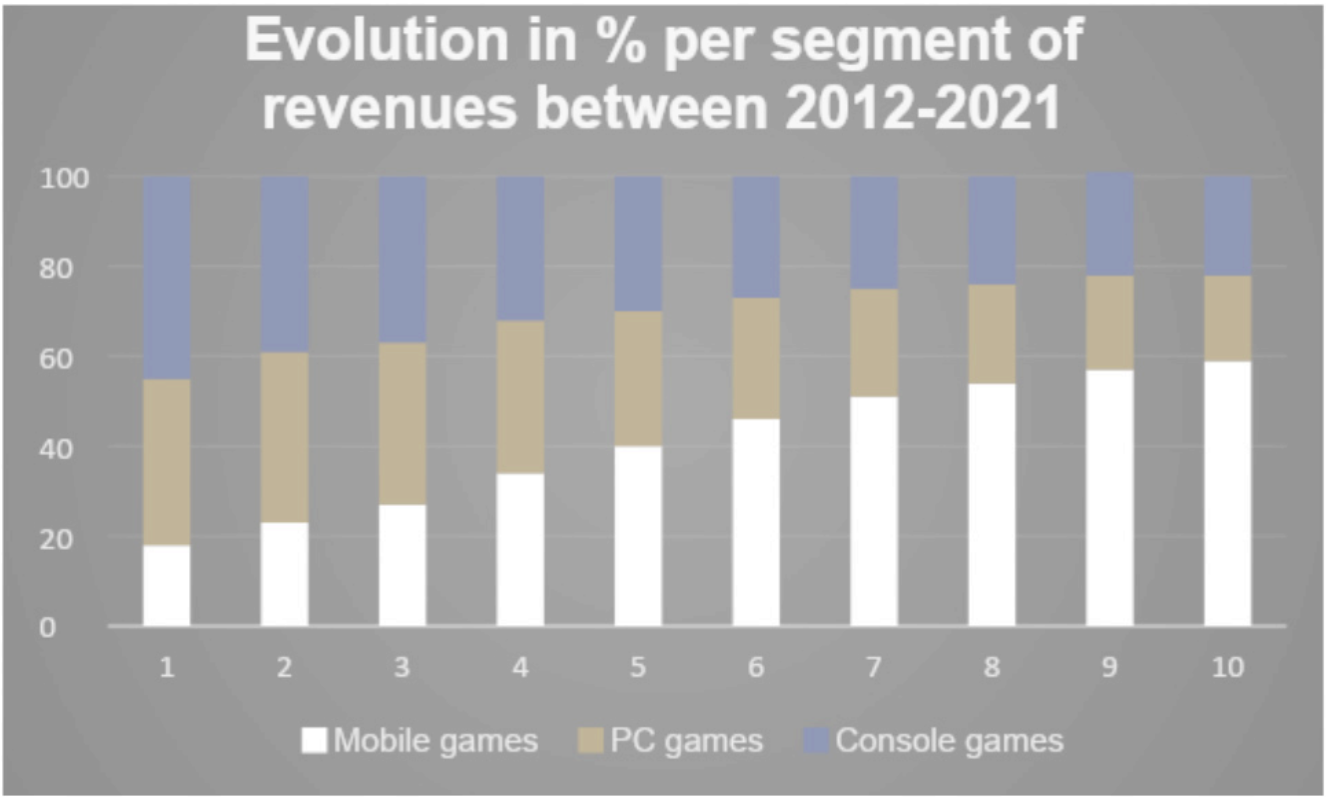
The Video Games market

French video games market

- **In France**
  The video game reaches new heights in France. The market reached a new record in 2018 with a turnover of 4.9 billion euros, up 15% from the one generated in 2017 (4.3 billion). The console ecosystem is the strongest with 2.75 billion euros (+ 15% in 2018), ahead of the PC gaming ecosystem (1.24 billion, + 10%) and the mobile ecosystem (946 million, + 22%).

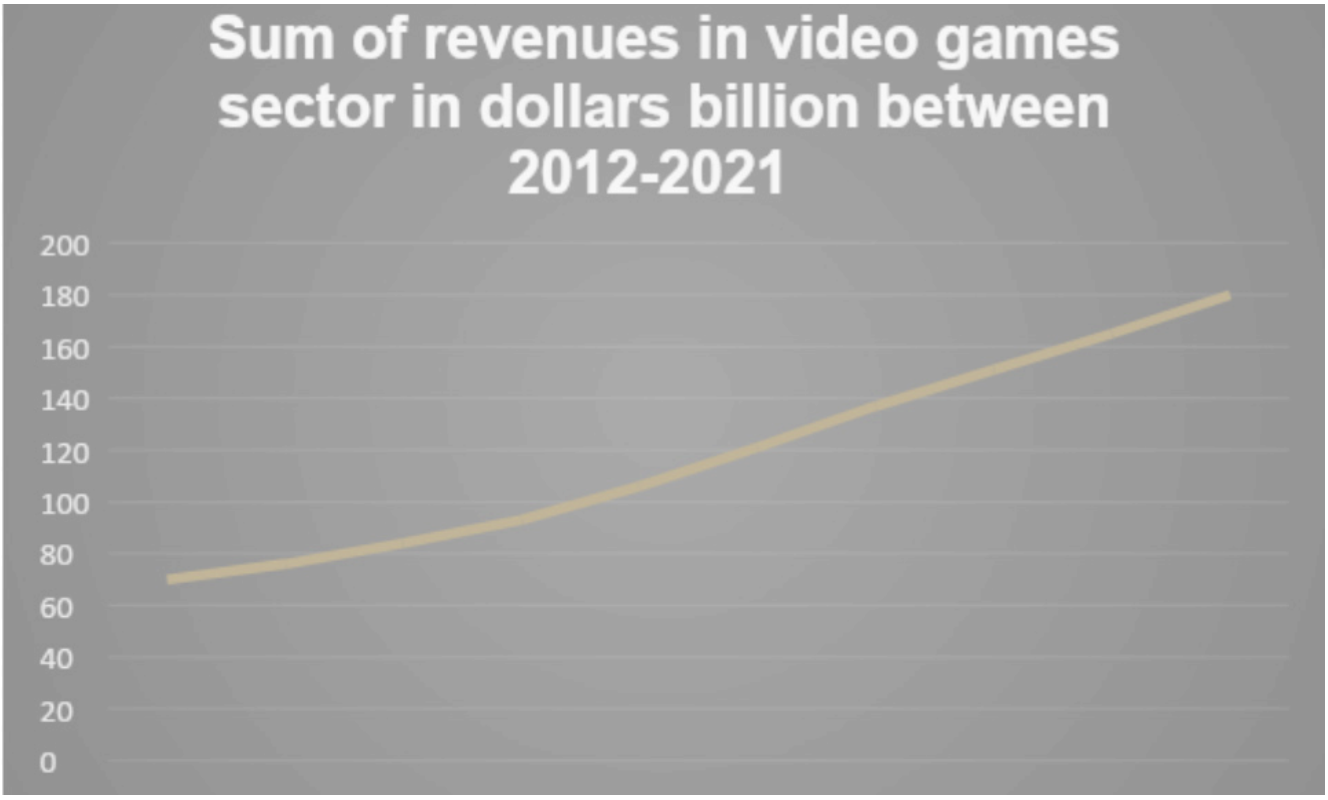
- **At the world level**
  A 138 billion turnover for 2018.

The video game industry has grown on several platforms in recent years, which consists of three categories, mobile games, PC games and console games. In particular, we see a clear advance for mobile games like the other two categories (with a 10% to 60% increase between 2012 and 2020).
It should be known that our product can impose itself in the video game market whatever the platform. Indeed, the graphic engine Unity is based on a compiler language, it only requires an interpreter in order to run, this is considered a real asset for our product.



(https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/)

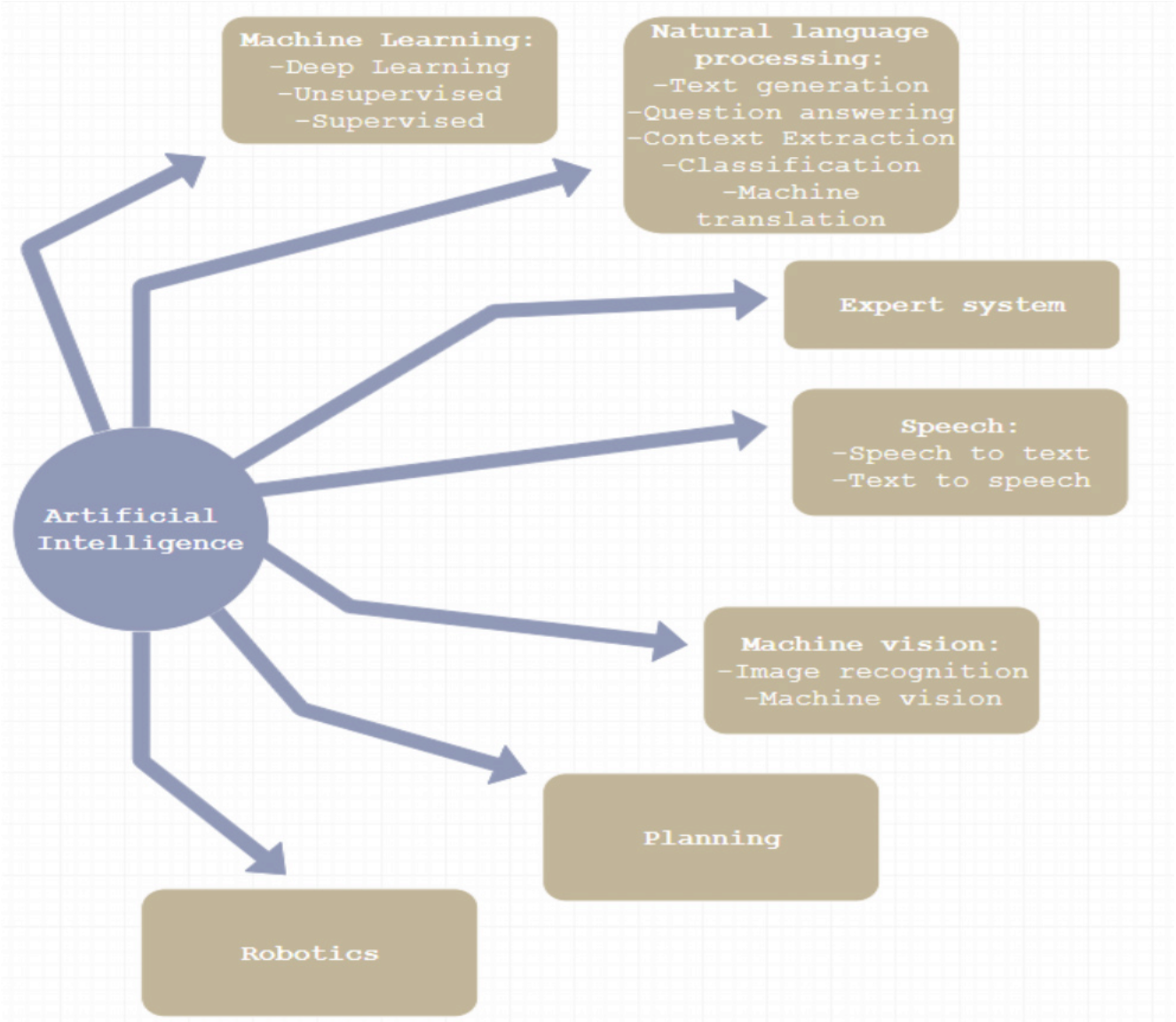
MARKET RESEARCH



This second graph describes the evolution of the turnover of the entire market between 2012-2021. Over the period studied, the overall turnover of the video game industry has more than doubled, from $ 65 billion to $ 190 billion.

Our strategic business area

We knew that artificial intelligence was already used in many sectors, but new opportunities continue to emerge for this very malleable technology.
Below is a summary (that we made inspiring articles) of the main and varied tasks performed by an AI, we can notice that these are potentially related to distant sector of activity branches. We can therefore argue that the IA has an infinity of opportunities whatever the branch and sector studied.



AYRTON

BUSINESS PLAN

MARKET STRATEGY

Principal Objective

Successfully outsource the development of driving AI in video games.

SMART Methodology

S	M	A	R	T
In the short term Propose to outsource vehicle AI to gaming companies specializing in racing games. **In the long term** Subcontracting the entire AI of a company whose project is an Openworld game (type Ubisoft, Rockstar, Konami)	Result from the validation of Benchmark Learning/auto-mode.	**Projection 1 year** Get 1 contract for the package. **Projection 3 years:** Obtaina subcontract **Projection 5 years:** 1open-world project	**Our skills and competencies** **Over 1 year period:** Able to develop a stable version. **Over five years:** Able to evolve in any graphic environment	By 2025, we should have signed 5 contracts with large companies.

Target Marketing

Target	CONSIDERATION OF PURCHASE	CRITERIA OF DECISION
Short term Small and medium-sized enterprises. It seeks labour and time savings through subcontracting to meet their deadlines imposed by their structure.	Affordable developers. Innovative ideas. Technical expertise.	Low cost. Reduced team.
Medium term Mobile game development company (Gameloft)	Time saving. Need more developers. Short project	Fast labour requirement. Fast delivery.
Long term Large AAA games company (Ubisoft, Arkame Studios)	Expertise Trust Higher Pricing Long-term project	Competitive value Superior experience

BUSINESS PLAN

MARKET STRATEGY

Customer purchase path

Awareness

After learning about the constraints of his project, he is proposed to outsource part of his project and a development in agile method.

Consideration

He says no, he recruits himself or else accept our offer and focus on another point of his game. Every 3 weeks it will have a detailed report of the evolution of the AI. With regular deliverables, a permanent look on the subcontracting

Decision

Cost and expertise. With a fixed-price project, if dissatisfaction with the deliverables, the customer only charges for what is useful to him. Reduces financial risk on its side.

Products solutions

Our system is based on an overall consideration of the environment because we are putting in place a neural system that takes into account hundreds of trees.
This AI solution will provide an ultra-realistic video learning experience.

SEO reference

- Artificial Intelligence - Technical expertise - Open world
- Outsourcing
- Ayrton
- Experience
- Quality

What can we recycle ?

Our Ayrton starting neuronal system can be applied to each customer command in different forms. As projects progress, our teams contribute to the continuous improvement of this artificial intelligence and its optimization.

Understand the customer request

Understanding customer demand and the related environment is fundamental to enable our experts to integrate perfectly in-house. The following questions will therefore be asked.

Customer Questionnaire

- What do you want?
- For what type of game?
- How much work does the project require?
- What biases of level design and game design.
- In what kind of environment does the action take place? - Frequency of deliverables?
- deadlines?
- The problems they have to solve in their programming

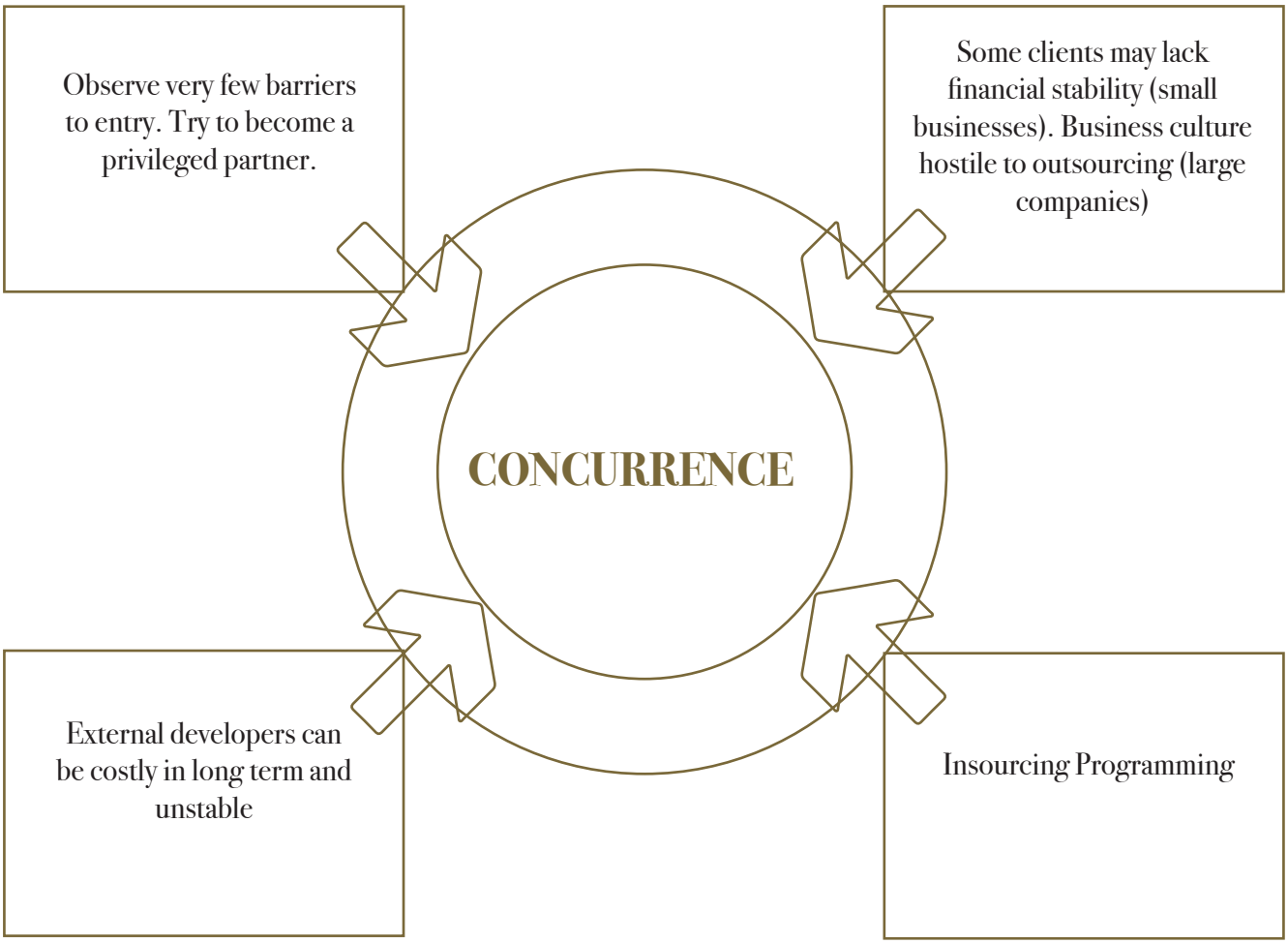
BUSINESS PLAN

MARKET STRATEGY

Observation of competitors

The competitors are the companies themselves.
Our project is part of a Blue Ocean strategy. So no direct competitors. But there are substitutable offers : the companies have the possibility and are already doing their own programming. It induces nitre narrowing of our field of action. Our competitors are initially our targets.

Porter Forces



Measuring Results Achieved

Our KPIs focus on several strategic areas

- Number of contracts in progress and completed - Frequency of contracts
- Compliance ratio of deadlines
- Customer Satisfaction

BUSINESS PLAN

COMMUNICATION STRATEGY

The purpose of this communication plan is to make us known to the various companies of video games creation in order to submit our services and develop our activity.
It therefore aims to raise awareness of our services and the added value we can bring.

First, artificial intelligence (AI) and video games are two sectors that have evolved closely. The video games represent a very good ground for the development of AI, this through chess games or Go.

Since then, AI has always been used to make video games and their virtual environments ever more natural, more human, in order to increase the player's immersion.

The main challenge of this alliance is to make AI's allowed resources efficient. In fact, in the context of AI research, almost all the computer's resources are allocated to the operation of the program. However, in the context of a video game, AI is only one aspect of the program, so the question of the efficiency of its implementation is much more important.

Today, the power of the machines has evolved well, the place and resources that an AI can take in a game are much more substantial and allow the implementation of much more complex and efficient versions of it.

In addition, the explosion of Virtual Reality (VR) puts artificial intelligence under the spotlight, immersion being at the very heart of the development of programs dedicated to VR.

Our main objective is to make ourselves known to video game development companies. It is therefore a communication from professionals to professionals (B to B), where we want to make ourselves unavoidable in terms of adapting artificial intelligences to video games.

These compagnies are the ones to whom we can bring our know-how in order to enhance their game design work by further immersing the player.

We will therefore have to approach them in a professional context with concrete examples of what we can bring in terms of "realism" of our artificial intelligences.

In order to do so, we need to position ourselves as a company that can magnify the work of game designers and level designers. It is in the details that a game becomes immersive, one of them is artificial intelligence whether it is of Non-Player Characters or of the evolution of a game environment.

The goal of our campaign will be to show how we can make virtual environments come to life.

BUSINESS PLAN

First, our target being professionals, our objective is not to make ourselves known to the public at first. Our mission is to work in the shadow of video game development houses. We are a service company on one of the aspects of the game being developed.

It is therefore logical to go to their meetings through the various fairs dedicated to professionals in the sector.

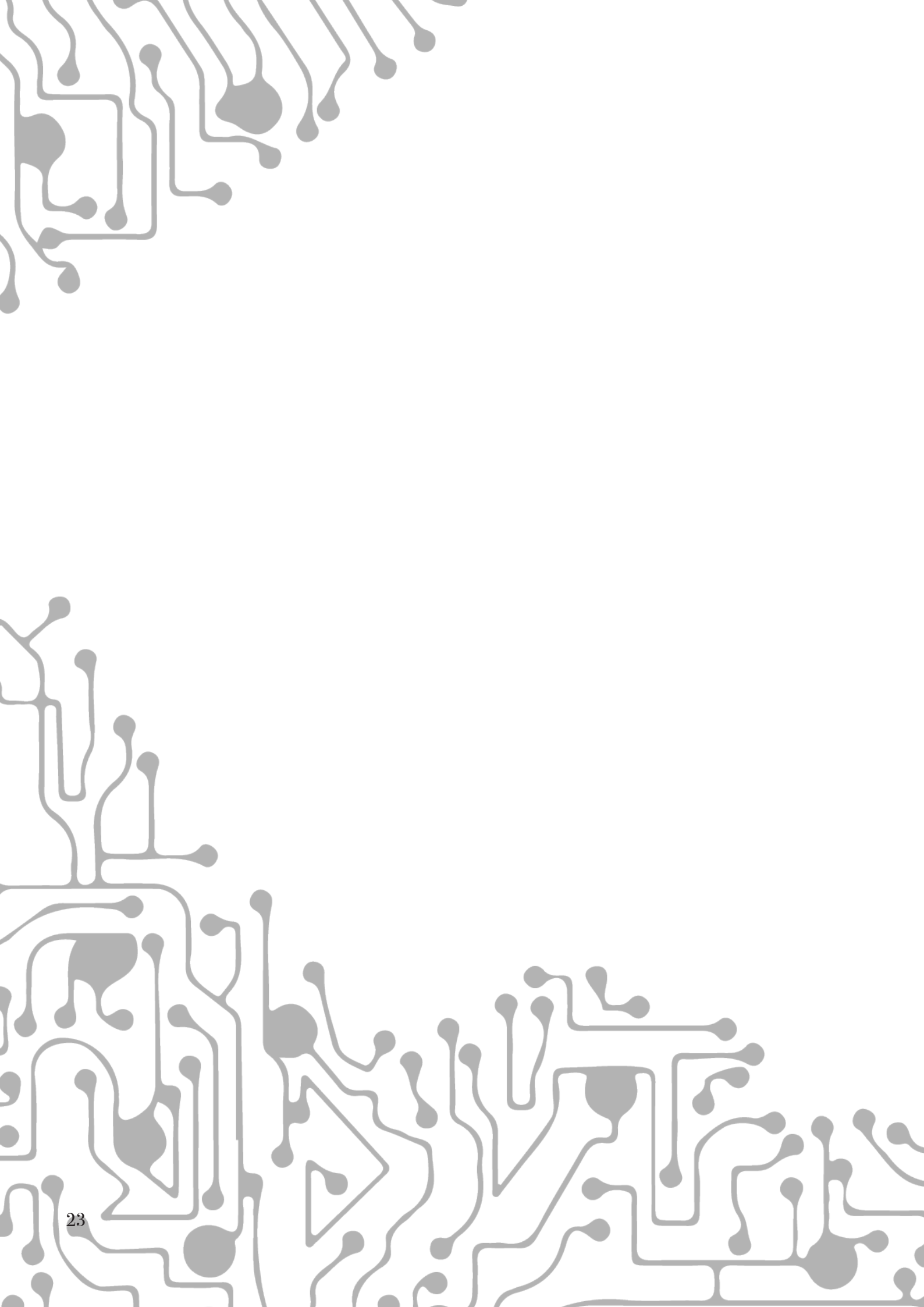
An example of such a show would be the IDEF 2019, which will take place in Juan les Pins from 1 to 3 July. Organised by the Syndicat des Editeurs de Logiciels de Loisirs, it is a place bringing together video game development houses and emphasizing the technological advances of the sector. With more than 500 participants, this type of event will allow us to make our services known, in addition to having a direct relationship with content creators.

That would be our primary communication vehicle. The second is to physically canvass these same companies through physical encounters where we could expose them our services and the added value we could bring.

In a second step, the organization of events around artificial intelligence in video games would be a good way to attract the attention of companies of the sector and to establish ourselves as a major player in the market.

“Perfection is in the details”. Who has never heard that phrase? That’s the message we want to provide. Artificial intelligence is a tool that, although not major in the creation of video games, has a real impact on the user’s experience. What’s the point of creating a great world if the player can’t feel it live? The services we offer are there to partially solve this problem.

Moreover, the fact that artificial intelligence is only a part of the creation of a game, allows us to logically propose an outsourcing of its development process. Companies having a hard time keeping specialized developers in their structures occupied.



STATEMENT OF WORK

Our world attends the explosion of the virtual hobbies. As such, the sectors of the video game and the artificial intelligence are in full bang.

The sector of the video game, ultra-competitive, is in height chase of realism and immersion. Between this growing race and the power of computers of the users, the systems of artificial intelligence take more and more place in the needs of the houses of video game creation. Previously, they were limited by the power of computers and the optimization of the algorithms. Today, more and more complex systems are integrated into the sources programs.

The idea being to propose an outsourcing solution of the creation process of the artificial intelligences, we can propose a personalized solution realized by developers specialized in the domain.

From an economic point of view, the digital technology generally is in constant progress, pulling in its trail most of the numerical domains. The period is convenient to the setting-up and development of companies specialized in the field of the artificial intelligence. Companies trying to do always better in this domain.

The neuronal systems of machine learning and deep learning are more and more fashionable. Therefore, Frameworks facilitating the implementation and the training of neuronal systems evolves constantly to allow a better optimization of the training and the resources dedicated to the optimal functioning of the system.

The organizational needs for our company are relatively limited. In terms of staff, we are a team of 4 developers and will proceed our self to the design and the realization of the artificial intelligences. We shall need offices as well as computers to finalize our solutions.

The realized neuronal systems will aim at making more human the characters not players in the video games. We propose an outsourced solution, companies can subcontract these systems. We are able of proposing various technologies in terms of learning according to the desired result. For example, we can use the training overseen to make an artificial intelligence which can evolve according to the actions of the players. The speed of learning of the IA will also be one of axes of discussion to adapt at best our solution to the needs for the customer.

We shall be present also during the implementation in the virtual environment to make sure of its running smoothly. Also, we shall stay at the disposal of our customers in case of evolutions of their games to adapt our solutions.

CONCEPT NOTE

We submitted the idea of a machine-learning project. We wanted, during these Y-days, to introduce ourselves to a field that we did not know: artificial intelligence. Artificial intelligence will play a major role in the very near future. France, the great representative of the world's research, must do its best.
We must understand the ins and outs of this new form of computer science. The resources at our disposal are numerous and the fields of research are consistent.
The field is full of topics to be covered, ideas to be refined and projects to be implemented.
On our scale of students in the first year of computer science, we must relativize the fields in which we can evolve and try to operate in a field where our skills, still to be perfected, may be useful.

The video game industry is constantly growing. We think it's a good idea to move into that sector. As large consumers of video games, we have seen a great technical evolution, both in terms of graphics and gameplay.

The Next-Gen was the most important graphic advance. However, overall, the artificial intelligence of these games did not have the jump it deserved. Some games have been successful in "playing" like GTA. But a handful of titles can claim to possess such automation of their environment. Many games, open-world oriented suffers from a disappointing AI.

It seems to us that one of the greatest video disappointments is the conduct of the characters not playable. Blockbuster intelligences like Watch Dogs 2, Mafia 3 or Just Cause tend to show that the world of video games needs to focus on driving automation. A lots of critics was made in the specialized press about this kind of problems. Current AI is based on predefined patterns. Our tends to learn from itself.

So we want to look at this case and move towards arti cial intelligence capable of fully understanding its environment to improve the player's experience. This is thanks to a learning strengthened by man. To achieve to move smoothly in an open world.
An intelligence that, once trained, would no longer need the help of man to interact with the vastness of an open world.

To achieve such a result, we must rst start with something that is technically affordable.
During our discussions and research we decided to move towards the automation of a virtual car in a program. The project therefore takes the following form:

A car must nd its way through a prede ned circuit. The program must ensure that the car crosses the nish line through enhanced learning. This learning is done by a neural network, which combines mathematics and computer code. This part is developed on Python. It must try to inform arti cial intelligence of the actions of a human being who moves the car. This one will analyse them and train to a smooth course.

This part is complicated by our lack of understanding of the process. We must learn everything to succeed.

This neural network is nothing without a de ned environment. The creation of a 3D world and a race track is therefore fundamental. Unity is a software, with a large and accessible graphics engine. It therefore appears as a rst formatting exercise.

So after the creation of this world, it is necessary to model a car and all the attributes that compose it (acceleration, direction, speed, etc.). It is here that the C# language, an object-oriented language, will enable us to achieve our ends.

This part is tough, the creation of the environment will be done without 3D animator and we will have to understand the overall functioning of Unity and C#.

But once the data of the neural system and the interpretation of Unity are linked, the car will meet our expectations: autonomy.

It will be a small step towards more substantial projects

DIFFICULTIES

DIFFICULTIES MACHINE LEARNING

The main problem met during the development of the neuronal system was the adaptation of the code realizing the back-propagation and more exactly the recovery of the digital gradients.

Most of the documentation on which we spent time at the beginning of our project explained how to realize a neuronal system with one output. However, we had thought of realizing our system with two exits managing 4 possible orders (accelerate, brake, to turn right, to the left). Assuming that one of the neurons of the output would take the values :

- 0 for braking
- 1 for the acceleration
- 0.5 to do nothing

However, we realized quickly that it wasn't going to work. The values of activation of neurons wasn't enough precise for it. They correspond more to probabilities between 0 and 1.

Of more, the dismissal of a tuple in the form (O1, O2) complicated enormously the calculation of the loss function and of the back-propagation. We did not know if we had to make the average of the return of the loss function for each of the elements of the tuple or handle each of the losses in an independent way by echoing them each on the system. We did not manage to find a calculus which seemed to us coherent or logical mathematically.

Thus, it was necessary us to realize either a complex system gives 4 outputs or plan a system by possible action. We finally opted for the second solution. However, less optimized in terms of allowance of resources. Particularly during the phase of training of the system, where we had to train 4 systems at the same time. Our level of knowledge on the subject made us choose the second solution.

Second major problem we met concerns more the theory. Indeed, the implementation of these neuronal systems asks for mathematical knowledge we did not have. Thus, it was necessary to form us in the matrix calculation and in the partial derivatives. We did that through Internet, with explanatory videos and the comeback in (old) courses of final year of high school.

DIFFICULTIES

DIFFICULTIES UNITY

During the implementation of the Unity part, various difficulties arose. First of all the lack of knowledge of the Unity environment. It had to be tamed through the tutorials to try to understand physics, graphics, the notion of 3D objects and all the constraints that this generates. It is after several uses that Unity reveals a consequent. field of possibilities.

The creation of a terrain was carried out without any real difficulties, thanks to the ease of the tools at our disposal (the tool Paint Brush, Grower/Lower Terrain). However the creation of the track was much bolder. It was necessary to understand the rotation 's system of objects and how they evolve in the scene. This part was pretty hard.

Indeed, create a replica of the Imola circuit was not entirely restful and imposed many technical constraints that resulted on adjustments of the track. The track is not the perfect copy but tends to get as close as possible to it.

Subsequently the code part was the most difficult. We had to learn a new language: the C#. Having had courses of C at the beginning of the year, the C# is very close to it but includes a significant number of pre-defined functions that must be understood in its entirety in order to be able to use them to their maximum capacity. Thus much of the time was in reading the documentation C# Unity. Creating the movement of the car was tedious. Two notions were to be understood: wheelCollider and quaternion. We also had to learn about the rotational forces we can exert on the car to move it.

As a result it was necessary to adjust the driving by testing various weights, acceleration speed and braking systems.
The most complicated part was the creation of sensors around the car. It combines position, Drawline, maths and distance. This is all in order to recover distances between sensors and colliders. This is the part on which we most needed help. Given the busy schedules of the Unity specialist responders, we had to hold on to achieve our goals.

The audio management was quite complicated, the sound coming out of the car must interact with acceleration or braking, because depending on the kept key different noises are turned on or off.
For the music of the Menu a Slider is put in place to manage the volume of it as desired and it was not easy to make the connection between these two elements.
The only problem with the animation of the camera around Formula 1 was making it possible to start the Game by stopping the video animation immediately.

Time management was also tough, and we were unable to link the Python and Unity code.

SOURCES

MACHINE LEARNING PART

http://blogs.wefrag.com/drak/ -- **Article 2**

https://github.com/ArztSamuel/Applying_EANNs https://www.youtube.com/watch?v=rEDzUT3ymw4 https://arztsamuel.github.io/en/projects/unity/deepCars/deepCars.html

Tuto Machine Learning and Neuronal Sytem

Ep 1 : https://www.youtube.com/watch?v=aircAruvnKk&t=997s

Ep 2 : https://www.youtube.com/watch?v=IHZwWFHWa-w

Ep 3 : https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=646s

Ep 4 : https://www.youtube.com/watch?v=tIeHLnjs5U8

https://www.youtube.com/channel/UC0e3QhIYukixgh5VVpKHH9Q/videos https://www.youtube.com/watch?v=r428O_CMcpI

Biblio python deep learning

https://www.tensorflow.org/

Unity et Deep Learning

tuto: https://www.youtube.com/watch?v=OTMlElDuN3k

Deep learning tutos

http://lumiverse.io/series/neural-networks-demystified

SOURCES

UNITY PART

Learn Unity3D et le C#

Chapter 1

EP1 interface= https://www.youtube.com/watch?v=MXcJOVucEls

EP2 physics = https://www.youtube.com/watch?v=EINEm5agfgs

EP3 graphics= https://www.youtube.com/watch?v=K6YIyvK9qjE

EP4 animations = https://www.youtube.com/watch?v=gF0FYjz0jgU

Chapter 2

EP1 Variables and fonctions = https://www.youtube.com/watch?v=jNCK4-L6q3I

EP2 conditions = https://www.youtube.com/watch?v=pwA-uNVxMNM

EP3 loops = https://www.youtube.com/watch?v=O6mVdtR5RK8

EP4 GameObjects = https://www.youtube.com/watch?v=0xOkqy26tt8

EP5 Objects = https://www.youtube.com/watch?v=HOhOGS9FYOU
Documentation Unity C# : https://docs.unity3d.com/Manual/index.html

PROTOTYPE

MACHINE LEARNING PROTOTYPE

Creation of a virtual environment

First, we could have used a high-level framework like TensorFlow to quickly build a complex model using a function kit. However, we wanted to understand in depth how do IA really works and learn.

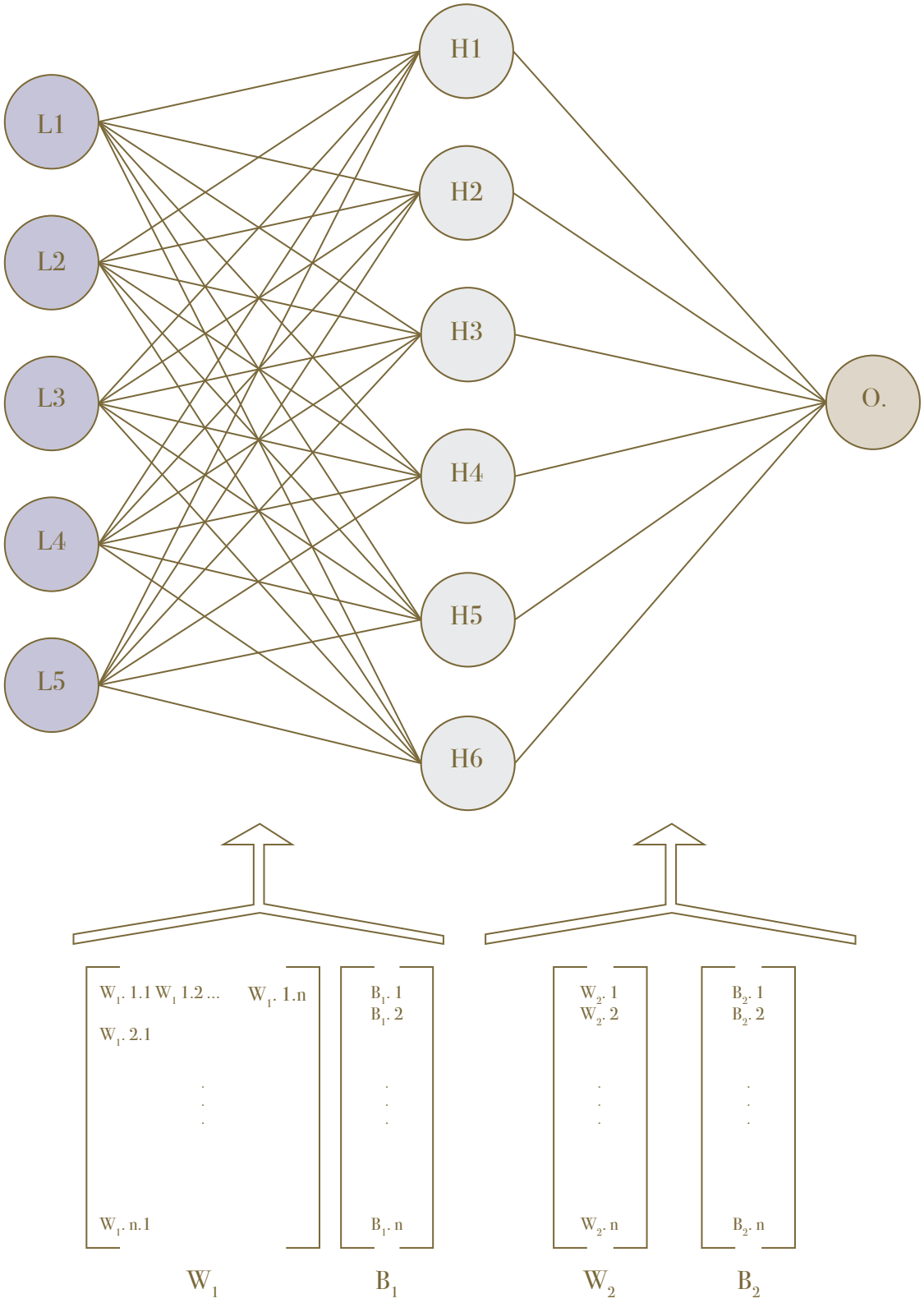
So, we decided to build a fully operational neural network using only NumPy, the fundamental package for scientific computing with Python (mainly used in our project for matrix calculations). Moreover, it was too difficult to develop a double outputs neural network. This is the reason why we made one network to control car's direction (left, right) and another one to control car's acceleration (acceleration, break).

Before we started programming, we prepared a basic roadmap of our neural network (see sketch after). We had to think about a program capable of creating a connected neural network with the desired architecture (number and size of layers and appropriate activation function). Indeed, the choice of activation function was important and one of our priority, because it allows us to fix the accuracy degree of trainings and the system's efficiency.

Here are listed operations which will have to be performed during the training of Ayrton

- Data generation, that will be the base of test for Ayrton
- Each time we recover data, they will be transmitted to Ayrton
- Data will process into Ayrton's neural network
- Ayrton's output will be compared with the results obtained with the recovered data from human driving
- We will estimate the error between Ayrton's answer and the human's reaction (Loss function)
- The error will be back-propagated into Ayrton's system to calibrate the weights and bias of the system and make it more precise

PROTOTYPE



AYRTON

PROTOTYPE

MACHINE LEARNING PROTOTYPE

Initiation of neural network layers

We started by initiating the weights matrix W and the bias vector b for each layer. Each item in the list is a dictionary describing the basic parameters of a single network layer

- input_dim: the size of the signal vector supplied as an input for the layer
- output dim: the size of the activation vector obtained at the output of the layer
- activation: the activation function used inside the layer.

```
class Neural_Network(object):

# Création de l'architecture du systeme neuronal
NN_ARCHITECTURE = [
    # Ici «input_dim» correspond a la taille de l'input de la couche, «output_dim» a la taille de
    # l'output (layer d'après) et «activation» a la fonction d'activation choisie pour la couche
neuronale
    # de destination.
    # On crée un dictionnaire en python afin de pouvoir accéder aux valeurs plus tard dans le code.
    {«input_dim»: 5, «output_dim»: 10, «activation»: «relu»},
    {«input_dim»: 10, «output_dim»: 10, «activation»: «relu»},
    {«input_dim»: 10, «output_dim»: 1, «activation»: «sigmoid»},

  ]

# Initialisation de la matrice représentant le systeme neuronal
def init_layers(nn_architecture, seed = 99):
    np.random.seed(seed)
    number_of_layers = len(nn_architecture)
    params_values = {}
```

Initiation of neural network layers

```
    # idx représente l'index et layer la valeur à l'index dans nn_architecture
    for idx, layer in enumerate(nn_architecture):
        layer_idx = idx + 1



        layer_input_size = layer[«input_dim»]
        layer_output_size = layer[«output_dim»]



        # On rempli les valeurs de chaque matrice de nombres aléatoires représentants les poids de
notre systeme
        # Ces nombres aléatoires sont divisés par 10 afin d'obtenir des nombres petits
        # On a besoin de valeurs aleatoires parce que si tous les poids et biais de notre systeme
étaient initialisés
        # avec la même valeur, quelque soit l'entrée, le systeme nous retournerai le même résultat et
nous ne serions
        # pas en mesure de modifier ses valeurs avec une variable.
        params_values['W' + str(layer_idx)] = np.random.randn(
          layer_output_size, layer_input_size) * 0.1
        params_values['b' + str(layer_idx)] = np.random.randn(
          layer_output_size, 1) * 0.1

    return params_values
```

Then, we filled matrix W and vector b with small, random numbers. Indeed, if weights values are initialized with the same number, no matter what the input X was, all units in the hidden layer will be the same too. Ayrton would be stuck in the initial state, no matter his training.

PROTOTYPE

MACHINE LEARNING PROTOTYPE

Activation functions

As we said earlier activation functions gives the neural networks non-linearity and expressiveness that it needs. So Sigmoid and Relu has been used to allow our system to go full circle and pass both forward and backward propagation. We coded them after studying their level of efficiency in relation with each layer.
Knowing that we will need their derivatives later on, we coded them as well.

```
# Définition de la fonction Sigmoid qui prends un float en entrée et retourne un float entre 0 et 1
def sigmoid(Z):
    return 1/(1+np.exp(-Z))

# Définition de la fonction Relu qui prends un float z en entrée et retourne 0 s'il est négatif et z s'il est positif
def relu(Z):
    return np.maximum(0,Z)

# Dérivée de la fonction Sigmoid (utile pour la backpropagation)
def sigmoid_backward(dA, Z):
    sig = sigmoid(Z)
    return dA * sig * (1 - sig)

# Dérivée de la fonction Relu (utile pour la backpropagation)
def relu_backward(dA, Z):
    dZ = np.array(dA)
    dZ[Z <= 0] = 0
    return dZ
```

Forward propagation

Forward propagation is defined as the process that will transform our five inputs into one output, propagating weights between layers.

In our four networks, the information flows in one direction,it is delivered in the form of an X matrix, and then travels through hidden units, resulting in the vector of predictions AyrtonBrain.

```
    # Fonction permettant de determiner quelle fonction d'activation servira à définir la valeur du
neurone.
    # Elle prend comme argument le vecteur de valeurs d'activation du layer A, les poids associés W et
les biais
    # associés b.
    def single_layer_forward_propagation(A_prev, W_curr, b_curr, activation=«relu»):
        # Ici Z est la transformation affine (poids * valeur de l'input layer + biais) a laquelle on applique la
        # fonction d'activation choisie.
        # Numpy et les opérations sur les matrices permettent d'éviter de parcourir la matrice de façon
itérative
        # et augmente grandement l'efficacité et la rapidité de notre systeme.
        Z_curr = np.dot(W_curr, A_prev) + b_curr

        if activation is «relu»:
            activation_func = relu
        elif activation is «sigmoid»:
            activation_func = sigmoid
        else:
            raise Exception('Non-supported activation function')

        return activation_func(Z_curr), Z_curr
```

MACHINE LEARNING PROTOTYPE

Forward propagation

```
    # Cette fonction retourne un dictionnaire python contenant toutes les valeurs d'activation Z et A
qui représente
  # les valeurs avant activation.
  def full_forward_propagation(X, params_values, nn_architecture):
    memory = {}
    A_curr = X

    # Pour chaque élément d'index «idx» et de valeur «layer». Ainsi on crée le dictionnaire de
données.
    for idx, layer in enumerate(nn_architecture):
      layer_idx = idx + 1
      A_prev = A_curr

      activ_function_curr = layer[«activation»]
      W_curr = params_values[«W» + str(layer_idx)]
      b_curr = params_values[«b» + str(layer_idx)]
      A_curr, Z_curr = single_layer_forward_propagation(A_prev, W_curr, b_curr, activ_
function_curr)

      memory[«A» + str(idx)] = A_prev
      memory[«Z» + str(layer_idx)] = Z_curr

    return A_curr, memory
```

Forward propagation

In brief, this propagation takes input signal from the previous layer, computes it with affine transformation Z and then apply selected activation function. By the choice of using NumPy, we could use matrix operations for the whole layer, and consider our calculations as blocs. It also eliminates iteration and significantly speeds up our calculations. In addition to the calculated matrix A, our function also returns an intermediate value Z which is the affine relation between the previous activation, the weight and the bias under the relation:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

MACHINE LEARNING PROTOTYPE

Loss function

Loss function designs the gap between obtained result and expected one at the output layer. In our case, it designed AyrtonBrain and Y (from the database).
The idea is to determine how wrong the answer of the system is, so we can adjust our weights and bias to make it more accurate.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$$

```
# Fonction Cost qui détermine l'erreur entre le résultat obtenu et celui attendu
def get_cost_value(Y_hat, Y, eps = 0.001):
    m = Y_hat.shape[1]
    cost = -1 / m * (np.dot(Y, np.log(Y_hat + eps).T) + np.dot(1 - Y, np.log(1 - Y_hat  + eps).T))
    return np.squeeze(cost)
```

Backward propagation

After defining the Loss function result of the current event, we needed to backward propagate the information into the input layer. This method is based on the partial derivative of according to every weights and bias.

$$\mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} = \frac{1}{m}\mathbf{dZ}^{[l]}\mathbf{A}^{[l-1]T}$$
$$\mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{dZ}^{[l](i)}$$
$$\mathbf{dA}^{[l-1]} = \frac{\partial L}{\partial \mathbf{A}^{[l-1]}} = \mathbf{W}^{[l]T}\mathbf{dZ}^{[l]}$$
$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} * g'(\mathbf{Z}^{[l]})$$

Just like in the case of forward propagation, I decided to split the calculations into two separate functions. The idea is to divide this function in two steps.

The first one focuses on a single layer and is all about rewriting above formulas in NumPy. The second one, representing full backward propagation, deals primarily with key juggling to read and update values in three dictionaries (memory, parameters values and gradients values).
The idea is to take each pair of layers as a unit, then looping on the system backwards to get back to our input layer.

We start by calculating a derivative of the loss function with respect to the prediction vector—result of forward propagation. This is quite trivial as it only consists of rewriting the following formula.

MACHINE LEARNING PROTOTYPE

Backward propagation

$$\frac{\partial L}{\partial \hat{\mathbf{Y}}} = -\left(\frac{\mathbf{Y}}{\hat{\mathbf{Y}}} - \frac{1-\mathbf{Y}}{1-\hat{\mathbf{Y}}}\right)$$

Then iterate through the layers of the network starting from the end and calculate the derivatives. Ultimately, function returns a python dictionary containing the gradient we are looking for (grad_values).

```
    # Fonction permettant de déterminer quelle fonction servira à l'activation des neurones lors de la back propagation
    # L'idée consiste a remonté dans le systeme en remplacant la dérivée de la relation affine des neurones entre eux
    # par d'autres dérivées connues.
    def single_layer_backward_propagation(dA_curr, W_curr, b_curr, Z_curr, A_prev, activation="relu"):
        m = A_prev.shape[1]

        # On détermine quelle fonction d'activation on utilise pour le layer
        if activation is "relu":
            backward_activation_func = relu_backward
        elif activation is "sigmoid":
            backward_activation_func = sigmoid_backward
        else:
            raise Exception('Non-supported activation function')
```

Backward propagation

```
        # on applique la dérivée
        dZ_curr = backward_activation_func(dA_curr, Z_curr)
        dW_curr = np.dot(dZ_curr, A_prev.T) / m
        db_curr = np.sum(dZ_curr, axis=1, keepdims=True) / m
        dA_prev = np.dot(W_curr.T, dZ_curr)

        return dA_prev, dW_curr, db_curr

    # Fonction qui back propage l'erreur afin d'ajuster les poids du systeme
    def full_backward_propagation(Y_hat, Y, memory, params_values, nn_architecture, eps =
0.000000000001):
        grads_values = {}
        m = Y.shape[1]
        Y = Y.reshape(Y_hat.shape)

        dA_prev = - (np.divide(Y, Y_hat + eps) - np.divide(1 - Y, 1 - Y_hat + eps))

        # On parcours le dictionnaire
        for layer_idx_prev, layer in reversed(list(enumerate(nn_architecture))):
            layer_idx_curr = layer_idx_prev + 1
            activ_function_curr = layer[«activation»]

            # La dérivée actuelle devient la dérivée précédente
            dA_curr = dA_prev
```

MACHINE LEARNING PROTOTYPE

Backward propagation

```
 # On met a jour le dictionnaire des données des valeurs d'activation avant et apres avoir ajouté le biais
        A_prev = memory[«A» + str(layer_idx_prev)]
        Z_curr = memory[«Z» + str(layer_idx_curr)]

        # On met à jour les valeurs des poids et des biais dans le dictionnaire
        W_curr = params_values[«W» + str(layer_idx_curr)]
        b_curr = params_values[«b» + str(layer_idx_curr)]

        # On back propagate sur le layer précédent en envoyant les dérivées des valeurs d'activation du neurone précédent, les poids
        # les biais et la valeur Z du neurone ainsi que la fonction d'activation utilisée
        dA_prev, dW_curr, db_curr = single_layer_backward_propagation(dA_curr, W_curr, b_curr, Z_curr, A_prev, activ_function_curr)

        # On sauvegarde les gradients dans un dictionnaire python
        grads_values[«dW» + str(layer_idx_curr)] = dW_curr
        grads_values[«db» + str(layer_idx_curr)] = db_curr

    # On retourne les valeurs des gradients
    return grads_values
```

Backward propagation

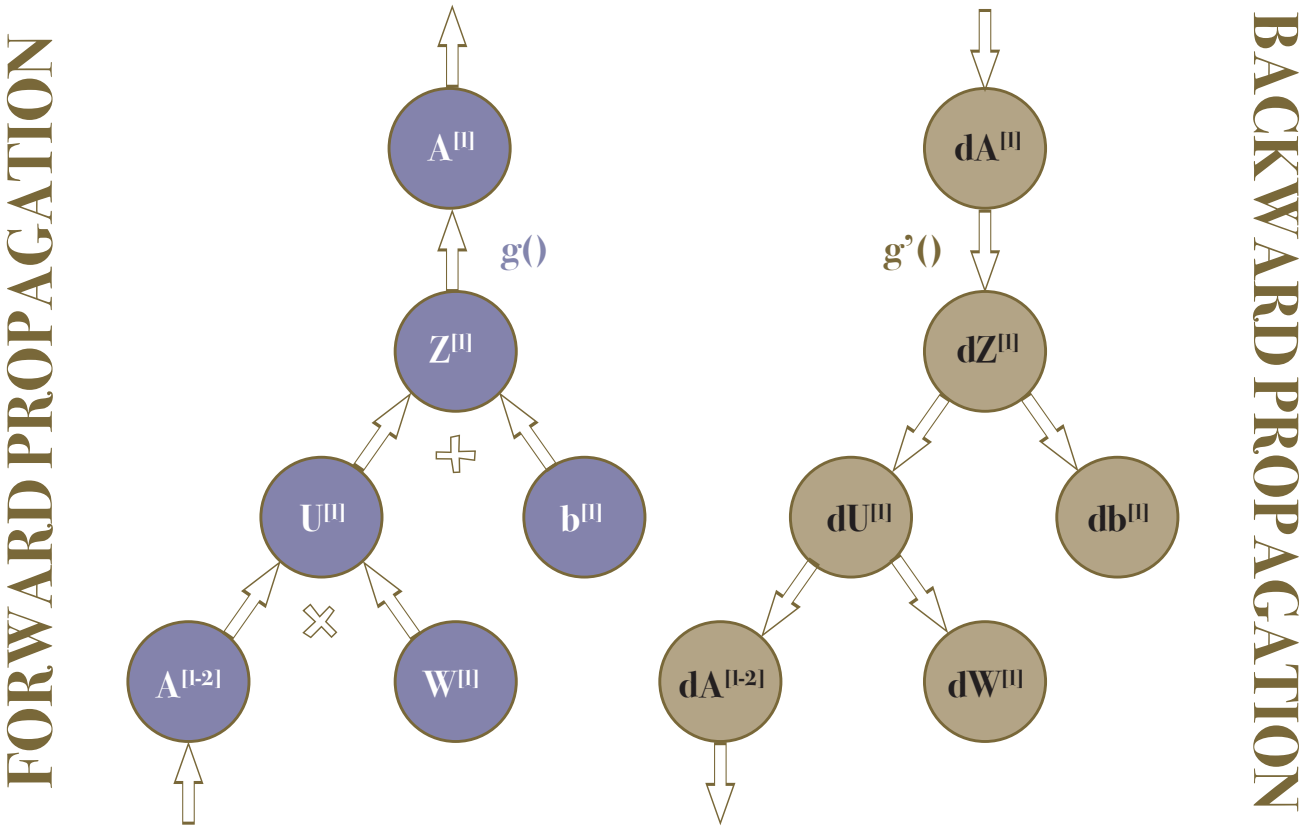


Figure 6. Forward and backward propagation for a single layer.

Source : https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795

MACHINE LEARNING PROTOTYPE

Updating parameters values

The goal of this method is to update network parameters using gradient optimization.
This way, we are looking for the minimum value of the loss function in respect to the parameters (weights and bias).

Now you only need to apply the following equations for each layer.

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \mathbf{dW}^{[l]}$$
$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \mathbf{db}^{[l]}$$

This is a very simple optimization algorithm, but we decided to use it because it is a great starting point for more advanced optimizers.

```
    # Fonction permettant l'optimisation du systeme: on se sert des dictionnaires contenants les poids
et les biais ainsi que celui
    # contenant les dérivées de chacun d'entre eux. On multiplie la dérivée par le rythme d'apprentis-
sage défini plus tôt et on
    # soustrait le tout aux poids et biais concernés.
    def update(params_values, grads_values, nn_architecture, learning_rate):

        for layer_idx, layer in enumerate(nn_architecture, 1):
            params_values["W" + str(layer_idx)] -= learning_rate * grads_values["dW" + str(layer_idx)]
            params_values["b" + str(layer_idx)] -= learning_rate * grads_values["db" + str(layer_idx)]

        return params_values
```

The training

Now that we have prepared all the necessary functions, we just need to put them together in the correct order. This function will be called each time we train our network based on data we designed in this purpose. Those data will be pulled out of the training mode of our program.
Player can drive the car around the track, while sensors will register the distance between the car and the edges of the road. Recording those measurements and associating them with the actions done by the player (accelerating, braking or turning left or right). Those data will be recovered each frame by the unity part of the program and saved in a database managed by SQL.

Therefore, we call it supervised learning. We first show the to network how humans do, then use the data to train the network to act in the same way.

```
  # Fonction d'entrainement du systeme neuronal afin de pondéré correctement ses poids et le rendre plus efficace
  # Elle consiste en l'appel des précédentes fonctions de façon à entrainer notre systeme de manière supevisée
  # Ici X sont les paramètres d'entrée, y les paramètres de correction du systeme (ceux qu'on aurait du obtenir)
  # nn_architecture représente le dictionnaire contenant la taille de chaque layer et sa fonction d'activation,
  # epochs est le nombre d'itérations d'entrainement, learnin_rate est le taux d'apprentissage défini
  def train(X, Y, nn_architecture, epochs, learning_rate):
    params_values = init_layers(nn_architecture, 2)
    cost_history = []
    accuracy_history = []
```

MACHINE LEARNING PROTOTYPE

The training

```
# Ici on itère sur le nombre d'entrainements
  for i in range(epochs):
      Y_hat, cashe = full_forward_propagation(X, params_values, nn_architecture)

      # Ici on sauvegarde la marge d'erreur et la précision de la valeur retournée par le systeme
      cost = get_cost_value(Y_hat, Y)
      cost_history.append(cost)
      accuracy = get_accuracy_value(Y_hat, Y)
      accuracy_history.append(accuracy)

      # On calcule les nouvelles valeures du systeme au travers de propagation avant dans le systeme
      grads_values = full_backward_propagation(Y_hat, Y, cashe, params_values, nn_architecture)
      params_values = update(params_values, grads_values, nn_architecture, learning_rate)

  return params_values, cost_history, accuracy_history
```

PROTOTYPE

UNITY PROTOTYPE

The prototype consists of a program in the following form:

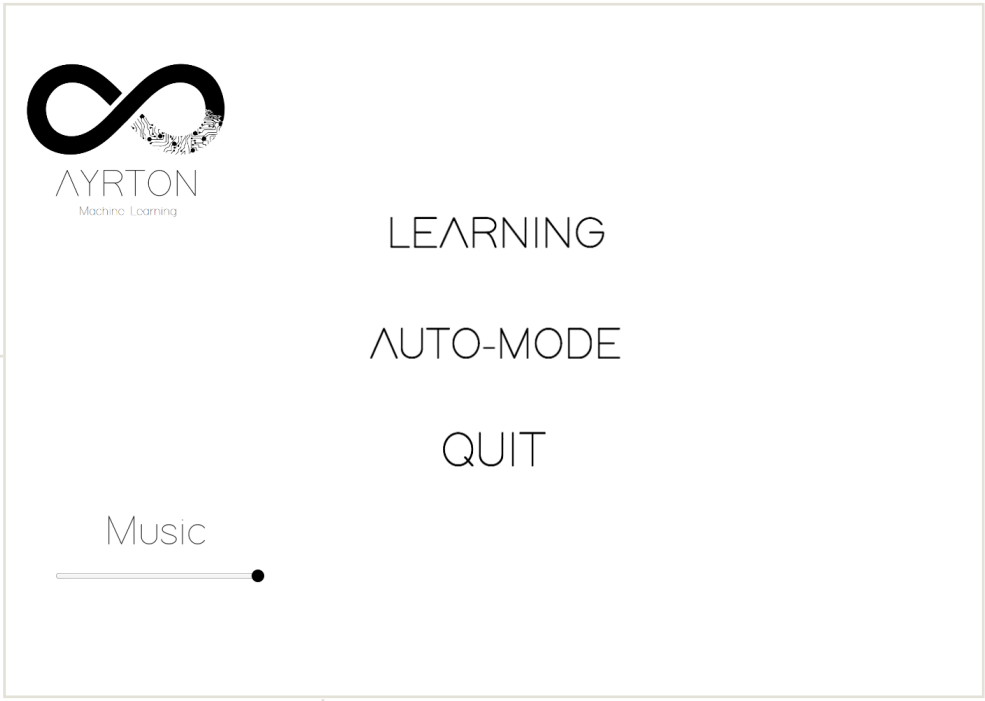
MENU

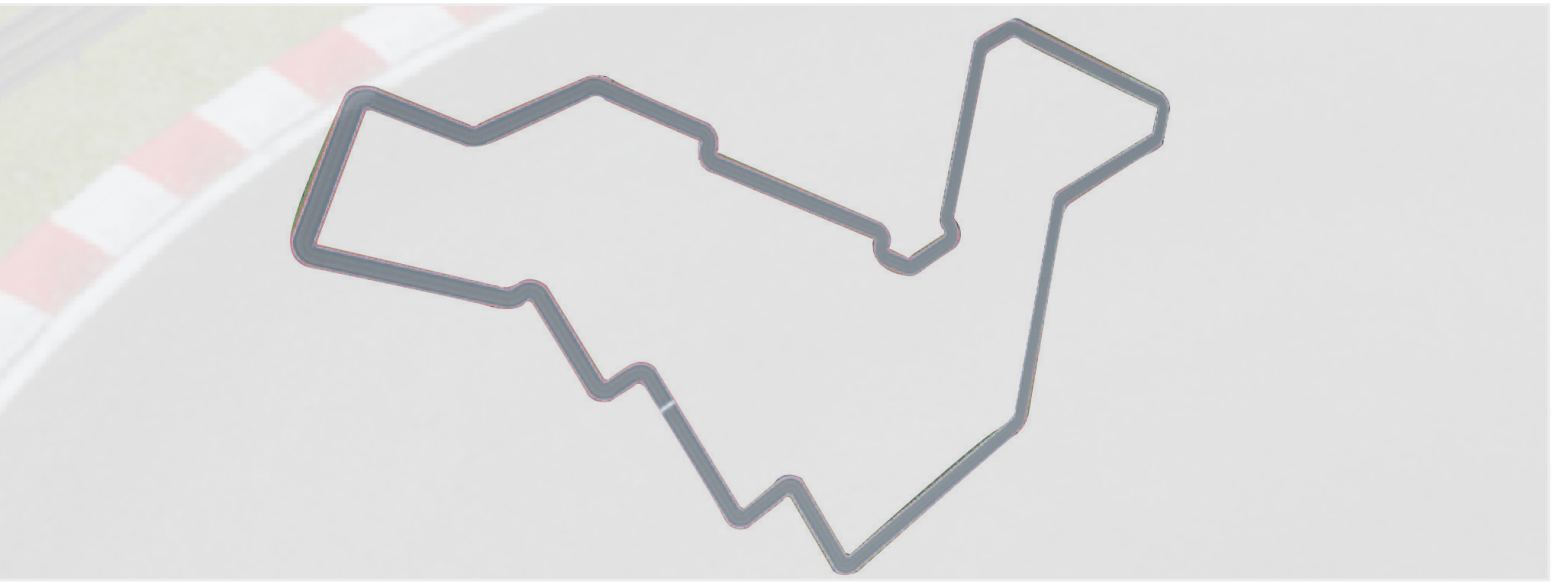
- The program opens to a home menu. This menu consists of 3 user options. Learning mode, auto mode and quit.
- Learning mode is the training part of AI. It is in this mode that we will retrieve the data of the «driving player» and send it back to the neural system so that he can interpret it.
- Auto-mode is the result of learning mode. It is the application of the interpretation system. This mode validates the proper functioning of AI. The quit mode simply closes the application

CIRCUIT

- Each mode is based on the same world.
- A course consisting of mountain ranges and a race track.
- The circuit takes the form of the Imola Grand Prix of Bologna where Ayrton ran for the last time.
- The race track was based on a 2-road asset. A straight stretch of road that allows you to create straight lines. A stretch of road with curves that allows to create angles. These two assets have been duplicated 200 times to create a viable track that meets our expectations

PROTOTYPE





PROTOTYPE

UNITY PROTOTYPE

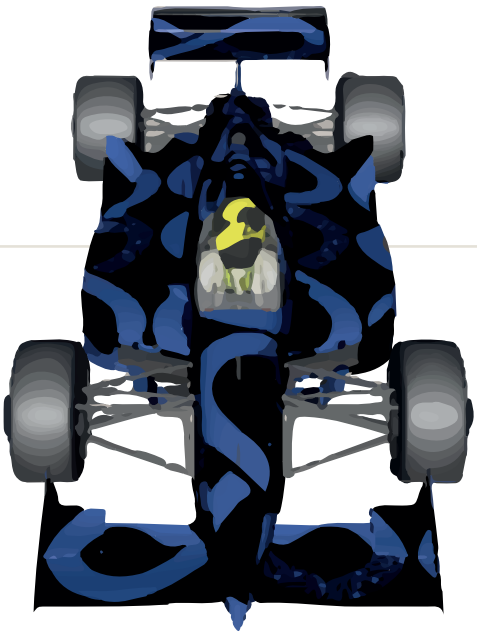
CAR

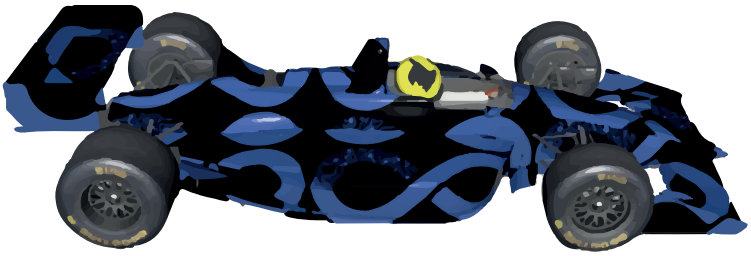
- The car carcass came from an asset. It had to be textured to customize it. It proudly displays the company logo and our colours.
- The entire movement of the car (acceleration, driving, steering and braking) had to be coded using a C# script. And we gave it the physics accordingly (rigidbody and wheelColliders).
- We have also incorporated car noise using audio files.
- It was also necessary to create sensors to recover the position of the car in relation to the various colliders of the circuit to return them to artificial intelligence. The colliders are based on a system of cubes placed at the roadside. It was necessary to remove the renderer to make them invisible in play.

SENSOR

- Each mode is based on the same world.
- A course consisting of mountain ranges and a race track.
- The circuit takes the form of the Imola Grand Prix of Bologna where Ayrton ran
for the last time.
- The race track was based on a 2-road asset. A straight stretch of road that
allows you to create straight lines. A stretch of road with curves that allows to create angles. These two assets have been duplicated 200 times to create a viable track that meets our expectations

PROTOTYPE





AYRTON

PROTOTYPE

UNITY PROTOTYPE

MAP

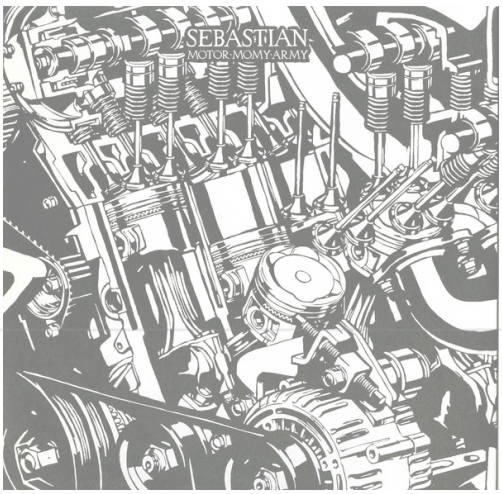
- For the map environment we decided to carry on mountains
- These are reminders of many Formula 1 circuits.
- This choice has also gained in simplicity.
- It consists of a field where mountains were modeled using the Lower/Grower Terrain tool.
- Textures applied in this field are sober.Greenery on all mountains accompanied by a yellow earth texture to give it a little life.

AUDIO

The sound coming out of the game is essential for more realism. Several audios were imported for formula 1, menu or map :

- When the machine accelerates an acceleration sound is triggered, so is braking.
- For the menu, the choice was to put a music a in the theme “Motor - Sebastian”, which is basically designed with Car Samples.
- Finally for the Map (Game)

The operation of Audio in Unity consists of assigning an audio listening(Audio listener) to the camera that will follow F1 throughout the circuit and one or more audio source(s) (Audio Source) to the car that will emit the desired sound.



PROTOTYPE

TOP VIEW

AYRTON

UNITY PROTOTYPE

AudioLola Script

CODE

This script focuses on acceleration sounds.

```
public class AudioLola : MonoBehaviour
{
  //déclaration d'une source audio et son clip audio
  AudioSource audiosource;
  public AudioClip son_acceleration;

  void Start()
  {
    audiosource = GetComponent<AudioSource>();
  }

  void Update() {
    //si la touche flèche haut est appuyé alors le clip audio est lancé
    if (Input.GetKey(«up») && audiosource.isPlaying == false)
    {
      audiosource.clip = son_acceleration;
      audiosource.Play();
    }
    // si la touche est lâché le clip s'arrèe
    else if (Input.GetKeyUp(«up»))
    {
      audiosource.Stop();
     }
  }
}
```

AudioEngine Script

This script focuses on brake sound.

```
public class AudioEngine : MonoBehaviour
{
  //déclaration d'une source audio et son clip audio
  AudioSource audiosource;
  public AudioClip son_deceleration;

  void Start()
  {
  audiosource = GetComponent<AudioSource>();
  }

  void Update()
  {
    //si la touche flèche haut est appuyé alors le clip audio est lancé
    if (Input.GetKeyUp(«up»))
    {
      audiosource.clip = son_deceleration;
      audiosource.Play();
    }
    // si la touche est lâché le clip s'arrête
    else if (Input.GetKey(«up»))
    {
      audiosource.Stop();
    }
  }
}
```

UNITY PROTOTYPE

Animation Script

This script focuses on Cinematic animations.

```
public class Animation : MonoBehaviour
{
    //déclaration de la source d'animation et ses clip d'animation
    Animation anim_cam;
    public AnimationClip anim_ini;
    public AnimationClip anim;

  void Start()
  {
    anim_cam = GetComponent<Animation>();
    //lancement de l'animation 'anim' lors du lancement du game
    anim_cam.clip = anim;
    anim_cam.Play();
  }

  void Update()
  {
    //si la touche flèche haut est appuyé alors 'anim' est stoppét 'anim_ini' est lancé
    if (Input.GetKey(«up»))
    {
    anim_cam.Stop(); anim_cam.clip = anim_ini; anim_cam.Play();
      }
   }
}
```

UNITY PROTOTYPE

Car Controls Script

This script focuses on Car Controls.

```
public class CarControls : MonoBehaviour
{

    // Pour les wheel Collider
    public WheelCollider WheelFL;//*
    public WheelCollider WheelFR;
    public WheelCollider WheelBL;
    public WheelCollider WheelBR;

    // Pour les roues en tant qu'objet
    public GameObject FL;
    public GameObject FR;
    public GameObject BL;
    public GameObject BR;

    public float topSpeed = 650f; // Vitesse maximum de la voiture
    public float maxTorque = 200f;// La valeur maximum des freins
    public float maxSteerAngle = 45f; // la valeur de l'angle de direction
    public float currentSpeed; //La vitesse de la voiture
    public float maxBrakeTorque = 2200; //force du freinage

    private float Forward;// Direction avant
    private float Turn;// Direction de turn
    private float Brake;// Les freins

    private Rigidbody rb;// Pour le rigidbody de la voiture
```

UNITY PROTOTYPE

Car Controls Script

```
void Start()
{
    rb = GetComponent<Rigidbody>(); //Trouver l'objet des le début auquel on a attaché ce script
}
void FixedUpdate()
{
    Forward = Input.GetAxis(«Vertical»);
    Turn = Input.GetAxis(«Horizontal»);
    Brake = Input.GetAxis(«Jump»);

    WheelFL.steerAngle = maxSteerAngle * Turn;
    WheelFR.steerAngle = maxSteerAngle * Turn;

    currentSpeed = 2 * 22 / 7 * WheelBL.radius * WheelBL.rpm * 60 / 1000;
//formule pour calculer en km/s

    if (currentSpeed < topSpeed)
    {
      WheelBL.motorTorque = maxTorque * Forward;// Fait tourner les roues
      WheelBR.motorTorque = maxTorque * Forward; // motorTorque = Couple moteur sur l'es-
sieu de la roue exprimé en Newton-mètres. Positif ou négatif selon la direction.
    }

    WheelBL.brakeTorque = maxBrakeTorque * Brake; //Couple de freinage exprimé en Newton-
mètres. Doit être positif.
    WheelBR.brakeTorque = maxBrakeTorque * Brake;
    WheelFL.brakeTorque = maxBrakeTorque * Brake;
    WheelFR.brakeTorque = maxBrakeTorque * Brake;
}
```

UNITY PROTOTYPE

Car Controls Script

```
    void Update()// Script appellé une fois par frame

    {   //Les quaternions sont utilisés pour représenter les rotations.

        Quaternion flq;//rotation du wheelCollider FL (Front Left = Devant a gauche)
        Vector3 flv;//position du wheelCollider
        WheelFL.GetWorldPose(out flv, out flq);//Recupère la position et la rotation
du wheelCollider
        FL.transform.position = flv;
        FL.transform.rotation = flq;

        Quaternion Blq;//rotation du wheelCollider BL (Back Left = Derrière à gauche) Vector3 Blv;
        WheelBL.GetWorldPose(out Blv, out Blq);
        BL.transform.position = Blv;
        BL.transform.rotation = Blq;

        Quaternion fRq;//rotation du wheelCollider FR (Front Right = Devant à droite)
        Vector3 fRv;
        WheelFR.GetWorldPose(out fRv, out fRq);
        FR.transform.position = fRv;
        FR.transform.rotation = fRq;

        Quaternion BRq;//rotation du wheelCollider BR (Back Right = Derrière à droite)
        Vector3 BRv;
        WheelBR.GetWorldPose(out BRv, out BRq);
        BR.transform.position = BRv;
        BR.transform.rotation = BRq;
    }
}
```

UNITY PROTOTYPE

Sensor Controls Script

This script focuses on creating sensors on the car.

```
public class DetectionSensor : MonoBehaviour
{

    public GameObject Car;
    public GameObject Collider;

    public float sensorLength = 250f; //taille du capteur
    public Vector3 sensorFrontPosition = new Vector3(0f, 0.0f, 2f); //La position du capteur de
devant.
    public float sensorSidePosition = 0.81f;
    public float sensorSideDiagonal = 15;

    public float positionCollider;

    //Test de récuperation des distances

    public float distanceFront; // Permet de connaitre le distance entre le capteur front et le collider
    public float distanceSide;  // Permet de connaitre le distance entre le capteur
front side 1 et le collider
    public float distanceSide2; // Permet de connaitre le distance entre le capteur front side 2 et le
collider
    public float distanceDiag1; // Permet de connaitre le distance entre le capteur diagonal et le
collider
    public float distanceDiag2; // Permet de connaitre le distance entre le capteur diagonal2 et le
collider
    private Vector3 sensorEndPosition;
```

Sensor Controls Script

```
//création des sensors

    public void Sensors()
    {
        // Crée un rayon vers un collider
        RaycastHit hit;
        // Definit la position d'un objet 3D. Ici le centre de la voiture
        Vector3 sensorStartPosition = transform.position;
        //Placer le capteur

        // Permet de Vector3 la position du collider pour pouvoir l'utiliser dans la
fonction Vector3.distance
        Vector3 positionCollider = Collider.transform.position;

        sensorStartPosition += transform.forward * sensorFrontPosition.z;
        sensorStartPosition += transform.up * sensorFrontPosition.y;

        // La fonction suivante fonctionne en prenant le Vector 3 d'origine, la
direction (prenant en compte la rotation de l'objet), sur le point de collision(hit)
        // d'une distance max de sensorLength
        //Pour le premier tracé du front middle
        //le capteur du front center
        if (Physics.Raycast(sensorStartPosition, transform.forward, out hit, sensorLength))
        sensorStartPosition += transform.right * sensorSidePosition;
```

UNITY PROTOTYPE

Sensor Controls Script

```
{
        //Créee une ligne qui part de la position du sensorStartPosition vers le
prochain point de collision
        Debug.DrawLine(sensorStartPosition, end: hit.point, Color.white);

        Vector3 sensorEndPosition = hit.point;
        //La distance entre le debut du capteur et l'objet
        distanceFront = Vector3.Distance(sensorStartPosition, sensorEndPosition);

    }

    //Le capteur du front right
    sensorStartPosition += transform.right * sensorSidePosition;
    if (Physics.Raycast(sensorStartPosition, transform.forward, out hit, sensorLength))
    {
        //Créee une ligne qui part de la position du sensorStartPosition vers le
prochain point de collision
        Debug.DrawLine(sensorStartPosition, end: hit.point, Color.white);
        Vector3 sensorEndPosition2 = hit.point;

        //La distance entre le debut du capteur et l'objet
        distanceSide = Vector3.Distance(sensorStartPosition,sensorEndPosition2) ;
    }

    //Le capteur du front right diagonal

    if (Physics.Raycast(sensorStartPosition, Quaternion.AngleAxis(sensorSideDiagonal, transform.
up) * transform.forward, out hit, sensorLength))
```

UNITY PROTOTYPE

Sensor Controls Script

```
    {
        //Créee une ligne qui part de la position du sensorStartPosition vers le
prochain point de collision
        Debug.DrawLine(sensorStartPosition, end: hit.point, Color.white);
        Vector3 sensorEndPosition3 = hit.point;
        //La distance entre le debut du capteur et l'objet
        distanceSide2 = Vector3.Distance(sensorStartPosition, sensorEndPosition3);

    }
    //Le capteur du front left
    sensorStartPosition -= transform.right * sensorSidePosition * 2; // 2 fois la right pour faire la
left
    if (Physics.Raycast(sensorStartPosition, transform.forward, out hit, sensorLength))
    {

        Vector3 sensorEndPosition4 = hit.point;
        //Créee une ligne qui part de la position du sensorStartPosition vers le
prochain point de collision
        Debug.DrawLine(sensorStartPosition, end: hit.point, Color.white);
        Vector3 sensorEndPosition = hit.point;
        //La distance entre le debut du capteur et l'objet
        distanceDiag1 = Vector3.Distance(sensorStartPosition, sensorEndPosition4);

    }
    //Le capteur du front left diagonal
    if (Physics.Raycast(sensorStartPosition, Quaternion.AngleAxis(- sensorSideDiagonal, trans-
form.up) * transform.forward, out hit, sensorLength))
```

UNITY PROTOTYPE

Sensor Controls Script

```
    {

        Vector3 sensorEndPosition5 = hit.point;
        //Créee une ligne qui part de la position du sensorStartPosition vers le
prochain point de collision
        Debug.DrawLine(sensorStartPosition, end: hit.point, Color.white);
        //La distance entre le debut du capteur et l'objet
        distanceDiag2 = Vector3.Distance(sensorStartPosition, sensorEndPosition5);

    }

  }

  // Start is called before the first frame update

  // Update is called once per frame
  void Update()
  {
Sensors();
  }
}
```

CONCLUSION

To conclude, these 2019 Ydays have allowed us to learn how to manage our time on a larger project than usual. Good agreement, hard work, and leveraging each other's capabilities made us grown as students.

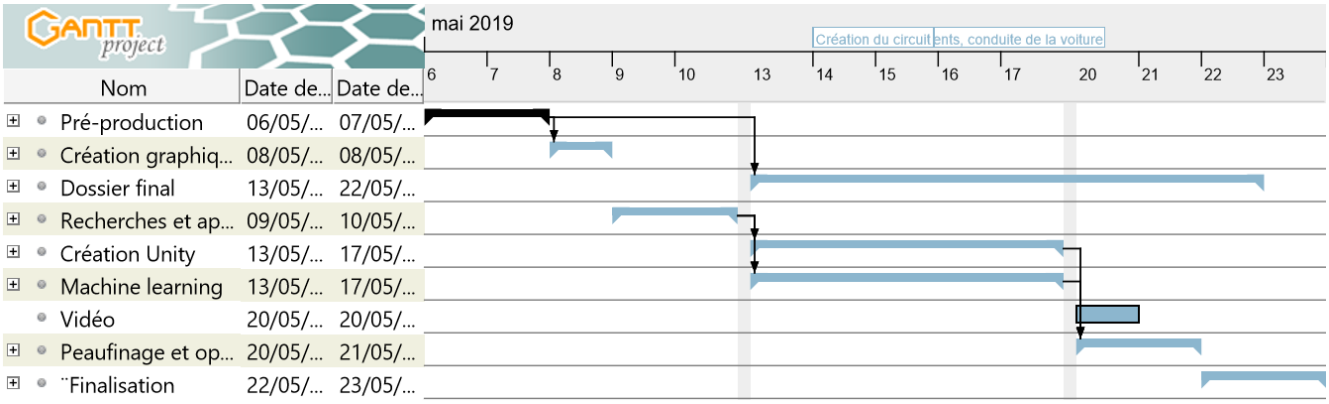
We were able to start handling the Unity software and its various functionalities (audio, interactions between objects, scene creation, etc.) and resume our object-oriented programming courses by applying it to the python language.

Although our project could not be functional in the time allotted, we are going beyond our expectations and will finish this project and learn from our mistakes.

It is now time to thank the people who helped us get this project to where it is today, namely: Eva Ponthieu--Havard, for her help on the graphic layout of this file and our presentation support and Mathieu Sallé for his advices on the business plan. We would also like to thank everyone who gave us time or attention during these Ydays.

APPENDICES

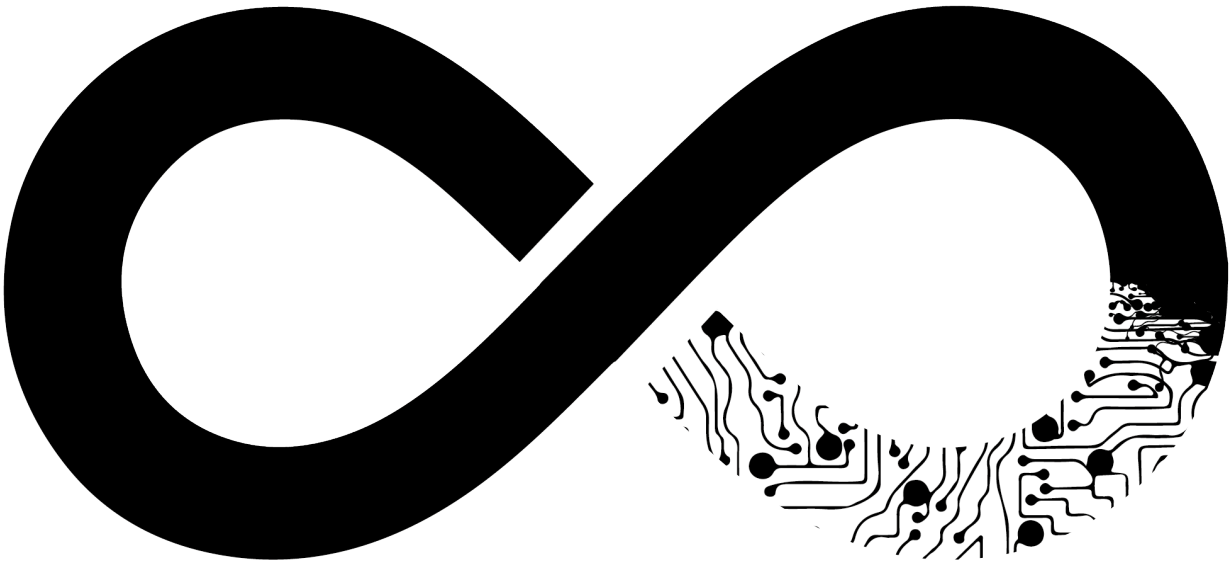
GANTT PROJECT



Tâches

Nom	Date de début	Date de fin
Recherches et apprentissage	09/05/19	10/05/19
Apprentissage Machine learning	09/05/19	09/05/19
Apprentissage Unity	09/05/19	09/05/19
Identification des difficultés et prise en main	10/05/19	10/05/19
Création Unity	13/05/19	17/05/19
Création du terrain et application des textures	13/05/19	13/05/19
Habillage de la voiture	13/05/19	13/05/19
Création du circuit	14/05/19	14/05/19
Animation de départ	15/05/19	15/05/19
Mouvements, conduite de la voiture	15/05/19	15/05/19
Création du menu	16/05/19	16/05/19
Création sonore	16/05/19	16/05/19
Création des capteurs	17/05/19	17/05/19
Machine learning	13/05/19	17/05/19
Création de la matrice	13/05/19	13/05/19
Faire propager les données depuis les inputs au output en les faisant passer par la couche cachée	14/05/19	14/05/19
Calculer la perte (loss function)	15/05/19	15/05/19
Calculer les gradients	16/05/19	16/05/19
Répercuter l'erreur sur notre systeme (back-propagation)	17/05/19	17/05/19
Vidéo	20/05/19	20/05/19
Peaufinage et optimisation	20/05/19	21/05/19
Unity	20/05/19	21/05/19
Gestion du volume	20/05/19	20/05/19
Amélioration conduite	20/05/19	20/05/19

Créations de capteurs fonctionnels	21/05/19	21/05/19
Ergonomie Menu	21/05/19	21/05/19
Machine learning	20/05/19	21/05/19
Récupération et traitement des données vouées à l'entrainement de la machine	20/05/19	20/05/19
Débugs	21/05/19	21/05/19
¨Finalisation	22/05/19	23/05/19
FInalisation dossier	22/05/19	22/05/19
Préparation oral	23/05/19	23/05/19
Textes	23/05/19	23/05/19
Support visuel	23/05/19	23/05/19



AYRTON