Challenge TUR-ROX-Y : Write up

Installation du jeu ZZT, ouverture du fichier

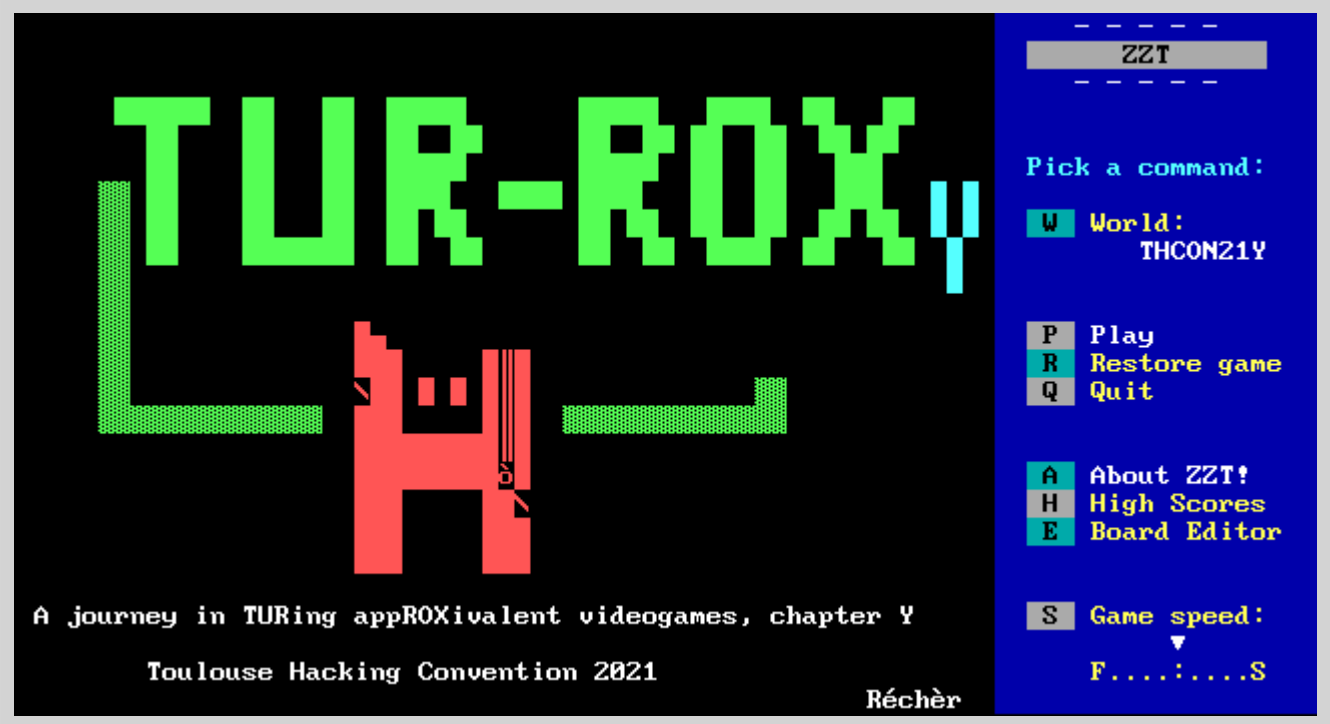
Le challenge se présente sous la forme d'un unique fichier THCON21Y.ZZT.

Effectuez une recherche sur internet, soit sur l'extension ZZT, soit avec tous les mots de la description : "ZZT", "MUSEUM", "ZETA". Vous devriez arriver sur cette page : museumofzzt.com/zeta

Suivez les instructions du lien pour installer ZZT et Zeta (pas besoin de KevEdit). Le fichier THCON21Y.ZZT est un "world" du jeu vidéo ZZT, placez-le dans le répertoire de Zeta, puis lancez le jeu.

Après avoir passé le message initial, appuyez sur "W" pour charger le world et sélectionnez THCON21Y.ZZT.

L'image de présentation permet de confirmer que le fichier a été bien chargé, et qu'il concerne bien notre challenge.



Vous pouvez ouvrir le world avec l'éditeur de niveau (appuyez sur "E"), mais le mieux est de commencer par le découvrir en jouant (appuyez sur "P").

Exploration du jeu

Le premier "board" (c'est le nom donné à une salle dans le jeu) indique le format du flag et sa première lettre : **THCon21{Y ... }**.

Le board à droite contient uniquement un message promotionnel pour mon super-projet de création et partage de jeux vidéos : http://squarity.fr. N'hésitez pas à l'essayer (mais ça n'a rien à voir avec le challenge).

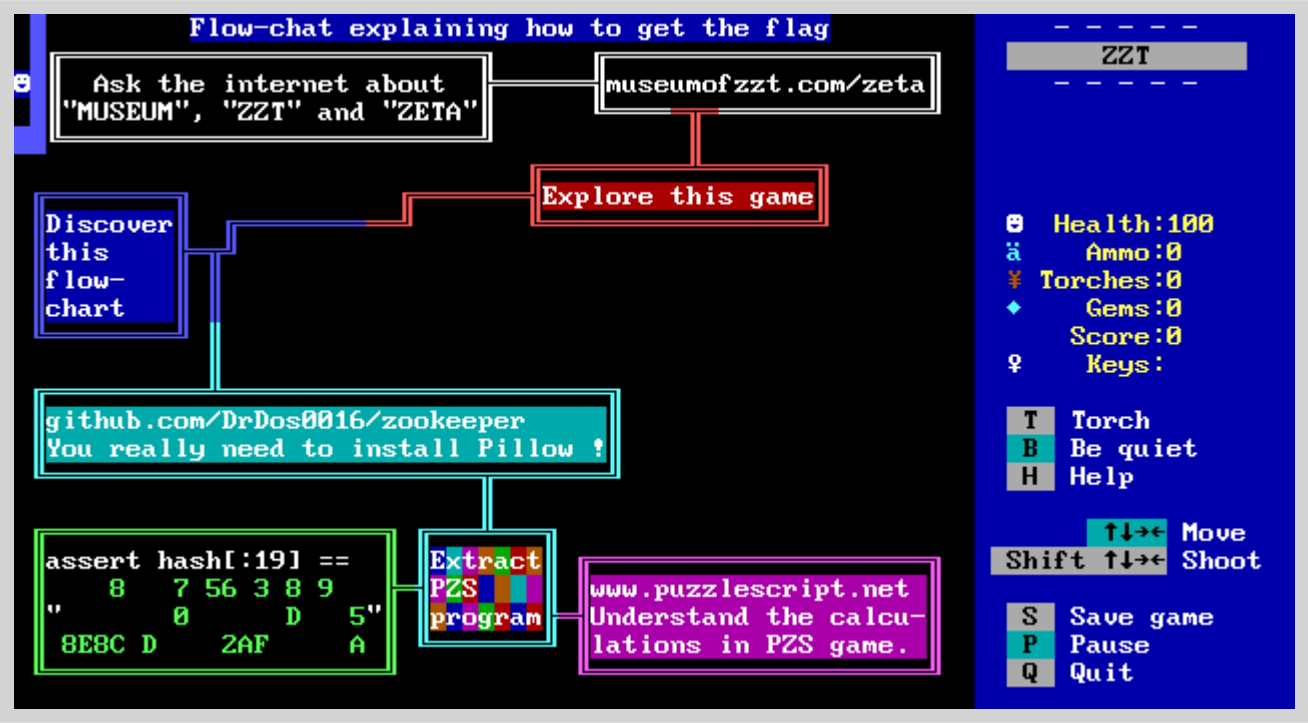
Le passage à gauche mène vers une série de board affichant du texte appelé "PuzzleScript". Les couleurs de fond du texte constituent un patchwork assez horrible à regarder. En explorant jusqu'au bout, on compte 26 pages numérotées, certaines prenant toute la largeur de l'écran, d'autres sur une moitié d'écran.

Le board tout en haut contient un code en python.

```
pzs_code = <code from all PuzzleScript pages>
pzs_code = pzs_code.strip()
assert " \n" not in pzs_code
assert "\n\n" not in pzs_code
assert len(pzs_code) == 7214
assert "\r\n" not in pzs_code
import hashlib
b_pzs = bytes(pzs_code, encoding="ascii")
hash = hashlib.sha256(b_pzs).hexdigest().upper()
print("Here is a part of the flag : ")
print("THCon21{-" + hash[-2:] + "----}")
```

Les lettres "g" et "j" du texte s'animent pour manger le code. C'est très amusant.

Il reste un board un peu plus difficile à trouver : à partir du premier board, allez directement en bas. Le passage n'est pas immédiatement visible, car il se trouve dans le coin, et n'est accessible que par une seule case. Mais vous pouvez aussi passer par le board PuzzleScript voisin.



Ce board contient une espèce de flow-chart indiquant, en très résumé, les étapes à effectuer pour réussir le challenge. On retrouve le lien d'installation de Zeta.

Le titre tout en haut comporte une faute à "flow-chart". Je ne l'ai vue que trop tard. Tant pis !

Il y a également deux liens :

- github.com/DrDos0016/zookeeper : librairie python permettant de parser des fichiers .ZZT
- www.puzzlescript.net : un outil de création de mini-jeu en ligne. Le code utilisé pour créer les mini-jeux ressemble fortement aux textes dans les pages PuzzleScript.

Le cadre en bas à gauche est une ligne de code affichant le début de la variable `hash` (définie dans le code python précédent). Les caractères du hash se mélangent, c'est à nouveau très amusant.

Exploration du jeu avec l'éditeur de niveau

L'affichage des boards par l'éditeur de niveau intégré à ZZT permet de trouver le flow-chart, si vous ne l'aviez pas vu avant. Cela permet également de récupérer plus sereinement le code python et le début du hash (mais un simple screenshot pendant le jeu peut aussi faire l'affaire).

Il existe un "useless board", qui, comme son nom l'indique, est inutile pour le challenge. Pour info, on ne peut pas supprimer de boards avec l'éditeur de ZZT.

Contrairement au challenge du chapitre Z, l'éditeur de niveau n'apporte rien de plus. Il n'y pas d'objets programmables, à part les lettres qui bougent.

De toutes ces explorations, il faut en déduire les choses suivantes :

- Les textes PuzzleScript doivent être extraits et réarrangés selon les numéros des pages, pour obtenir un "pzs-code".
- Ce pzs-code ne comporte pas d'espace en fin de ligne, pas de ligne vide, les sauts de ligne sont au format "\n" et non pas "\r\n".
- Ce pzs-code a une taille de 7214 caractères, encodés en ascii.
- Il faut calculer son hash sha-256 et récupérer les deux derniers caractères pour obtenir une partie du flag.
- Le début de ce hash est : "8E8C8D70562AF3D859A". Cela permettra de vérifier qu'on ne s'est pas trompé lors de l'extraction des textes.
- Il faudra utiliser ce pzs-code sur le site puzzlescript.net, pour obtenir un mini-jeu qui constituera la suite du challenge.

Extraction du "pzs-code"

Il est possible d'employer diverses manières bourrines :

- faire des screenshots de toutes les pages et les passer dans un OCR,
- tout recopier à la main,
- ouvrir le fichier THCON21Z.ZZT en mode binaire, identifier les endroits où se trouvent les caractères et écrire un petit programme pour les extraire.

Le challenge suggère une librairie python pour parser le fichier : Zookeeper. Elle est un peu artisanale, et nécessite obligatoirement la librairie "Pillow", bien qu'on ne fera pas de traitement d'image. Si vous ne l'installez pas, vous obtiendrez un message d'erreur absolument pas explicite.

Donc :

```
python3 -m pip install --upgrade Pillow
```

Les codes d'exemple dans le repository de zookeeper expliquent comment ouvrir un fichier ZZT et comment itérer sur les boards.

La documentation permet de trouver la fonction pour récupérer une case dans un board (`Board.get_element(x, y)`), ainsi que l'attribut contenant le numéro du caractère (`Element.character`).

Comme son nom ne l'indique pas, `Element.character` est une valeur numérique, à convertir en texte. L'encodage utilisé dans ZZT est le "CP437". Mais si vous vous contentez de faire des conversions en ascii, c'est suffisant, car il n'y a que de l'ascii dans le code PuzzleScript.

Voici le programme d'extraction utilisant zookeeper. Il faut le copier, ainsi que le fichier THCON21Y.ZZT, dans la racine du repository zookeeper, puis l'exécuter.

Il effectue les actions suivantes :

- Lecture de tous les boards du fichier.
- Affichage, sur la sortie standard, des textes de tous les boards.
- Calcul du hash selon les directives indiquées, vérification du début du hash.
- Récupération de la partie du flag se trouvant dans le hash et affichage de ce flag.
- Écriture, dans le fichier "thcon_pzs_reconstruct.txt", de tout le code PuzzleScript extrait.

```
import zookeeper.zookeeper

PAGE_DEFINITIONS = [
    # large page
    {
        "page_x": 2,
        "page_y": 4,
        "label_x": 7,
        "label_y": 2,
        "line_length": 56,
        "line_qty": 19,
    },
    # small page 1
    {
        "page_x": 2,
        "page_y": 4,
        "label_x": 4,
        "label_y": 2,
        "line_length": 26,
        "line_qty": 19,
    },
    # small page 2
    {
        "page_x": 32,
        "page_y": 4,
        "label_x": 34,
        "label_y": 2,
        "line_length": 26,
```

```
        "line_qty": 19,
    },
]
CHECK_LABEL = "PuzzleScript Page"


def parse_board_text(lines):
    """
    Analyse toutes les lignes de texte provenant d'un board.
    Renvoit une liste de 0, 1 ou 2 éléments, correspondants aux pages PuzzleScript trouvées.
    """
    code_pages = []
    for page_def in PAGE_DEFINITIONS:

        # Ça c'est moche, mais osef.
        keys = ("page_x", "page_y", "label_x", "label_y", "line_length", "line_qty")
        vals = [page_def[key] for key in keys]
        page_x, page_y, label_x, label_y, line_length, line_qty = vals

        line_label = lines[label_y]
        label_x_end = label_x + len(CHECK_LABEL)
        label_value = line_label[label_x:label_x_end]
        if label_value != CHECK_LABEL:
            continue
        page_index = line_label[label_x_end : label_x_end + 3]
        page_index = page_index.strip()
        if not page_index.isdigit():
            continue

        page_index = int(page_index)
        code_lines = lines[page_y : page_y + line_qty]
        code_lines = [
            line[page_x : page_x + line_length].strip() for line in code_lines
        ]
        code_pages.append((page_index, code_lines))

    if code_pages:
        print(f"!! number of code pages found : {len(code_pages)}")
    return code_pages


def get_pzs_code():
    """
    Ouvre le fichier THCON21Y.ZZT, extrait toutes les pages de code PuzzleScript,
    puis les réordonne, et renvoie tout le code sous forme d'une grande string.
    """
    z = zookeeper.zookeeper.Zookeeper("THCON21Y.ZZT")
    # all_code_pages est une liste de tuple de 2 elem :
    # le numéro de la page, les lignes de code de la page.
    all_code_pages = []
    for board in z.boards:
        print(board.title)
        print("-" * 40)
        board_lines = []
        for y in range(1, 26):
            val_bytes = []
            for x in range(1, 61):
                elem = board.get_element((x, y))
                val_bytes.append(elem.character)
```

```
            line = bytes(val_bytes).decode("CP437")
            print(line)
            board_lines.append(line)
        code_pages = parse_board_text(board_lines)
        all_code_pages.extend(code_pages)

    print(f"Number of total code pages : {len(all_code_pages)}")
    all_code_pages.sort()
    # On vérifie qu'il n'y a pas de trous dans les numéros de pages.
    page_indexes = [page_index for page_index, code_page in all_code_pages]
    assert page_indexes == list(range(1, len(all_code_pages) + 1))

    all_code_puzzlescript = []
    for page_index, code_page in all_code_pages:
        all_code_puzzlescript.extend(code_page)
    pzs_code = "\n".join(all_code_puzzlescript)
    pzs_code = pzs_code.strip()
    return pzs_code


def check_and_show_flag(pzs_code):
    """
    Vérification du pzs_code.
    Cette fonction est un copié-collé des codes python que l'on trouve dans le challenge.
    """
    # pzs_code = <code from all PuzzleScript pages>
    pzs_code = pzs_code.strip()
    assert " \n" not in pzs_code
    assert "\n\n" not in pzs_code
    assert len(pzs_code) == 7214
    assert "\r\n" not in pzs_code
    import hashlib

    b_pzs = bytes(pzs_code, encoding="ascii")
    hash = hashlib.sha256(b_pzs).hexdigest().upper()

    print(hash)
    assert hash[:19] == "8E8C8D70562AF3D859A"

    print("Here is a part of the flag : ")
    print("THCon21{-" + hash[-2:] + "----}")


def main():
    pzs_code = get_pzs_code()
    check_and_show_flag(pzs_code)
    with open("thcon_pzs_reconstruct.txt", "w", encoding="ascii") as out_file:
        out_file.write(pzs_code)


if __name__ == "__main__":
    main()
```

Et on récupère une nouvelle partie du flag : **THCon21{-95----}**

Le code PuzzleScript extrait se trouve à la fin du write-up.

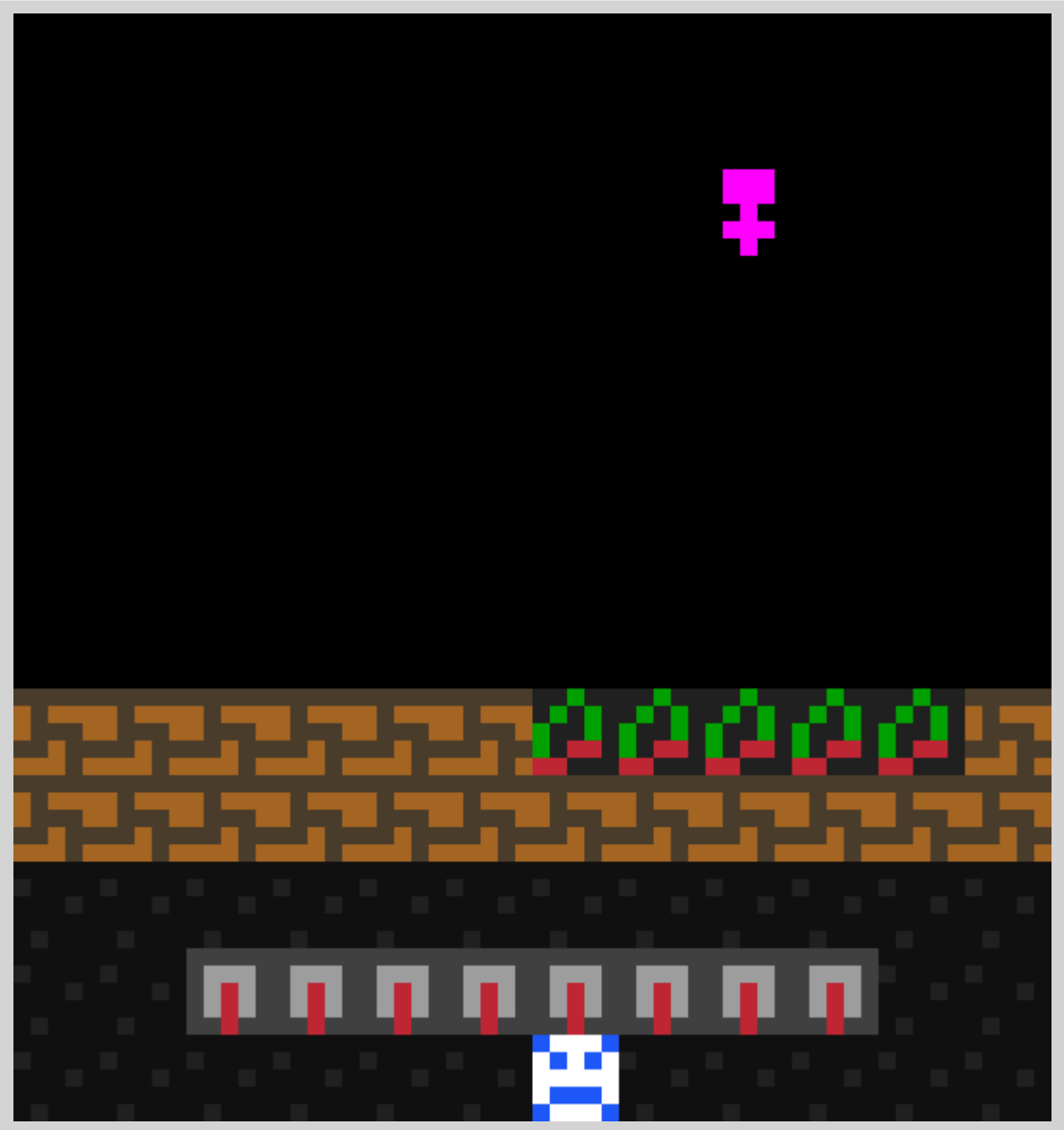
Lancement du jeu PuzzleScript

Ouvrez le fichier thcon_pzs_reconstruct.txt, copiez tout son contenu.

Allez sur puzzlescript.net, cliquez sur "make a game". Collez le contenu dans la zone de texte à gauche, cliquez sur le bouton "run" dans le menu en haut.

Pour démarrer une nouvelle partie, appuyez sur la touche "X".

Un message indique que vous devez activer les manettes pour faire apparaître 5 "grappes de raisin violettes".



Déplacez le personnage avec les flèches. Lorsque vous allez sur l'une des 8 manettes, celle-ci change d'état (rouge/vert), et certains des 5 objets au-dessus peuvent changer.

Ces objets sont censés représenter des fruits, même si c'est difficile à dessiner sur des carrés de 5x5 pixels. Nous avons donc :

- des cerises,
- une banane,
- un fruit rouge quelconque, mettons que ce soit une pomme,
- une orange,
- un triangle violet censé être la fameuse grappe de raisin,
- une espèce de mangue verte-orange-rouge,

- une poire verte.

Le premier niveau est assez simple, et sert juste à montrer le fonctionnement global du jeu. Pour le gagner : activez une manette, si celle-ci fait apparaître une grappe de raisin, gardez-la, sinon désactivez-la. En les testant toutes, vous trouverez les 5 manettes faisant chacune apparaître une grappe.

Un message s'affiche, vous indiquant d'aller récupérer le "venus symbol". Il s'agit de l'objet rose en haut du jeu. Le fait d'avoir fait apparaître les 5 grappes a supprimé le mur tout à droite. Vous pouvez donc aller sur le symbole et gagner ce niveau.

Petite info culturelle : les clés dans le jeu ZZT sont représentées par des symboles de venus (caractère "♀"). Il y en a de différentes couleurs, et dans les jeux officiels, les clés principales sont toujours roses. Ce "venus symbol" est un petit hommage.

Le niveau suivant comporte deux caractères au milieu de l'aire de jeu. On comprend assez vite qu'ils affichent la valeur hexadécimale correspondant à la valeur binaire de la combinaison des manettes.

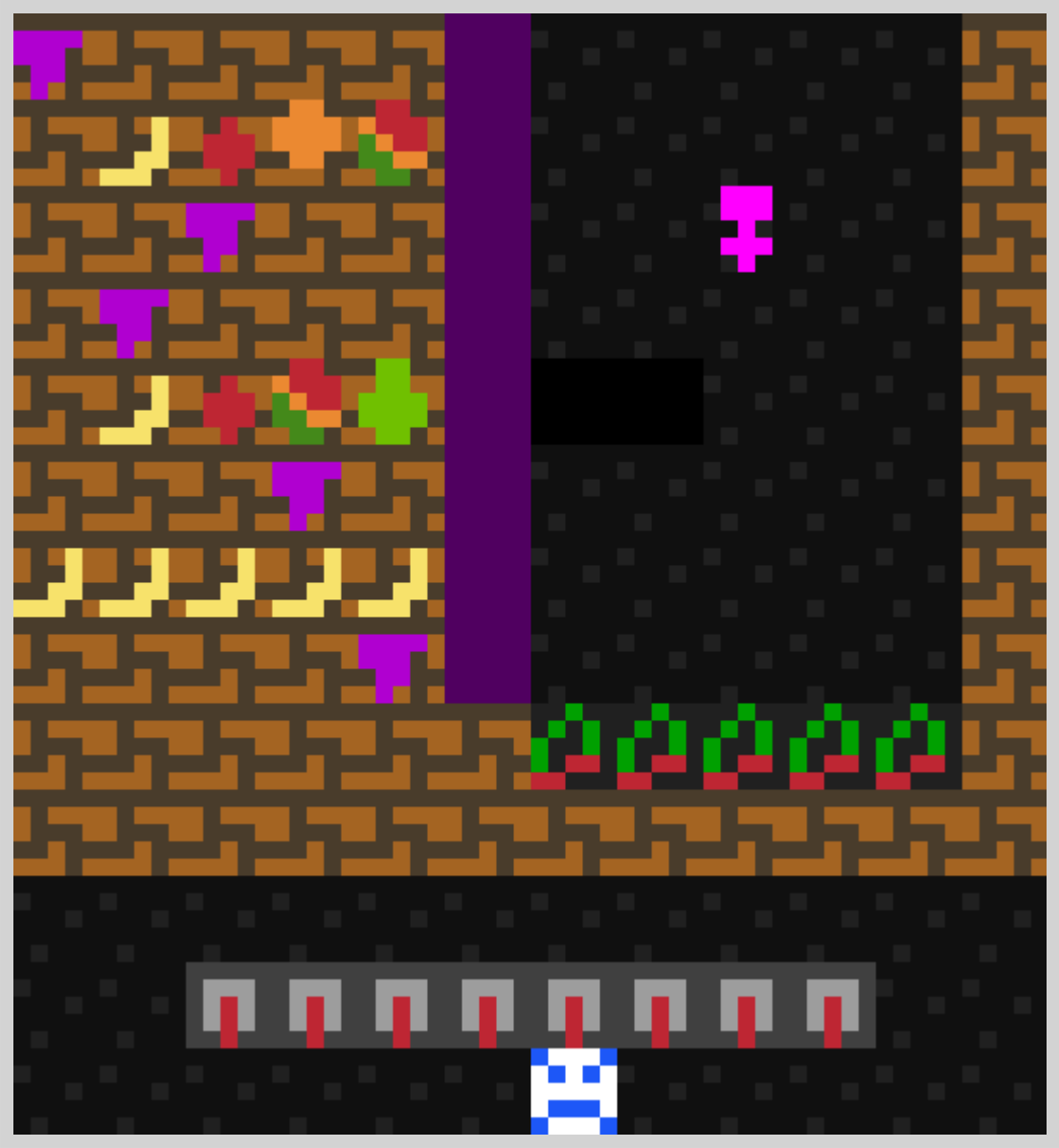
Ce niveau fonctionne sur le même principe que le premier, mais les manettes modifient les 5 fruits de manière bien plus compliquée. Il est difficile de trouver la bonne combinaison par des essais/erreurs.

Analyse plus détaillée du jeu PuzzleScript

Indice 1

Le début du script du jeu contient un objet intitulé `change_the_black_below_to_transparent_to_debug`. Modifiez le texte à la ligne juste en-dessous, écrivez `transparent` à la place de `black`, puis relancez le jeu et démarrez une nouvelle partie.

Le fonctionnement interne est maintenant un peu plus visible.



Chaque manette correspond à une combinaison de fruits affichées à gauche, sur une ligne. En activant une seule manette, on voit que la combinaison se retrouve telle quelle dans le résultat final. Les cerises ne sont pas affichées dans les combinaisons initiales, mais on va dire que c'est un "détail d'implémentation".

Pour le niveau 2, on voit aussi les combinaisons initiales. Elles sont plus compliquées que le niveau 1, et dès qu'on active plus d'une manettes, on ne comprend pas comment est déterminée la combinaison finale.

Indice 2

Dans le script, les noms des objets représentants les fruits sont des nombres.

- cerise : _0
- banane : _1
- pomme : _2
- orange : _3
- raisins : _4
- mangue : _5
- poire : _6

Indice 3

Les plans des levels se trouvent à la fin du script. Les lettres "T", "H", "C", "O", "N", "P", "Z" correspondent chacune à un fruit.

Modifiez les combinaisons du niveau 1 pour en mettre des plus simples, puis relancez le jeu.

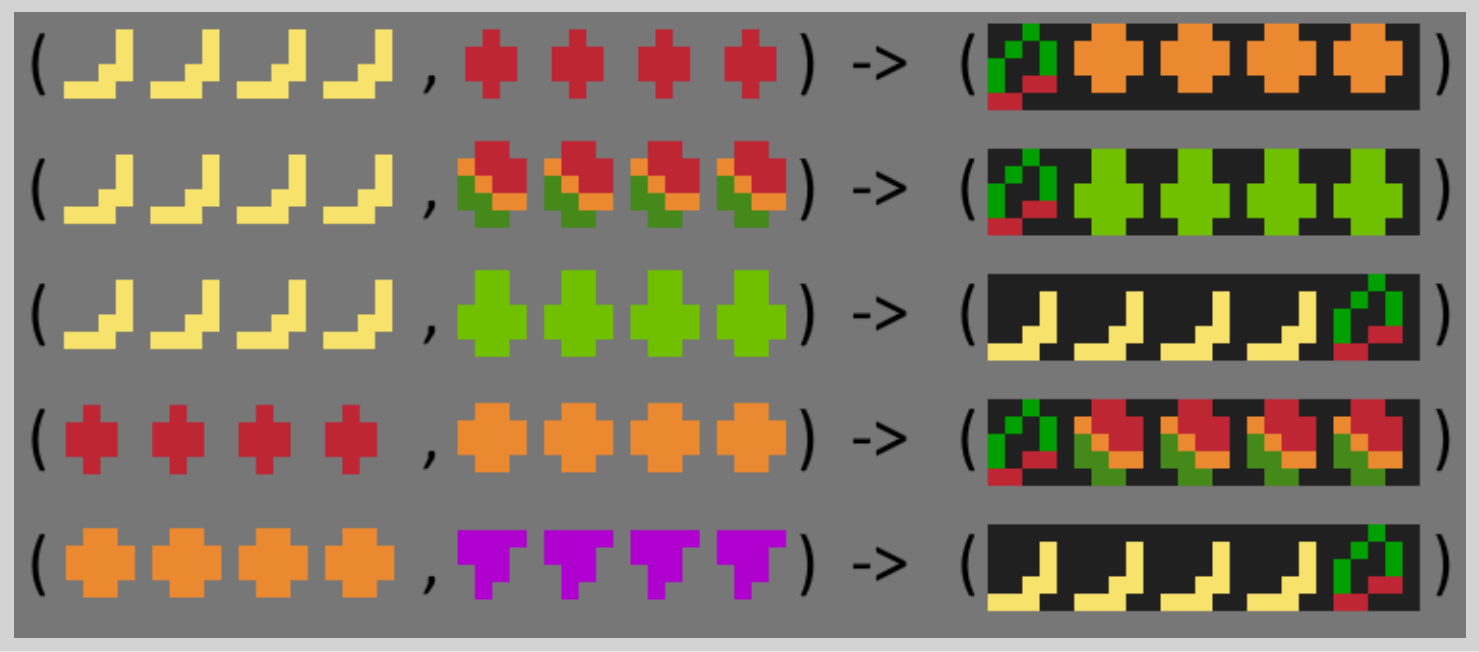
Par exemple, vous pourriez redéfinir le premier niveau comme ceci :

```
TTTTT0....XW
THHHH1.....W
TCCCC2..U..W
TOOOO3.....W
TNNNN4*%...W
TPPPP5.....W
TZZZZ6.....W
TTTTT7.....W
WWWWWW/////W
WWWWWWWWWWWW
............
.A01234567..
......F.....
```

Activez différentes manettes, pas plus de 2 à la fois, afin d'avoir des situations les plus simples et les plus compréhensibles possibles.

Après modification, certaines activations de manettes peuvent faire bugger le jeu, et font apparaître un fruit supplémentaire à gauche de la combinaison de résultat. C'est un autre "détail d'implémentation".

Voici quelques exemples de combinaisons que vous pourriez essayer, et qui peuvent aider à comprendre :



Déduction du fonctionnement

Les indices précédents devraient vous permettre de déduire que les fruits correspondent à des nombres, et que l'opération réalisée est une addition en base 7.

Exemple :


```
[pomme pomme orange orange] + [pomme poire pomme mangue] + [orange cerise mangue banane rais
[2233 en base 7] + [2625 en base 7] + [30514 en base 7] =
808 + 999 + 7459 =
9266 =
[36005 en base 7] =
[orange poire cerise cerise mangue]
```

Résolution du niveau 2

Convertissez toutes les combinaisons de fruits vers leurs valeurs numériques correspondantes.

fruits	valeur en base 7	valeur en base 10
orange raisin banane pomme	"3412"	1234
pomme pomme orange orange	"2233"	808
pomme raisin pomme mangue	"2425"	901
pomme poire pomme mangue	"2625"	999

fruits	valeur en base 7	valeur en base 10
banane pomme pomme banane	"1221"	456
pomme pomme pomme raisin	"2224"	802
raisin pomme raisin orange	"4243"	1501
orange cerise mangue banane raisin	"30514"	7459

Le résultat final à obtenir est "5 fois le raisin" -> "44444" en base 7 -> 11204

Réussir le niveau 2 est donc équivalent à résoudre le problème suivant :

Sélectionnez les bonnes valeurs parmi (1234, 808, 901, 999, 456, 802, 1501, 7459), de façon à ce que la somme fasse 1204

Il y a sûrement une élégante méthode mathématique pour cela. Mais le plus simple est de tester les 256 possibilités à la bourrin, avec un petit code python, que voici :

```
fruit_values = (1234, 808, 901, 999, 456, 802, 1501, 7459)

for selection in range(256):
    str_selection = bin(selection)[2:]
    selected_values = [
        val
        for bit, val in
        zip(str_selection, fruit_values)
        if bit == "1"
    ]
    if sum(selected_values) == 11204:
        print("On a trouvé une solution !!!")
        print(selected_values)
        print(str_selection)
```

Le code affiche une seule solution (et heureusement, sinon ce serait un peu embêtant). Les nombres à additionner sont : [1234, 808, 901, 802, 7459]. Les manettes doivent donc être activées comme ceci : 11100101

Après avoir mis la bonne combinaison, un message indique où placer les deux chiffres hexadécimaux dans le flag. On récupère donc une nouvelle partie du flag : **THCon21{---E5--}**

Résolution du niveau 3

Même principe, avec des combinaisons de fruits différentes.

fruits	valeur en base 7	valeur en base 10
banane cerise banane pomme pomme	"10122"	2466
banane cerise cerise banane raisin	"10014"	2412
mangue banane raisin pomme	"5142"	1794
mangue banane poire orange	"5163"	1809
banane cerise banane orange cerise	"10130"	2471
mangue orange banane pomme	"5312"	1871
mangue raisin mangue pomme	"5453"	1949
mangue mangue orange orange	"5533"	1984

Dans le code python du chapitre précédent, remplacez la première ligne par :

```
fruit_values = (2466, 2412, 1794, 1809, 2471, 1871, 1949, 1984, )
```

Re-exécutez le code, qui affiche à nouveau une seule solution. Les nombres à additionner sont : [2466, 2412, 2471, 1871, 1984]. Les manettes doivent être activées comme ceci : 11001101

La dernière partie du flag est "CD".

Conclusion

Il ne reste plus qu'à tout assembler. La valeur finale du flag est **THCon21{Y95E5CD}**

Réchèr.

Le script complet du jeu PuzzleScript

Attention, c'est long !

```
title THCon21 - TUR-ROX-Y
author Recher
run_rules_on_level_start
noaction
========
OBJECTS
========
change_the_black_below_to_transparent_to_debug
Black
Background
#202020 #101010
11111
01111
11101
11111
10111
Wall
brown darkbrown
11111
01000
01110
11011
00010
b
#500060
no_block
#F0E0E0
player
blue white
01110
10101
11111
10001
01110
_0
red #00A000
..1..
.1.1.
1..1.
1.00.
00...
_1
```

```
yellow
.....
...0.
...0.
..00.
000..
_2
red
.....
..0..
.000.
.000.
..0..
_3
orange
.00..
0000.
0000.
.00..
.....
_4
#B000D0
.....
0000.
000..
.00..
.0...
_5
red orange green
.00..
1000.
2100.
2211.
.22..
_6
#70C000
.00..
.00..
0000.
0000.
.00..
d
transparent
s0
transparent
s1
transparent
s2
transparent
s3
transparent
s4
transparent
s5
transparent
s6
transparent
s7
transparent
```

```
A
transparent
lever_down
#404040 red gray
00000
02220
02120
02120
00100
lever_up
#404040 #00E000 gray
00100
02120
02120
02220
00000
rv
transparent
N1
White
...0.
..00.
...0.
...0.
...0.
N2
White
..00.
.0..0
...0.
..0..
.0000
N3
White
..00.
.0..0
...0.
.0..0
..00.
N4
White
.0..0
.0..0
.0000
....0
....0
N5
White
.0000
.0...
.000.
....0
.000.
N6
White
..00.
.0...
.000.
.0..0
```

..00.
N7
White
.0000
....0
...0.
..0..
..0..
N8
White
..00.
.0..0
..00.
.0..0
..00.
N9
White
..00.
.0..0
..000
....0
..00.
N0
White
..00.
.0..0
.0..0
.0..0
..00.
NA
White
..00.
.0..0
.0000
.0..0
.0..0
NB
White
.000.
.0..0
.000.
.0..0
.000.
NC
White
..00.
.0...
.0...
.0...
..00.
ND
White
.000.
.0..0
.0..0
.0..0
.000.
NE
White

.0000
.0...
.000.
.0...
.0000
NF
White
.0000
.0...
.000.
.0...
.0...
v_t
transparent
v_u
transparent
final_val
#202020
won
transparent
rmv_w
transparent
venus_symbol
#FF00FF
.000.
.000.
..0..
.000.
..0..
g
#B000D0
..0..
.000.
0.0.0
..0..
..0..
lvl_1
transparent
lvl_1p
transparent
hide_digi_lvl_1
Black
msg_won_1
transparent
msg_won_1p
transparent
=======
LEGEND
=======
. = Background
T = _0 and Wall
H = _1 and Wall
C = _2 and Wall
O = _3 and Wall
N = _4 and Wall
P = _5 and Wall
Z = _6 and Wall
F = Background and Player
elem = _0 or _1 or _2 or _3 or _4 or _5 or _6

```
no_4 = _0 or _1 or _2 or _3 or _5 or _6
gr_t_1 = _2 or _3 or _4 or _5 or _6
gr_t_2 = _3 or _4 or _5 or _6
gr_t_3 = _4 or _5 or _6
gr_t_4 = _5 or _6
gr_t_5 = _6
W = Wall
0 = s0
1 = s1
2 = s2
3 = s3
4 = s4
5 = s5
6 = s6
7 = s7
select = s0 or s1 or s2 or s3 or s4 or s5 or s6 or s7
lvr = lever_down or lever_up
digi1 = N0 or N1 or N2 or N3 or N4 or N5 or N6 or N7
digi2 = N8 or N9 or NA or NB or NC or ND or NE or NF
digi = digi1 or digi2
* = v_t
% = v_u
Sh = v_t or v_u
/ = final_val
U = venus_symbol
X = lvl_1
Y = lvl_1p
dbg = change_the_black_below_to_transparent_to_debug
=======
SOUNDS
================
COLLISIONLAYERS
================
Background
s0,s1,s2,s3,s4,s5,s6,s7
g
Wall, player,v_t,v_u,b,no_block,final_val
_0
_1
_2
_3
_4
_5
_6
d
change_the_black_below_to_transparent_to_debug
lever_down, lever_up
A,rv
N0
N1
N2
N3
N4
N5
N6
N7
N8
N9
NA
```

```
NB
NC
ND
NE
NF
rmv_w
venus_symbol
won,lvl_1,lvl_1p,hide_digi_lvl_1,msg_won_1,msg_won_1p
======
RULES
======
[ A ][lvl_1][Sh] -> [ A ][lvl_1][Sh hide_digi_lvl_1]
RIGHT [A|...|select] ->  [A|...|select lever_down]
[ A ][ select no lever_down no b ] -> [ A ][ select b ]
[ A ][ s0 lvr ]->[ A ][ s0 lvr v_t]
[ A ][ s4 lvr ]->[ A ][ s4 lvr v_u]
[ A ][ b ] -> [ A ][ b dbg ]
HORIZONTAL[ A ][ dbg | ] -> [ A ][ dbg | dbg ]
[ A ] -> [ ]
[> Player|lvr s0][no lvr s0]->[Player|lvr s0 rv][s0 rv]
[> Player|lvr s1][no lvr s1]->[Player|lvr s1 rv][s1 rv]
[> Player|lvr s2][no lvr s2]->[Player|lvr s2 rv][s2 rv]
[> Player|lvr s3][no lvr s3]->[Player|lvr s3 rv][s3 rv]
[> Player|lvr s4][no lvr s4]->[Player|lvr s4 rv][s4 rv]
[> Player|lvr s5][no lvr s5]->[Player|lvr s5 rv][s5 rv]
[> Player|lvr s6][no lvr s6]->[Player|lvr s6 rv][s6 rv]
[> Player|lvr s7][no lvr s7]->[Player|lvr s7 rv][s7 rv]
[ lever_down rv ] -> [ lever_up ]
[ lever_up rv ] -> [ lever_down ]
[ b rv ] -> [ no_block ]
[ no_block rv ] -> [ b ]
[digi]->[]
RIGHT[Sh lever_up]->[Sh lever_up N8]
RIGHT[Sh | lever_up]->[Sh N4 | lever_up]
RIGHT[Sh | | lever_up]->[Sh N2 | | lever_up]
RIGHT[Sh | | | lever_up]->[Sh N1 | | | lever_up]
[N8 N4]->[NC]
[N4 N2]->[N6]
[N8 N2]->[NA]
[NC N2]->[NE]
[N2 N1]->[N3]
[N4 N1]->[N5]
[N6 N1]->[N7]
[N8 N1]->[N9]
[NA N1]->[NB]
[NC N1]->[ND]
[NE N1]->[NF]
[Sh lvr no digi]->[Sh lvr N0]
[v_t digi lvr][v_t no lvr]->[v_t lvr][v_t digi no lvr]
[v_u digi lvr][v_u no lvr]->[v_u lvr][v_u digi no lvr]
[ elem no Wall ] -> [ ]
RIGHT[_0|no b|no b|no b|no b|no b|no b]->[_0||||||_0]
RIGHT[_1|no b|no b|no b|no b|no b|no b]->[_1||||||_1]
RIGHT[_2|no b|no b|no b|no b|no b|no b]->[_2||||||_2]
RIGHT[_3|no b|no b|no b|no b|no b|no b]->[_3||||||_3]
RIGHT[_4|no b|no b|no b|no b|no b|no b]->[_4||||||_4]
RIGHT[_5|no b|no b|no b|no b|no b|no b]->[_5||||||_5]
RIGHT[_6|no b|no b|no b|no b|no b|no b]->[_6||||||_6]
[ _0 ] -> []
```

```
+ [ _1 gr_t_1 no d ] -> [    gr_t_1 d ]
+ [ _2 gr_t_2 no d ] -> [ _1 gr_t_2 d ]
+ [ _3 gr_t_3 no d ] -> [ _2 gr_t_3 d ]
+ [ _4 gr_t_4 no d ] -> [ _3 gr_t_4 d ]
+ [ _5 gr_t_5 no d ] -> [ _4 gr_t_5 d ]
+ DOWN [ _1 | no Wall no _1 ] -> [     | _1 ]
+ DOWN [ _2 | no Wall no _2 ] -> [     | _2 ]
+ DOWN [ _3 | no Wall no _3 ] -> [     | _3 ]
+ DOWN [ _4 | no Wall no _4 ] -> [     | _4 ]
+ DOWN [ _5 | no Wall no _5 ] -> [     | _5 ]
+ DOWN [ _6 | no Wall no _6 ] -> [     | _6 ]
+ DOWN [ _1 | no Wall _1 no d ] -> [    | _1 d ]
+ DOWN [ _2 | no Wall _2 no d ] -> [ _1 | _2 d ]
+ DOWN [ _3 | no Wall _3 no d ] -> [ _2 | _3 d ]
+ DOWN [ _4 | no Wall _4 no d ] -> [ _3 | _4 d ]
+ DOWN [ _5 | no Wall _5 no d ] -> [ _4 | _5 d ]
+ DOWN [ _6 | no Wall _6 no d ] -> [ _5 | _6 d ]
+ [ _5 no gr_t_5 d ] -> [ _6 ]
+ [ _4 no gr_t_4 d ] -> [ _5 ]
+ [ _3 no gr_t_3 d ] -> [ _4 ]
+ [ _2 no gr_t_2 d ] -> [ _3 ]
+ [ _1 no gr_t_1 d ] -> [ _2 ]
+ [ no elem d ] -> [ _1 ]
+ RIGHT [ no d | _6 d ] -> [ d | ]
[ final_val no elem ] -> [ final_val _0 ]
[ venus_symbol ] -> [ venus_symbol won ]
[ final_val no_4 ][ won ] -> [ final_val no_4 ][ ]
RIGHT[ won | ... | Wall ] -> [ won | ... | rmv_w ]
[won][rmv_w|...|Wall|no Wall]->[won][rmv_w|...|Wall g|]
VERTICAL[won][rmv_w|...|Wall] -> [won][rmv_w|...|]
[> Player][won][lvl_1]->[> Player][won][msg_won_1]
[msg_won_1]->[]message Great! Get the venus symbol
[> Player][won][lvl_1p]->[> Player][won][msg_won_1p]
[msg_won_1p]->[]message Write down the 2 hex digits.
late [ Player dbg ] -> [ Player ]
==============
WINCONDITIONS
==============
all Player on venus_symbol
=======
LEVELS
=======
message Use the levers to make 5 purple grapes.
NTTTT0....XW
THCOP1.....W
TTNTT2..U..W
TNTTT3.....W
THCPZ4*%...W
TTTNT5.....W
HHHHH6.....W
TTTTN7.....W
WWWWWW/////W
WWWWWWWWWWWW
............
.A01234567..
......F.....
message Same thing, but more complicated.
TONHC0....YW
TCCOO1.....W
```

```
TCNCP2..U..W
TCZCP3.....W
THCCH4.....W
TCCCN5*%...W
TNCNO6.....W
OTPHN7.....W
WWWWWW/////W
WWWWWWWWWWWW
............
.A01234567..
......F.....
message Good! flag part : THCon21{---XX--}
message Replace the two Xs with the digits you noted.
HTHCC0....YW
HTTHN1.....W
TPHNC2..U..W
TPHZO3.....W
HTHOT4.....W
TPOHC5*%...W
TPNPC6.....W
TPPOO7.....W
WWWWWW/////W
WWWWWWWWWWWW
............
.A01234567..
......F.....
message Good! flag part : THCon21{-----XX}
message Replace the two Xs with the digits you noted.
message The end. No more flag part here.
```