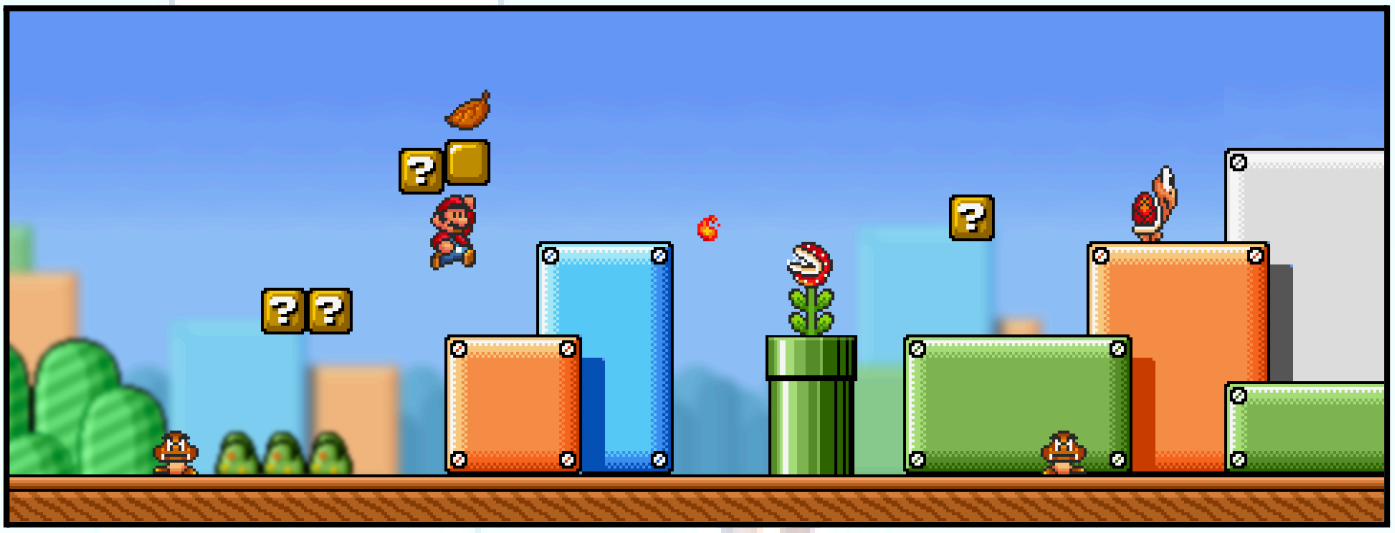
PATTERN: DEFINITION AND THEORY

Patterns are the elements inserted into the levels to **confront the player with the challenges present in the game**. They set the **pace** of a level (with consequent definition of the Rest Spaces) and define the **difficulty** of the latter.

To complete a level the player can decide to tackle it as he prefers, the patterns serve to **mark the progression phases** during the gameplay. After defining the **Challenges** and creating the **ingredients** necessary for **generating the patterns** (*contained in* ***Ingredients*** *and* ***Ingredients Correlation***), all the patterns that can develop in a level are defined.

The final patterns that determine the **creation of the levels** with the **added challenge of the Dash** are contained in the **Pattern Definition sheet.**



Pattern Difficulty

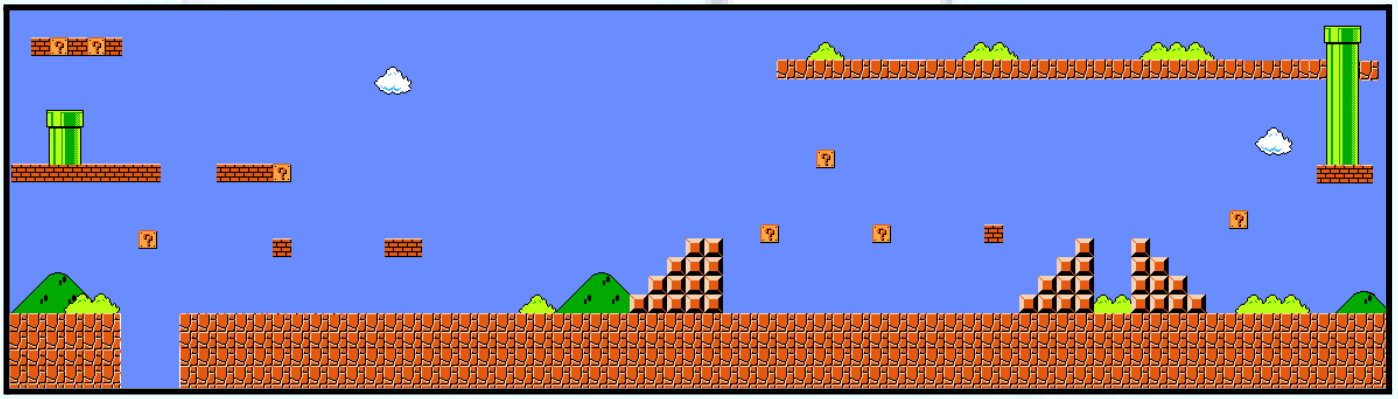
For each pattern a **difficulty** must be defined which will then define the **difficulty of the entire level**. The fundamental aspects are the **sequentiality of the patterns**, the **Rest Spaces** for the players, the **similarity** and the **Level Design context**.
The difficulty of a pattern is given by the following formula:

$$Difficulty = \sum_{i=1}^{n} (Ingredient_i) + c$$

where the **sum of the difficulties of all the ingredients that make up the pattern is added to a constant value that follows a pre-established difficulty curve.** The curve shows how the difficulty of a certain pattern derives from the **number of consecutive interactions** of which it is composed (*requires a high and constant level of attention from the player, exponential difficulty*).

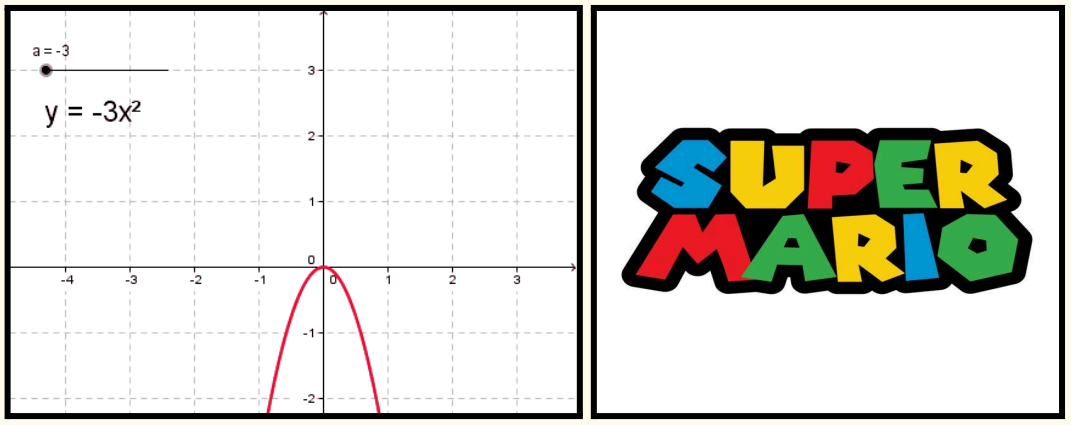
Sequentiality of the Patterns

Having numerous patterns consecutively without interruptions given by rest spaces causes the player a **level of stress and tension directly proportional to the duration of the patterns themselves**. This influences the **difficulty** of the entire level, forcing the designer to correctly position the **rest spaces** and give it a **satisfactory size** in order to start again with the progression for the player.



Rest Spaces: impact on the Formula

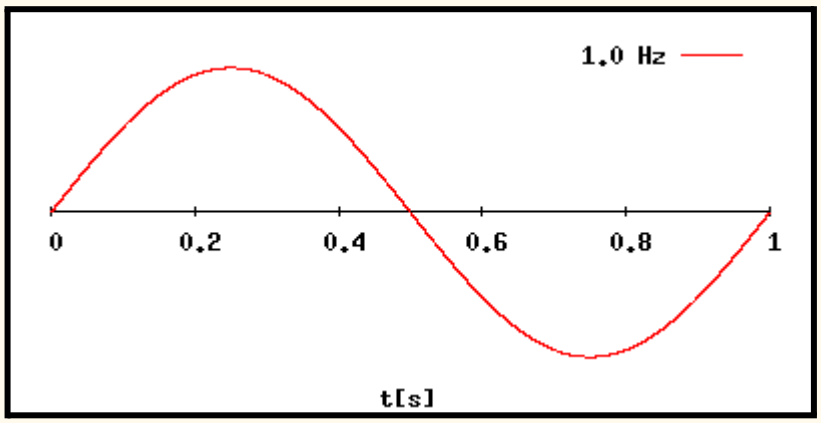
The **Rest Spaces** for the player are a fundamental element in the **positioning of the patterns** within the level because they **slow down the pace and modify the stress the player experiences** in facing the game challenges.
The main characteristics of a rest space are **length (time duration)** **and** **frequency**. These two aspects define the impact of a rest space on the pace curve. The effects of the coefficients of a parabola on its resultant can be observed in **GeoGebra**.



Rest Space time duration

The **duration** of a Rest Space is defined by the time in which the player does **not encounter challenges** while continuing at the **established base pace of the game** (*standard walking speed*). The increase in this distance is **directly proportional to the decrease in pace**, decreasing the **derivative of the pace graph** and keeping it negative *until another challenge is presented*.

The risk that arises from using rest spaces that are too long is that of **bringing the pace to a value below the threshold established by the game**, leading the player to get **bored** and *not feel any type of accomplishment in the gameplay.*

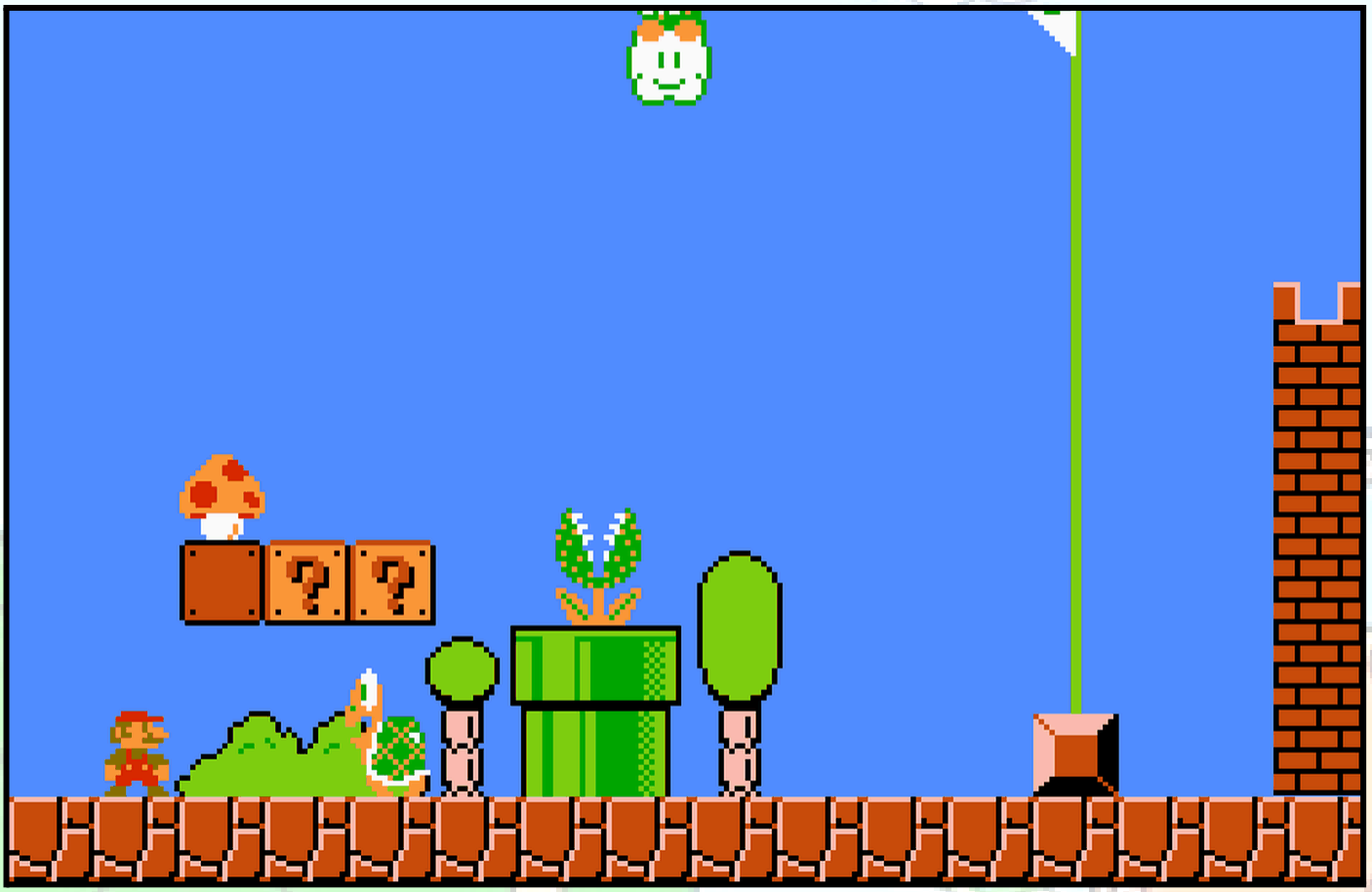


Rest Space Frequency

Once it has been established how the **impact of a Rest Space** varies based on its duration, it must be linked to the **frequency** with which they are presented to the player. When a **Rest Space** is presented to the player it begins to **provide a constant negative acceleration to the pace function**, gradually slowing it down throughout its duration.

As soon as a challenge pattern is presented again, the **acceleration of the pace graph** becomes **positive** again. Following this concept, the insertion of rest spaces must be measured based on the **impact** they have on the **pace graph** because if the player is not given time to **fully enter the challenge** he loses the **perception of the difficulty of the game** , also losing the sense of **accomplishment** during the level.

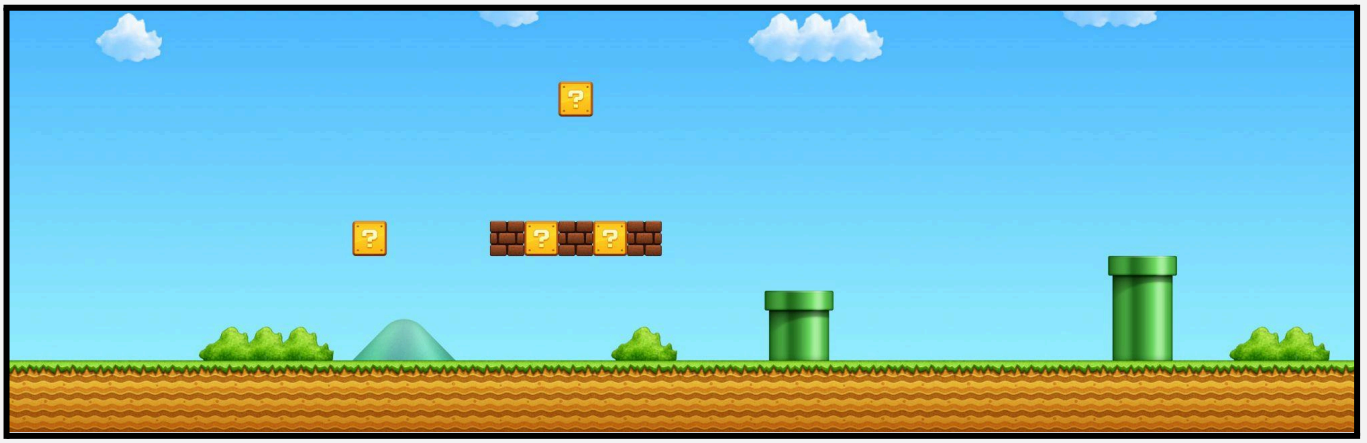
The **frequency** of the rest spaces must be **tuned** in order to **regulate the pace curve** within two **initial/finite thresholds** established by the designers.



Similarity & Internal Similarity Variation

In the definition of the ingredients (**Ingredients Correlation**) new ones have been **created** to provide a more precise analysis and develop more varied patterns.

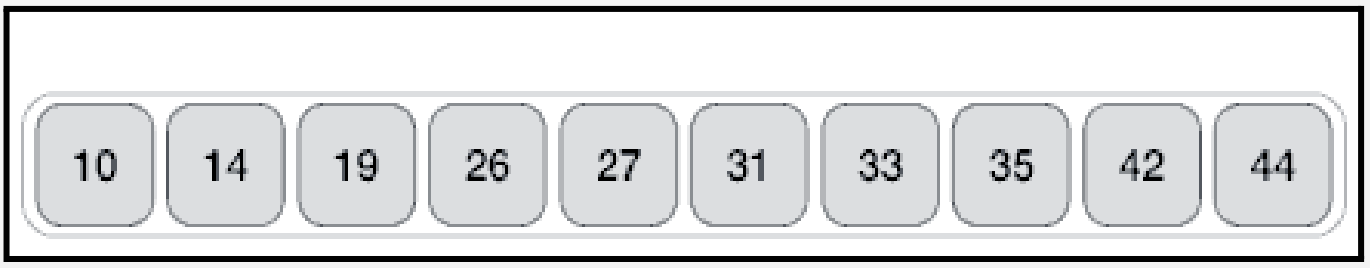
In Super Mario Bros, **movement, jumping and (after our additions) dash** are challenges **connected** to each other with such **depth** that they influence each other's functioning and rules.
The rules for **identifying the similarity and internal similarity** of a pattern are provided mathematically in the **Pattern Definition** file and establish **similarity values** to be attributed to any pair of patterns to establish whether it is possible to discard any of them.



Since we have not established a fixed length for the patterns, it is not possible to correctly calculate the total number of possible patterns (*each pattern can have n ingredients in each of its slots, therefore ingredients^n possible combinations*)

Similarity : Comparison Strategy

The **comparison between two patterns** is done using the correspondence with the **equality check of the array lists in programming** (*the arrays are chosen and not the sets because the order counts in giving a similarity value*).



Taking advantage of linear search and array scrolling, we run the algorithm on two patterns **P1** and **P2** that we are considering.

$$P_1 = (a, b, c) \mid P_2 = (x, y, z)$$

Each pattern is made up of **individual ingredients**. By exploiting the algorithm (*which is also done on patterns of different dimensions, for example patterns of the same dimension were used*) it is necessary to analyze the **first element of each of the two patterns.**

$$P_1(1) = a \mid P_2(1) = x$$

Now we are going to carry out a check on these two elements by following the following rules and obtaining the relevant scores.

- $a = x \mid Score = 1$ *(same ingredient for the pattern)*
- $a \subset x \mid Score = 0.5$ *(if a contains x, score is halved.)*
- $a \not\subseteq x \mid Score = 0$ *(if a doesn't contains x, score is 0)*

The **similarity value** between two patterns is given by the **sum of the values returned by the function**. For patterns of different lengths **the algorithm stops at the last element of the shortest pattern**. By applying variations to the algorithm it is also possible to find out if a pattern is contained in the middle of another in order to split them.

If the algorithm is called ***Sim*** and give it as parameters ***(index, P1,P2)*** it is possible to have the final formula for the **Similarity calculation**:

$$Similarity = \sum_{i=1}^{m} Sim(i, P_1, P_2)$$

with **m** equal to the size of the smaller pattern. To evaluate whether or not to discard a pattern, a **similarity threshold** must be established beyond which one cannot go. In the case of analysis it is inserted in **Ingredients Correlation**.

Internal Similarity: Comparison Strategy

For **Internal Similarity** the decision algorithm is much easier. Simply count the number of ingredients that make up the Pattern under consideration and decide on a **threshold** beyond which the latter should be discarded because it is too redundant in terms of ingredients.