Dragonfly 3RR0R

The Breakdown

Daniel Violato
dviolato@gmail.com

INTRODUCTION

Due to the predatory exploitation of natural resources, the Earth became unsuitable for human life a few centuries in the future. The only planet found to replace it, Exile Elysium, was thousands of light-years from here. Although interplanetary travel technology was much more advanced, there was no way to go there in time feasible to continue the species. The solution was to use cryogenic techniques to freeze people during the trip without them growing old.

Obviously, the spacecraft could not be piloted. To complicate matters, the ship should be constantly supplied with the necessary resources to keep the cryogenics activity. To circumvent these drawbacks, the spacecrafts called **Dragonflies** were developed. Such as the insects with the same name, they fly over their paths and make brief landings to gather the resources they need. The entire trip would take place fully automatically, with humans remaining in complete **Stillness.**

Players will control the Dragonfly 3RR0R, which had the misfortune to be subjected to a Breakdown. The programs that manage the necessary resources to keep the cryogenics are not working well. So from time to time, the crew wake up from hypersleep and need to pass computer commands to correct the deficiencies of the Dragonfly 3RR0R. As we know, computers are a **Different Audience** using a language of its own. Luckily, the crew are all trained in the necessary programming languages.

It happens that, while awake for programming Dragonfly 3RR0R, the crew grow older. The less efficient they are, the less time they will have to spend with their beloveds, all aboard of other dragonflies. In the ship's jargon, they speak of the **Abandonment.** Players will be able to do it in short enough time?

Preface

This game is a cooperative resource management boardgame where players' actions are passing computer codes with the use of cards. I'm fully aware that it is not mathematically a good game, being far from ready. In fact, I recognize that the game is a mathematical shame. However, I do not think one can make a boardgame such as this complete without playtests and in just some few days.

In fact, this is not the proposal of the Game Chef. My focus was just to present the ideas of a game that if arouse interest, will be further developed in the appropriate manner. Similarly, I acknowledge the graphics of the game is much in need of improvement.

Finally, before introducing the rules, I would like to inform you that English is not my first language. Therefore, I kindly request you to be patient about grammatical errors or other language vices contained in this text

Thank

Welcome aboard!.

Rules

This is a game for 2-6 players that uses:

A Gameboard https://goo.gl/sJl8xE;
Cards https://goo.gl/18cLYR;
Six four sided dice;
5 large markers (for the tens of resources);
5 small markers (for unities of resource);
1 Age Marker;
1 First Player Marker;
Reference sheets (I couldn't finish them).

A match is a sequence of 7 rounds, each with the following phases:

1. Resource consumption;
2. Choice of the Planet to Explore;
3. Programming Exploitation Code;
4. Exploration;
5. Calculating the Achieved Energy and aging time.

Resources

The Dragonflies utilize both liquid nitrogen (N) and liquid helium (He) for cryogenics. They can be obtained from a number of different circumstances in each planet. Nitrogen and Helium taken as raw materials require different Chemical Reagents (Re) to be put into liquid form and used in cryogenics process.

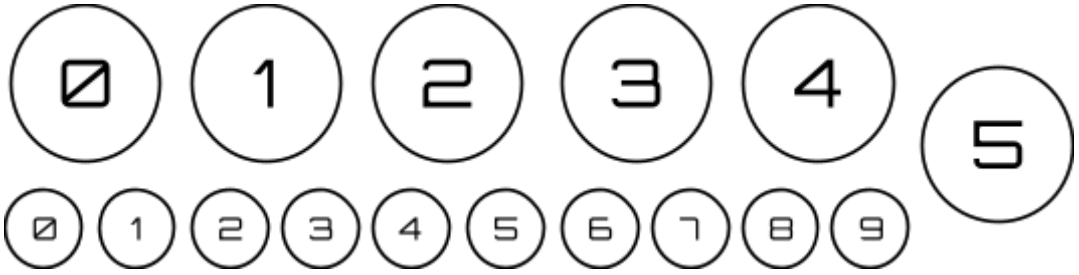
Moreover, when awakened, the crew of the Dragonfly 3RR0R need Provisions, such as oxygen and food. If the stock of this resource reaches zero, the game ends immediately.

Finally, to carry out its operations, Dragonfly 3RR0R spend energy.

All of these features are represented by the following symbols:

N	Nitrogem	He	Helium
Re	Reagents	P	Provisions
	Energy		

The amount of resources that Dragonfly 3RR0R contains, initially set at 50 each, are represented throughout the game by the next portion of the gameboard:



Larger circles represent the tens and the smaller units. When a unit has to be reduced to less than zero, one ten is reduced and the Unit Marker goes back to the nine position. The reduction then proceeds from there up to the total. Similarly, when it has to be increased to more than nine, one ten unit is increased and the marker returns to zero and continues to be increase thereafter.

The Dragonflies' load compartment is limited to 50 units of each resource, value that cannot be overcome throughout the game.

Example: the game begins with the large marker at 5 and the small at 0 for all resources. If a particular feature is to be reduced by 12, the score of tens goes to 3 and the units to 8. If it is increased by 7 units, the marker of the ten goes to 4 and the unit to 5.

Resource Consumption.

In the first phase of each round, roll-up dice to determine how many resources were consumed during the period in which the Dragonfly 3RR0R has been traveling between planets. These rolls are made only for N, He, Re and P. Spending Energy has a more complex mechanics, described in the following sections of these rules.

If one of the Cryogenics Resources (N, He or Re) is too low, the Dragonfly use the other in larger quantities to compensate for the lack of it. Yet, the lower the Cryogenic Resources are, more Provisions are required to maintain the crew. Thus, although for different grounds, mechanical consumption is basically the same for Cryogenics and Provisions..

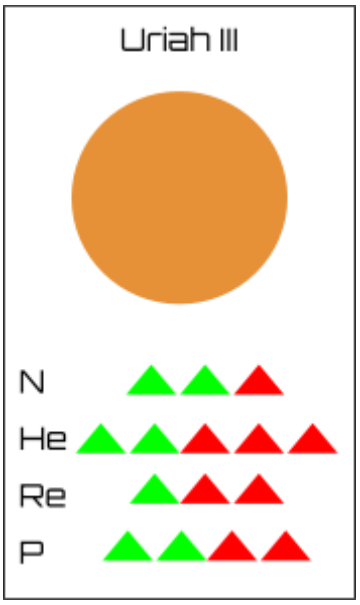
The amount of resources consumed is determined by rolling dice. For the resources with the least amount, four dice are rolled, their values are added and the result is subtracted from the stock. For resources with a ten higher than this one, roll up five dice and for resources with two or more tens, six dice.

Example: at some point of the match, N is the resource with the least amount, of 22 units. Re is at 41 units, while He is at 34. So you need to roll four dice to consume N, six dice for Re (two tens above N), and 5 dice to He (one ten above N). The first roll results in 9 (the dice show 1, 2, 2 and 4). N goes to 13. The second roll results in 16 (1, 1, 3, 3, 4 and 4). Re go to 25. The third roll yield 11 (2, 2, 2, 2 and 3), and He goes to 23.

After rolling dice for the four types of resources (N, He, R, P), proceed to the next phase.

Choice of Planet to Explore.

The planets are arranged in a deck containing cards showing their traits. The cards must be shuffled and the deck must be placed face down so that players cannot see the planets traits. This is an example of a Planet card:



At this phase of each round, players must draw three cards and choose one as the planet they will explore. To understand the strategy for choosing the world to explore, it is important to know how the exploration will be made. Therefore, I will forward in this section the mechanics of the fourth phase of each round.

The Planet traits reveal the (probable) suitability to extract and rarity of each resource on this planet. The suitability to extract determines the cost of energy to extract some amount of resources. The rarity indicates how many extractions can be made before the cost increase.

By the time the crew are awake and need to program the exploitation code for the Dragonfly 3RR0R, its algorithms are so messy that it is not possible to be certain about the suitability to extract and the rarity of resources on the planet.

The information about suitability to extract and rarity of the resources on the planet are presented respectively by green and red triangles following the icons of each resource in the planet card. The green triangles inform the potential cost in energy to extract certain amount of resource and red triangles how many extractions can probably be made at this cost before it gets higher (the resources become scarcer on the planet and therefore more difficult to be extracted).

There may be one, two or three triangles of both colors, and the higher the number, the smaller the chances for them being easily extracted and abundant on the planet. The more green triangles, harder will probably be its extraction and the more red triangles smaller tend to be the amount of resources that can be extracted at that cost of energy.

The number of green triangles determines how much dice will be rolled in the fourth phase of the round, the exploration, to determine the suitability for effective extraction of each resource. It will be equal to the lowest number rolled in the dice, and means how many units of the resource can be extracted at the cost of a unit of energy.

The number of red triangles determines the amount of dice to be rolled following to see how many extractions can be made by the cost determined in the previous rollover. Similarly, the number of extractions is equal to the result of the lowest of the dice rolled. Each time the Dragonfly 3RR0R perform this number of extractions, the suitability to extract decreases by 1. That is, one can extract a unit less per unit of energy expended.

Example: The players are in need of Re and therefore choose Uriah III as the planet to explore (the small number of dice in Re increases the probability of achieving a fair amount of Re at low energy cost). As provided in the card above, they should roll one die to determine the suitability to extract and two dice to determine its rarity. In the first die, they take 3, indicating that they can take 3 Re units for each unit of energy expended. In the second roll, they draw 2 and 4, which, being 2 the lowest, means that the energy cost to exploit this resource increases every two extractions. Thus, they can make two extractions of 3 Re units per consumed energy, two more extractions at the cost of one energy for every two units of Re and the following extractions will be an energy per unit worn. Thus, to extract 13 units of Re, they will have to pay 7 power units (each energy worth the following amount of Re: 3 + 3 + 2 + 2 + 1 + 1 + 1).

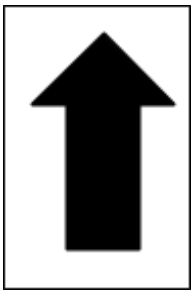
Note that all dice rolls are made only in the fourth phase of each round. Before it, there is another phase made without knowing exactly the suitability to extract and the rarity of the resources regarding the planet. It is explained now..

Programming the Exploration Code

Before beginning this phase, players can freely discuss which strategy they will use in programming. It is important to seize this moment because when building the code itself they cannot communicate. The systems of Dragonfly 3RR0R are so committed that communication between its programs is not efficient.

Expressions

To build the code, players use cards that symbolize the expressions that can be used in the programming language of Dragonfly 3RR0R. These expressions are as follows (the thorough understanding of their meaning will be possible only after the explanations of how they can be combined to form statements):

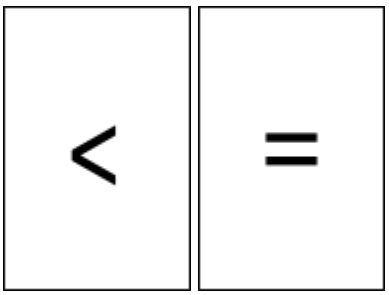


Expression used to perform extractions of resources of the planet;

****** **********

Conditional Expressions, which means that the code sequence following their conditions is executed only if the condition is true - concerning "else", that can only be used in conjunction with if, the execution of code that follows it occurs when the condition is false;

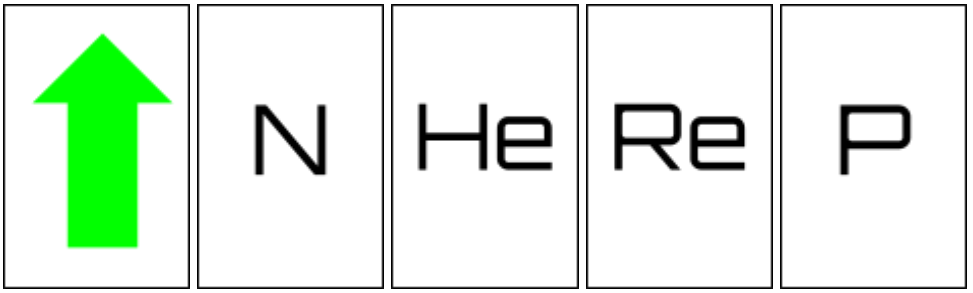
****



The same card is used to represent "less than" and "greater than", depending only on whether it is placed face-up or face-down (as <or>);

Expressions that mean "less than", "greater than" and "equa tol", used in conditional expressions to establish relations between number and suitability to extract or the amount of resources carried by Dragonfly 3RR0R at a given time.

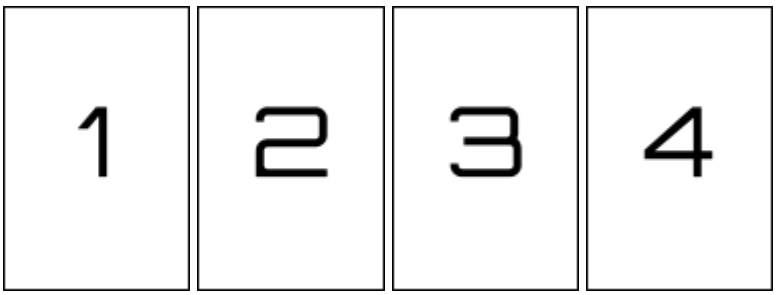
****



The green arrow represents the suitability for extraction and N, He, Re and P is the game resources. In the presentation syntax of the sentences, resources are generally represented by ◇

Expressions that can be used in conditional statements to specify the actual suitability for extraction of the planet (specified during the exploration phase) or the resource the condition refers to. Resources is also used along with the get statement to specify which is the feature that will be extracted;

****



When presenting the syntax of the statements, the numbers are generally represented by #.

Expressions used in conditional statements to specify which is the condition that must be satisfied so that the code is executed or in get statements to determine how many extractions will be made.

****

Writing the Code

As with all software, the language used by Dragonfly 3RR0R has its own syntax. This syntax regards the order in which the expressions of language (the cards) are arranged within the code. Only the codes built according to the correct syntax are run by Dragonfly 3RR0R's system.

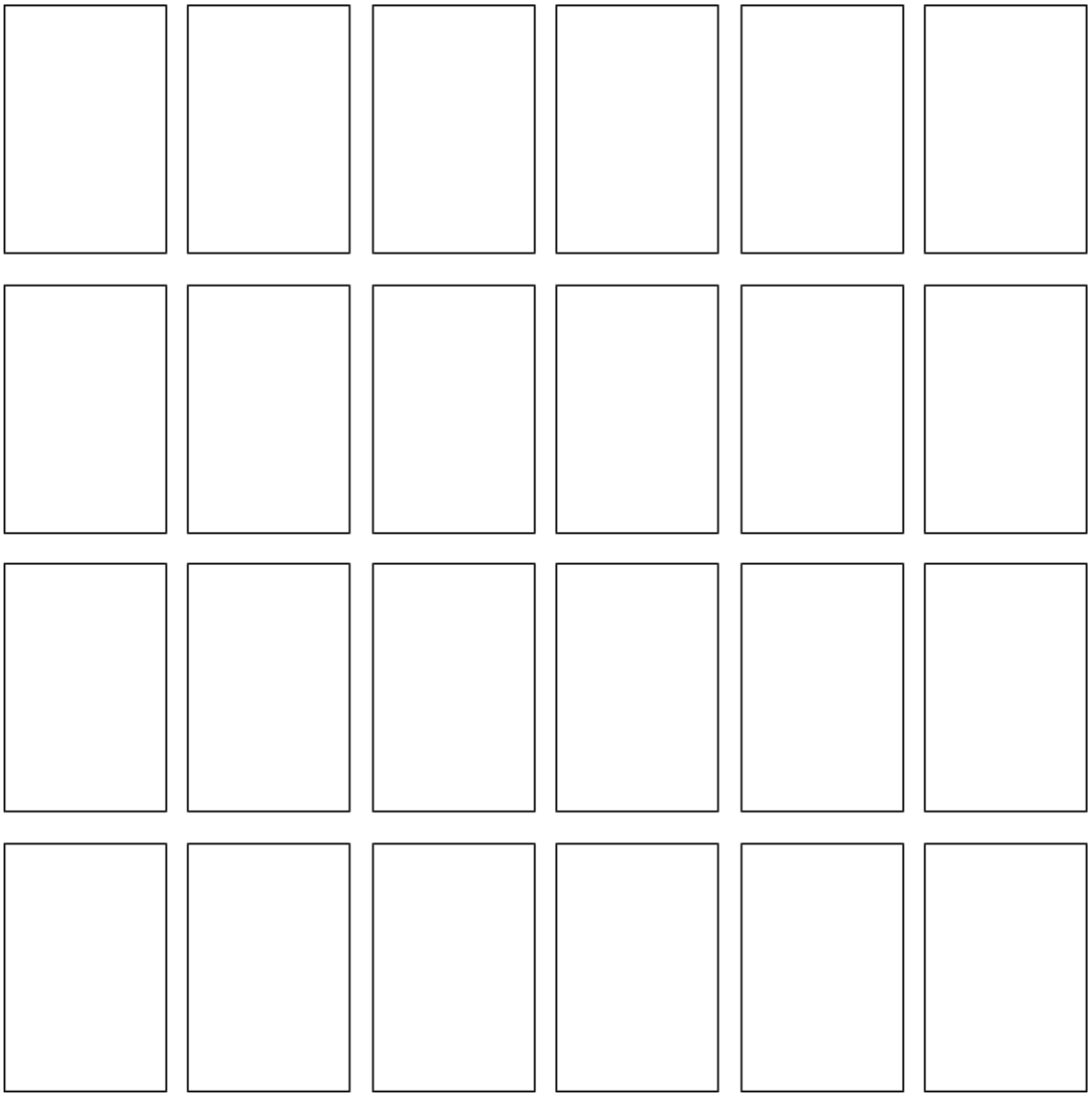
Before you start building the code, you must define who will be the first player. To this end, each player rolls three dice and add their values. Whoever has the highest result is the first player. In case of a tie, the two players who have the highest result make a new roll until one can tell which one will be the first player.

The first player changes every round. At this phase in subsequent rounds, the new first player is the one who is to the left of the first player from the previous round. That is, the role of first player is switched clockwise.

After the first player is determined, 30 expressions cards are distributed among the players. In a game with two players, each receives 15 cards. With three, each gets 10. Being four, the first and second players receive 8 and the other two 7. And so successively.

From the moment the expressions cards are dealt, players can no longer talk about the strategy they will use during this phase. Cards received, each player chooses which to keep and discard the rest. For each disposal, draws one new card. The set of cards in hand of each player are the expressions they you can use to form the code.:

The code is the group of cardsarranged in the following of the gameboard fild:



Each square of the field is a slot that can hold a card, and players can download **any card** in **any slot** (remembering that only well-constructed syntactically statements will be executed).

The first player download the first card and the other players download theirs in clockwise order.

Note that this session of the board consists of rows of slots, but the passage of the end of a line to the beginning of another is continuous. That is, it is as if there was only one line. The layout in different lines serves only to better organize the board.

During the fourth phase of the round, the exploration, the code will run on the sequence from the first to the last statement. So, when downloading the cards, it is very important that the players think hard before deciding the slot to put their cards. Remember that players cannot communicate at that moment (except to talk about subjects unrelated to the strategy they are adopting, of course).

This observation is important because players can download **any card** in **any** slot. If the expressions in a player's hand are not good to continue a statement already in construction, the start of a new statement is allowed. However, this is a risky maneuver because it tends to generate syntactically poorly constructed statements.

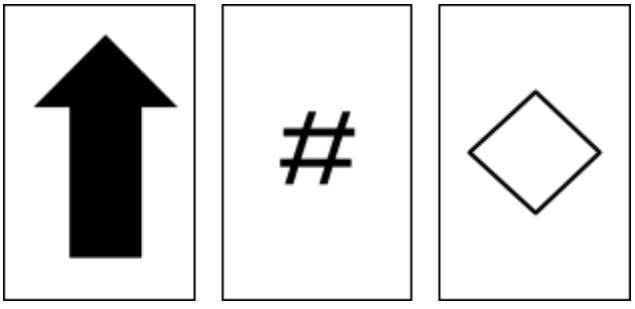
Finally, **blanks are ignored by Dragonfly** 3RR0R.That is, a sequence of expressions containing empty slots is compiled the same way as the case where there is no empty slots.

Statements

Statements are expressions sequences syntactically well constructed for passing commands to the Dragonfly 3RR0R. Statements can be get statements or conditional statements and conditional statements can be built with other conditional statements inside them. In the syntax described below, the word "Statement" can be a get statement or take any conditional statement.

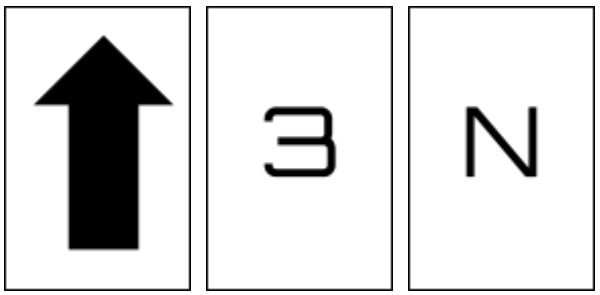
The simplest and most necessary statement executed by Dragonfly 3RR0R is to get resources. The command orders the Dragonfly 3RR0R to performs as many extractions of a resource type as indicated by the number.

Syntaxe:

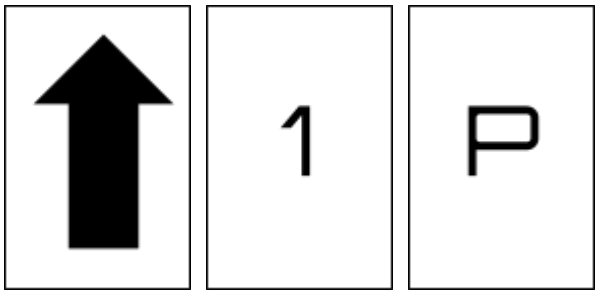


 # Represents one of four numbers, 1, 2, 3 and 4.
◇represents one of the four resouces, N, He, Re and P.

Examples:



orders the Dragonfly 3RR0R to conduct 3 Nitrogen extractions.



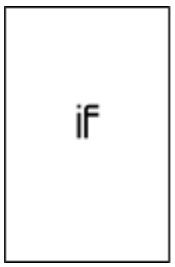
orders the Dragonfly 3RR0R to conduct 1 Provisions extraction.

Each extraction performed at the time of the next phase of the round, the operation realize an energy expenditure corresponding to the suitability to resource extraction. If the ship does not have Energy to run a command to exctract something, it does not run and the round finishes.

Conditional Statements

The "if" and "while" operators are conditional operators, which means that their code is executed only provided their condition is satisfied. In addition, the operator "else" also has conditional valeu, since it only makes sense within a "if" statement.

Syntax.:



+ condition + statement;

+ condition + statement +

+ statement;

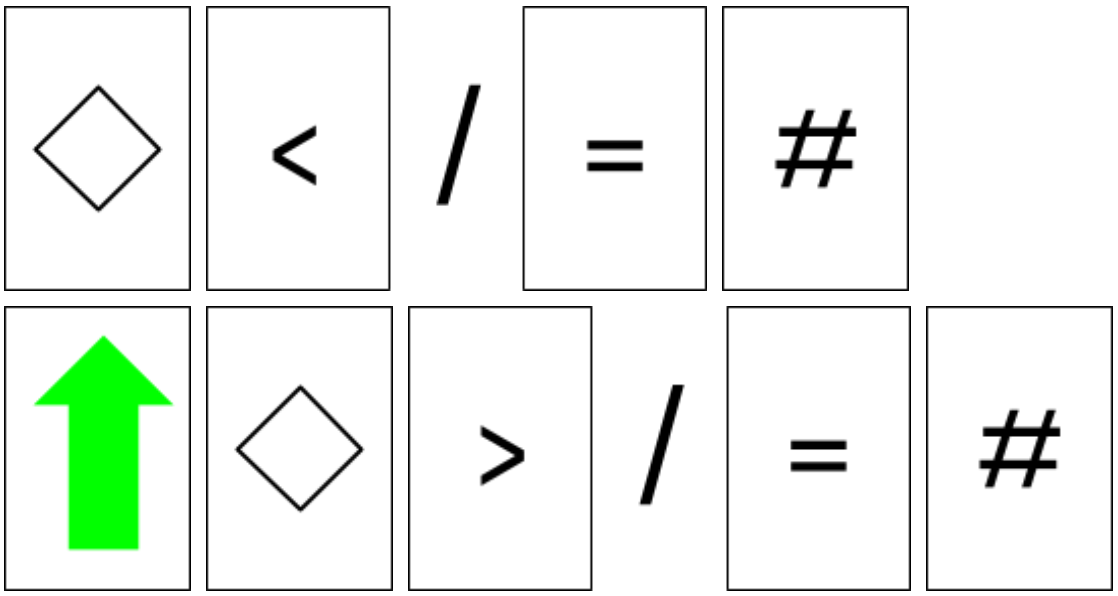
+ condition + statement;

Before giving examples of conditional statements, it must be explained what a condition is.

Condition

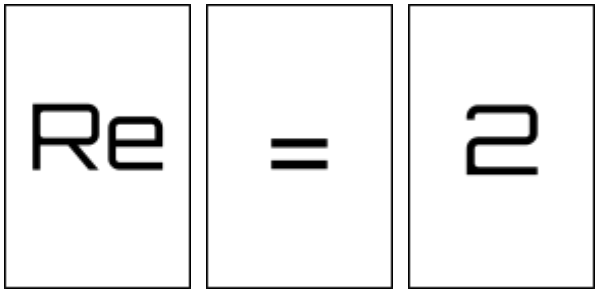
The condition is a code sequence that can be true or false.

Syntax:

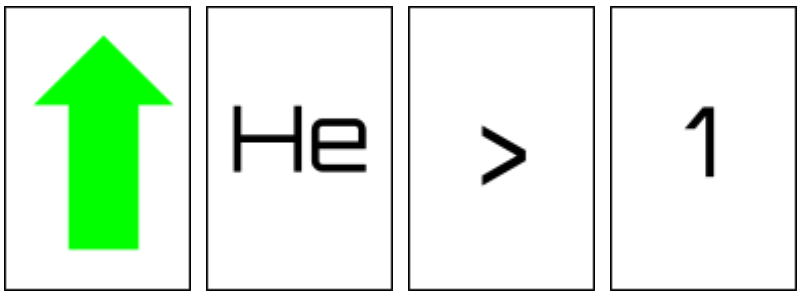


In the first case, the condition is true only if the **ten** of the resource is smaller / equal to the number that follows the < or =, while in the second case it is true only if the suitability for extraction of the resource specified is greater than the number that follows the > or =.

Examples::



This condition is true while Dragonfly 3RR0R's Re is in the house of the second ten (20 to 29).



This condition is true as long as the suitability for extraction of Helium is greater than 1.

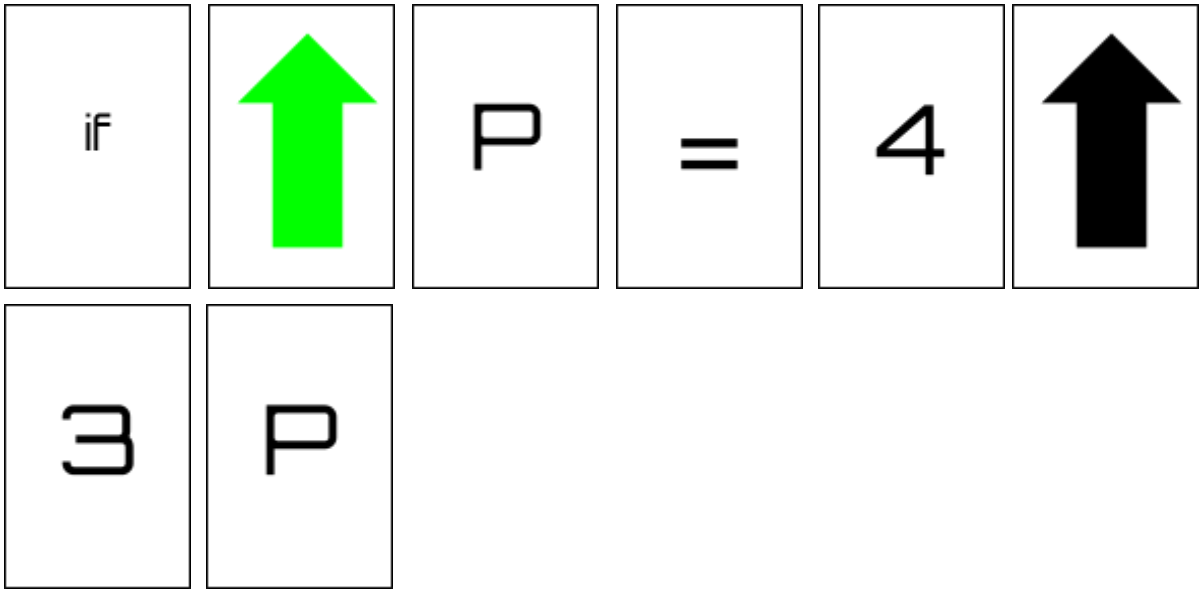
Resuming Conditional Statements

Both in case of statements with "if" as in case of statement with "while", the statement that follows the condition will only run if this is true. The difference is that it is performed only once in the case of an expression using "if" and possibly several times if it used the expression "while". That is, "while" is a looping expression, which runs until the condition is no longer met.

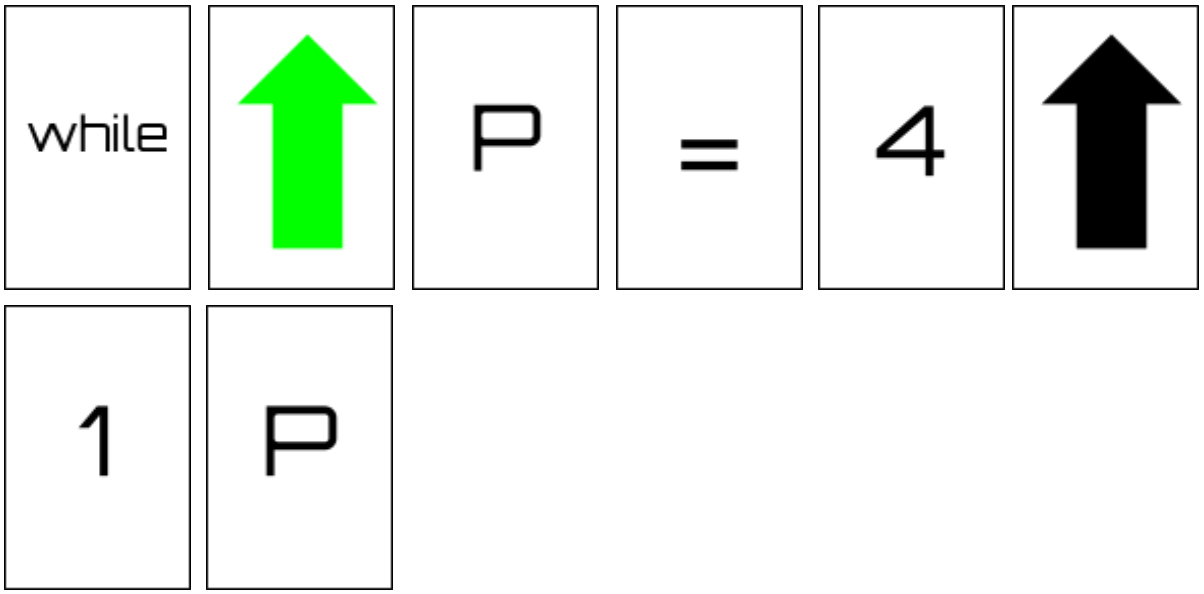
Finally, a conditional statement that has "else" + statement (after the statement that follows the condition) is executed only if the condition is false. Also, the statement following the condition is not executed.

Watch out for an infinite loop (with a statement with "if" + "else" after a condition "while", for example). It will drain your Energy!!! In this case, the loop is executed until the resource stocks sum 50 or the energy reaches 0, whichever comes first

Examples.:

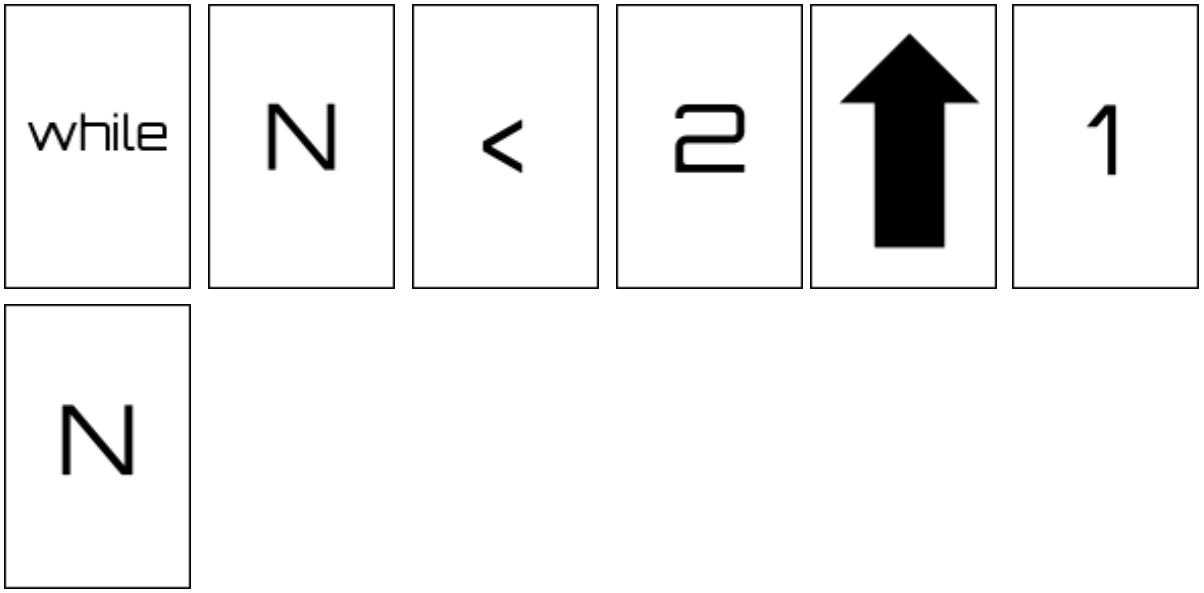


This statement orders the Dragonfly 3RR0R to do 3 P extractions if the suitability for extraction is equal to 4. Note that **the condition is checked only at the beginning of the execution of the statement**. If P becomes scarce throughout the extraction process, there may be the consuming of one energy to which there is the return 3 units of P (or even less), not 4. The next example illustrates how to extract P just as P is very easy to be extracted.

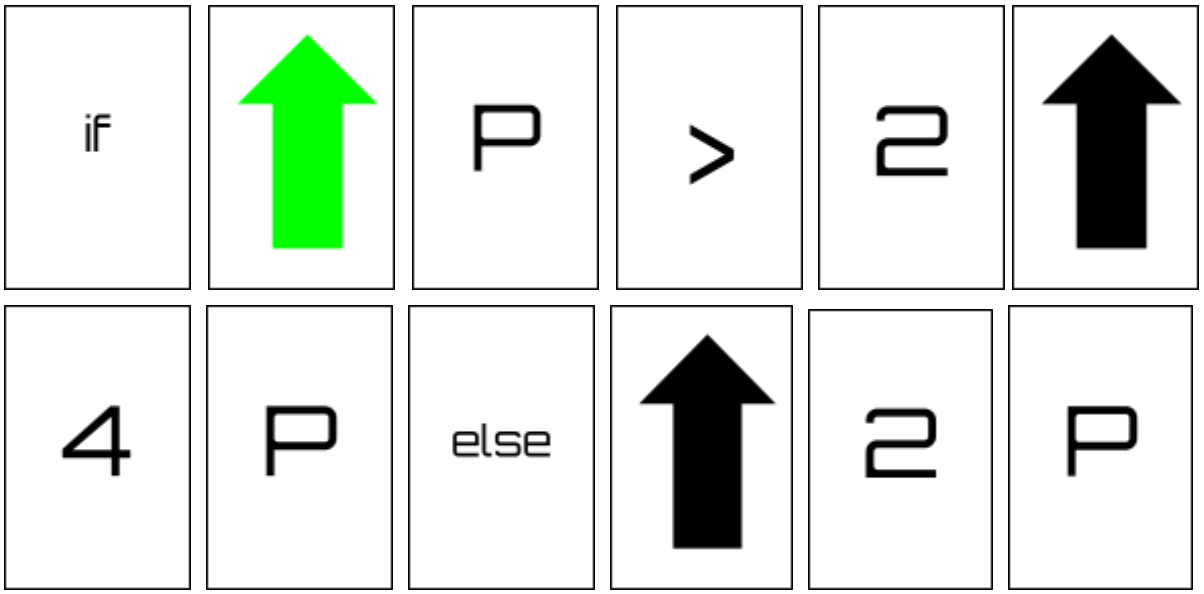


This statement orders the Dragonfly 3RR0R to make extractions of 1 P while its suitability for extraction is equal to 4. Note that **the condition here ALSO is checked only at the beginning of the execution of the statement**.

If instead of 1 there were a larger number, there would be extractions of 3 or less units of P at the cost of one energy.



This statement orders the Dragonfly 3RR0R to do one extraction of N while its stock is below 30 N, being very useful to decrease the aging (see below).



This statement orders the Dragonfly 3RR0R do 4 P extractions if its suitability for extraction is equal to 3 or 4, making only two if it is smaller.

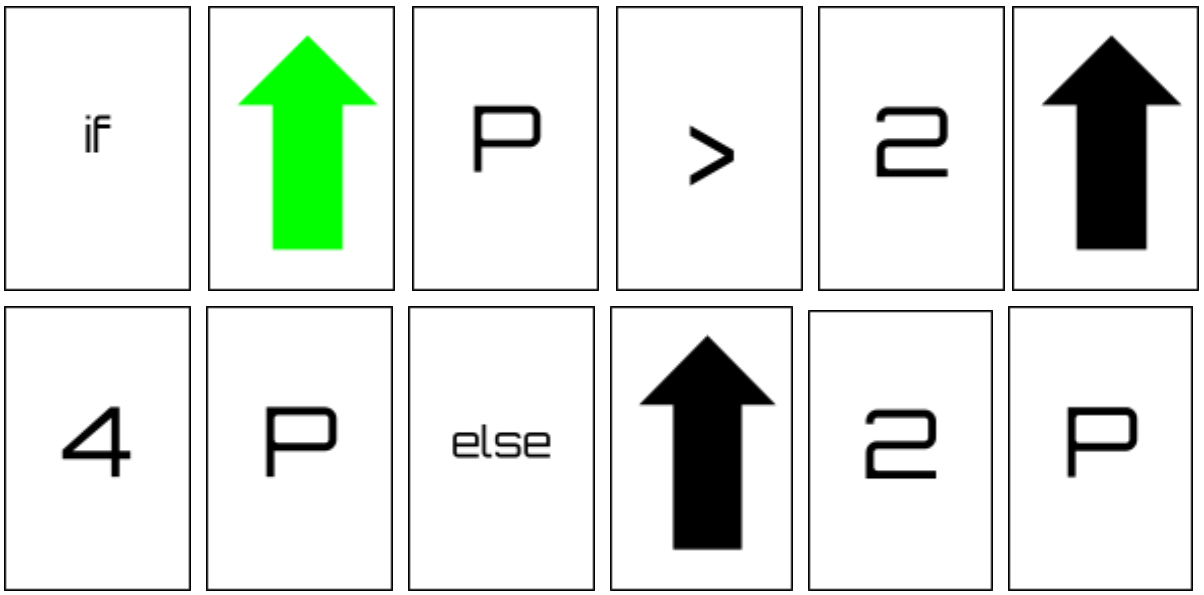
Exploration

During the exploration phase, the Dragonfly 3RR0R lands on the planet and run its code. First, determine the amount of resources the planet effectively has by rolling dice and only then the codes are executed.

For each Resource, roll as many dice as triangles are indicated for the resource type in the card. The lowest value of each roll is the effective value of the suitability for extraction and the resource rarity (see Choice of the Planet to Explore on how this is done). These are the values used for the effective implementation of the code.

Example: the last example's statement (repeated below) was in the code for the exploitation of Uriah III (also quoted in Choice of the Planet to Explore). It has two green dice (suitability for exploitation) and two red (rarity). In the first roll, it takes 3 and 3, so that the suitability to effectively extract the planet's P is equal to 3 (lowest rolled value). The second roll results in 1 and 4, making effective rarity equal to 1.

Therefore, the condition of the statement is true and 4 extractions are performed. But since P is quite rare on the planet, its energy costs increase rapidly. Thus, 10 P are extracted (4 + 3 + 2 + 1). Remember that the rarity determines the suitability for extraction to be reduced by one each time its amount's extractions are executed.



Conquered energy calculation and years aged

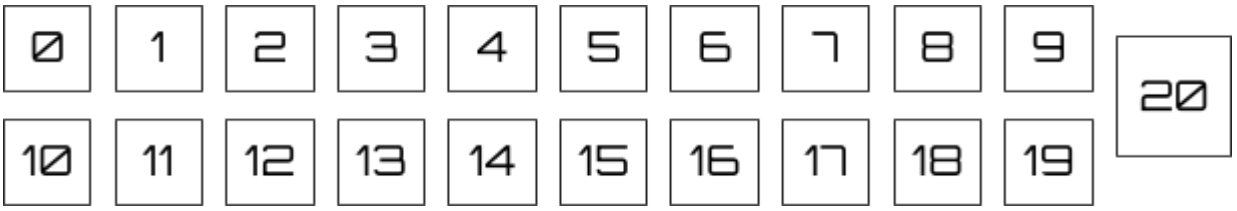
The Dragonflies are designed to get automatically the energy they need to make the trip as they explore the planets. In theory, energy is obtained very efficiently. However, the Dragonfly 3RR0R, due to the breakdown, does not have the same ease.

The more statements are designed to get the resources needed, less energy is returned to the Dragonfly 3RR0R. A ten of energy is obtained by 7 minus the number of statements constructed in the exploration phase. Syntactically poorly constructed statements are used in this calculation. Thus, each sequence of symbols that form a statement, even if it is not effectively enforced by Dragonfly 3RR0R, undertake to obtain energy.

Formula: 7 - Statements (tens of energy).

Example: During exploitation, Dragonfly 3RR0R finished with 8 energy. Graduates codes totaled 3 syntactically correct statements and 1 incorrect. The sum is four, so that it returns 30 units of energy. (7-3) x 10 = 30.

The aging of the crew is the greater the less Cryogenics Resources (N, He and Re) Dragonfly 3RR0R own. To do the reckoning, add up your scores and advance a number of years equal to 13 - the sum of the scores. Aging is registered in the following section of the board:



Formula: 13 - (N + He + Re) (years).

Example: After exploration, the Dragonfly 3RR0R got 31 N units, 47 of He and 35 of Re. As the sum of the dozens is 10, players age 3 years (13-10 = 3). Assuming that they had already sumed seven years of age, they have now aged 10 years more then their beloved.

End of the game and victory condition

After this phase, begins a new round until seven rounds are reached, when the game ends. Remember that at the beginning of each stage three of every round there is the switching of the first player clockwise.

Players win if they aged less than 20 years. Remember that the game also ends if P reaches 0, in which case the players lose.