Challenge TUR-ROX-Z : Write up

Ouverture du fichier

Le challenge se présente sous la forme d'un unique fichier THCON21Z.ZZT.

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

Suivez les instructions pour installer ZZT et Zeta (pas besoin de KevEdit). Le fichier THCON21Z.ZZT est un "world" du jeu vidéo ZZT, copiez-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 THCON21Z.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

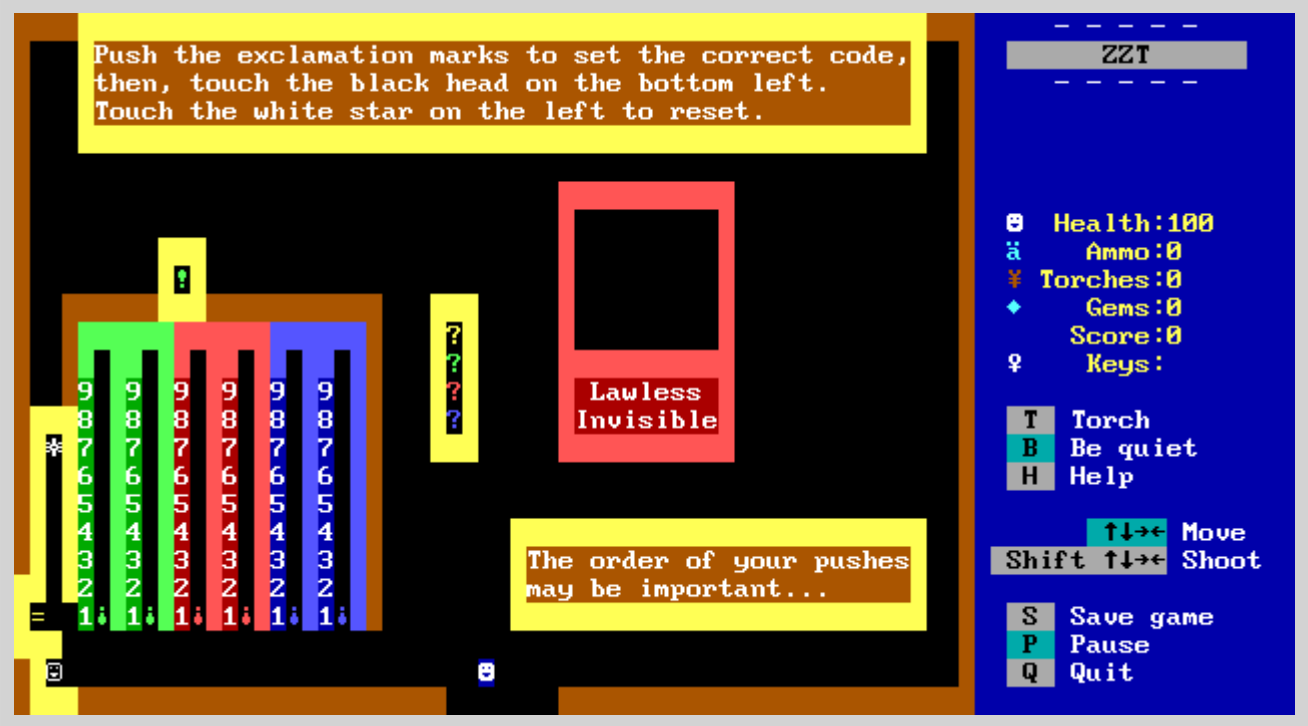
Le premier "board" (c'est le nom donné à une salle dans le jeu) permet de connaître immédiatement le format du flag, et sa première lettre : `THCon21{Z ... }`.

Vous pouvez parler à deux personnages en les touchant. Le premier suggère d'utiliser l'éditeur de ZZT. Le second raconte une histoire bizarre et totalement azimutée, sur un ton assez grave. Il s'agit d'un petit hommage à tous les jeux ZZT créés pendant les années 1990 par des adolescents. C'est le genre de propos que pourrait sortir un adolescent un peu dans son monde.

Le board à gauche contient juste un petit 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 board en bas 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, qui n'a maintenant plus trop d'utilité. L'intérêt principal de ce board est une url vers une documentation de référence décrivant les instructions de ZZT-OOP, le langage de programmation permettant de décrire le comportement des différents éléments du jeu. Ça servira pour la suite.

Le dernier board, en haut du board initial, est le plus important.



Il est possible de pousser les 6 caractères "¡" pour les placer en face des nombres 1 à 9. Nous appellerons ces objets des curseurs (c'est trop long d'écrire "point d'exclamation inversé"). Le personnage en bas à gauche, lorsqu'il est touché une première fois, répond que le code est incorrect. À la deuxième fois, il suggère de réinitialiser l'énigme. Il faut donc effectuer un reset entre chaque essai.

Touchez l'étoile blanche qui se trouve à gauche des nombres pour faire un reset. Pendant que les curseurs se replacent au bon endroit, vous êtes momentanément enfermé. C'était une contrainte obligatoire, car si vous vous placez sur leur chemin pendant qu'ils bougent, vous risquez d'empêcher leur mouvement et de les décaler.

Des tentatives supplémentaires de trouver le code ne vous apprendront pas grand chose. C'est le moment d'inspecter le board dans l'éditeur pour comprendre son fonctionnement.

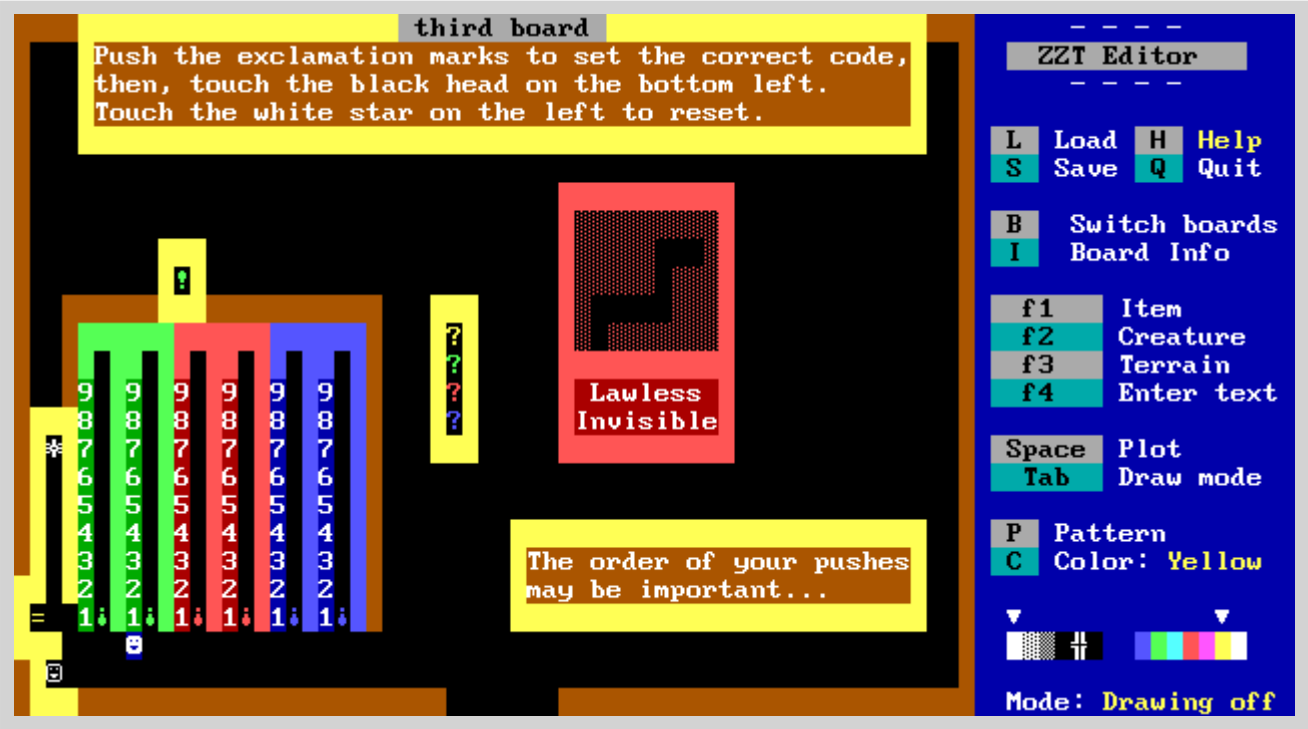
Analyse du fonctionnement global du board

Quittez la partie en cours, ouvrez l'éditeur, appuyez sur L pour loader, sélectionnez le fichier THCON21Z.ZZT, appuyez sur B pour changer de board, sélectionnez "third board".

Il y a beaucoup de fonctionnalités plus ou moins utile dans l'éditeur. Vous n'avez besoin d'en connaître que deux :

- la touche Espace permet d'effacer l'élément sur lequel est placé votre curseur.
- la touche Entrée permet d'afficher et d'éditer le code ZZT-OOP de l'objet où est votre curseur, si c'est un objet programmable.

La première chose qui saute aux yeux est la présence de murs dans le cadre rouge "Lawless Invisible". Ce sont des murs qui n'apparaissent pas pendant le jeu. Nous verrons cela plus tard.



Vous ne pouvez pas savoir à l'avance si un élément est un objet programmable ou pas. Il faut les tester en plaçant son curseur desssus et en appuyant sur Entrée. Le challenge étant relativement gentil avec vous, la plupart des objets programmables sont identifiables. Il n'y a pas un mur au milieu d'autres murs qui est un objet programmable, mais techniquement ce serait possible.

Regardez les codes des objets suivants :

- les curseurs,
- l'étoile qui fait reset,
- le personnage qui indique si le code est bon ou pas,
- les points d'interrogations,
- éventuellement le point d'exclamation vert.

Ces codes utilisent différents "flags" (des variables booléennes dans le langage ZZT-OOP) : flagFailZ, flagOkA1, flagFailB, ...

Le personnage donnant le résultat du code se nomme "GlobCheck". Le début de son code est un peu étrange, il envoie un message "lblMulti" à un mystérieux objet "Checker12". Et ensuite, sur réception d'un message "lblBack12", il effectue beaucoup de vérifications sur les flags. Une petit analyse de son code permet de voir que pour afficher le message de réussite, il faut que tous les flags "OK" (flagOkA1, flagOkA2, flagOkB1, flagOkC1) soient définis, mais que aucun des flags "Fail" ne soit défini.

Les points d'interrogation affichent l'état de différents flags, sur un label `:touch` (c'est à dire lorsqu'on les touche). Leurs noms ("dbgChalA", "dbgChalB", ...) laissent penser que ce sont des objets de debug, permettant d'analyser le comportement du board.

Mais ces points d'interrogation sont entourés par des murs et ne peuvent donc pas être touchés. Il faut enlever les murs dans l'éditeur de niveaux, puis sauvegarder le world et relancer une partie.

Dans le jeu, déplacez les curseurs, touchez les points d'interrogation et consultez les messages affichés. Vous verrez que certains flags peuvent devenir "set" (défini à True).

À cette étape de l'analyse, vous devez avoir déduit les infos suivantes :

- le challenge est décomposé en trois sous-parties, identifiés par les couleurs des éléments, les numéros des curseurs (appelées "line" dans le code) et les lettres de flags.
- la première sous-partie est constituée des éléments verts, des curseurs 1 et 2, et des flags "A" (OkA1, OkA2, FailA)
- la deuxième sous-partie est constituée des éléments rouges, des curseurs 3 et 4, et des flags "B" (OkB1, FailB)
- la troisième sous-partie est constituée des éléments bleus, des curseurs 5 et 6, et des flags "C" (OkC1, FailC)
- il y a un dernier flag : "FailZ", représentant un échec global de vos actions : vous avez refait un check sans faire de reset, ou bien vous avez poussé les curseurs trop loin.

Comme je suis quelqu'un d'assez amusant et d'un petit peu vilain, la difficulté est dans l'ordre inverse : la sous-partie la plus facile est la C, la moyenne est la B, et la plus difficile est la A.

Sous-partie C : les gemmes

Le code du premier curseur bleu contient l'instruction `#give gems 1`. À chaque fois qu'on le pousse, on voit effectivement le nombre de gemmes augmenter de 1 (c'est affiché dans l'interface du jeu dans la partie droite).

Le code du second curseur contient plusieurs blocs, dont certains commencent par les instructions `:touch`, puis `#zap touch`. Cela signifie que lorsqu'on pousse le curseur, l'exécution du code démarre au premier "touch", mais qu'il est tout de suite désactivé. Lorsqu'on pousse le curseur une deuxième fois, l'exécution du code partira donc du deuxième "touch", et ainsi de suite.

Voici le code exécuté lors du premier "touch".

```
:touch
#zap touch
?n
#set flag5Once
#take gems 4 lblPoor
' "gems 4 taken"
#end

:lblPoor
#set flagFailC
```

```
' "flag Fail C set"
#end
```

Les instructions commençant par une apostrophe sont des commentaires. Vous pouvez enlever ces apostrophes et relancer le jeu. Des textes apparaîtront à l'écran (soit en bas, soit dans une fenêtre bleue), permettant de suivre par où est passé le code.

En résumé, ce code vous reprend 4 gemmes. Mais si vous n'en avez pas assez, l'exécution se déplace au label "lblPoor", qui active un flag "Fail C".

Le code exécuté au deuxième "touch" est le suivant :

```
:touch
#zap touch
?n
#take gems 1 lblNoRich
"gems 1 taken so fail"
#set flagFailC
#end

:lblNoRich
#set flagOkC1
"flag Ok C1 set"
#end
```

C'est un peu l'inverse. Le code vous prend 1 gemmes. Si il y parvient, le flag "Fail C" est défini, sinon, le flag "OK C1" est défini. Il faut que ce code soit exécuté et que vous ayez le "OK". Donc il ne faut pas que vous possédiez de gemmes au moment de son exécution.

Le dernier code exécuté n a pas de `zap`. Il se contente d'activer "Fail C".

Il reste un dernier mécanisme à repérer : le flag "5Once" vous interdit de déplacer le premier curseur bleu après avoir déplacé le deuxième.

Il faut donc effectuer les actions suivantes :

- pousser 4 fois le premier curseur bleu
- vous possédez alors 4 gemmes
- poussez une fois le deuxième curseur bleu
- votre nombre de gemmes retombe directement à zéro
- poussez une dernière fois le deuxième curseur bleu

Pour vérifier que vous avez bien validé cette sous-partie, touchez le point d'interrogation bleu (que vous avez préalablement rendu accessible en éditant le niveau).

Le texte suivant devrait s'afficher :

```
"flag Fail C not set"
"flag OK C1 is set"
"flag 5 Once is set"
```

Le "Fail" n'est pas activé, mais le "OK" est activé. C'est bon pour cette sous-partie. **Les curseurs bleus sont sur les numéros 5 et 3.**

Sous-partie B : "Lawless Invisible"

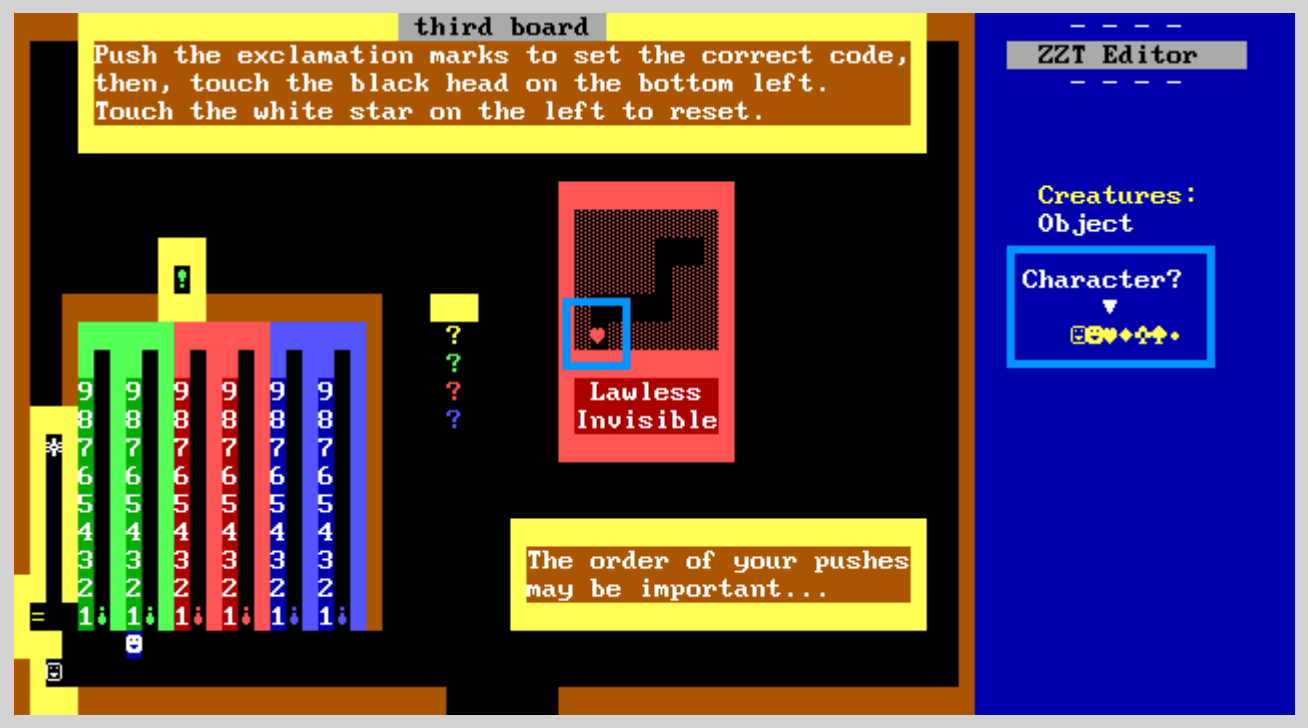
Pour la culture : le texte "Lawless Invisible" écrit dans le board est une référence à un autre jeu ZZT, intitulé Kudzu. Vous pouvez l'essayer ici : https://museumofzzt.com/file/k/kudzu.zip

Le code utile des deux curseurs rouges est assez simple : le premier envoie un message "lblGoUp" à un mystérieux objet intitulé "Invisible", le second envoie un message "lblGoEast" à ce même objet.

Il faut trouver cet objet Invisible, qui devrait, à priori, se situer dans le cadre rouge contenant déjà des murs invisibles.

On peut se douter que l'objet se trouve en bas à gauche du chemin, puisque les curseurs le font se déplacer vers le haut et vers la droite. Mais vous pouvez aussi le chercher par des essais/erreurs, en vous plaçant à différents endroits sur le chemin et en appuyant sur Entrée à chaque fois.

Lorsque vous êtes sur la bonne case et que vous appuyez sur Entrée, l'éditeur vous propose de redéfinir l'apparence de l'objet sur lequel vous êtes.



Il y a bien un objet invisible à cet endroit là ! Sélectionner n'importe quel caractère visible et re-appuyez sur Entrée. Laissez le code de l'objet tel qu'il est, sauvegardez le niveau et relancez une partie.

Le code de cet objet pourrait être étudié. Mais maintenant qu'il est visible, son fonctionnement peut être déduit à partir de simples tests.

Les deux curseurs déplacent effectivement l'objet vers le haut et vers la droite. Si on le cogne sur un mur invisible, le mouvement est annulé et le flag "Fail B" est activé.

Il faut donc amener l'objet au bout du chemin, sans le cogner. Les curseurs doivent être activés dans cet ordre :

- Premier curseur rouge, une fois.
- Deuxième curseur rouge, quatre fois.
- Premier curseur rouge, deux fois.
- Deuxième curseur rouge, deux fois.

Pour finir, touchez le point d'interrogation rouge.

Le texte suivant devrait s'afficher :

```
"flag Fail B not set"
"flag OK B1 is set"
```

Le "Fail" n'est pas activé, mais le "OK" est activé. C'est bon pour cette sous-partie. **Les curseurs rouges sont sur les numéros 4 et 7.**

Sous-partie A : Zzzzzzap !!

Lorsque vous déplacez les curseurs et que vous touchez le point d'interrogation vert, aucun des flags "A" ne s'activent. Il faut tenter une validation avec le personnage en bas à gauche (l'objet GlobCheck), et ensuite toucher le point d'interrogation vert.

Le code utile des deux curseurs verts est très court, mais un peu mystérieux :

- `#zap Checker12:lblMulti`
- `#restore Checker12:lblMulti`

D'autre part, le flag "1Once" vous interdit de déplacer le premier curseur vert après avoir déplacé le deuxième.

La documentation de ZZT-OOP donne des renseignements sur les instructions zap et restore.

Le `ZAP` permet d'annuler un label (sur l'objet courant, ou bien sur un autre objet). S'il y a plusieurs label du même nom, c'est le premier trouvé qui est annulé.

Le `RESTORE` permet de remettre un label annulé. Attention, ce n'est pas exactement l'instruction inverse de ZAP. Le label restauré est le premier label annulé trouvé, et non pas le dernier. Ça ne fonctionne pas comme des imbrications-désimbrications. C'est une particularité à retenir.

On avait vu que l'objet GlobCheck envoie un message "lblMulti" à un objet "Checker12". Cet objet est le point d'exclamation vert placé au-dessus des numéros. C'est lui qui s'occupe de définir les flags "A". Lorsque c'est fait, il renvoie un message "lblBack12" à l'objet GlobCheck, qui vérifie tous les flags.

Voici le code complet de l'objet Checker12 :

```
Checker12
#end

:lblMulti
#set flagOkA1
#zap lblMulti
#send lblMulti
' "set OK A1"
#end

:lblMulti
:lblMulti
:lblMulti
:lblMulti
:lblMulti
:lblMulti
```

```
#set flagFailA
' "pwnz 2 zzzz"
#send GlobCheck:lblBack12
#end

:lblMulti
#set flagOkA2
' "gg  !"
#send GlobCheck:lblBack12
#end

:lblMulti
:lblMulti
#set flagFailA
' "pwnz last"
#send GlobCheck:lblBack12
#end
```

Le premier bloc active le flag "OK A1", zappe un lblMulti (à priori, ce sera forcément le label de ce premier bloc), et renvoie un message lblMulti sur lui-même.

Tous les blocs après le premier se terminent par un envoi de lblBack12 à l'objet GlobCheck. Ce qui laisse penser qu'un seul bloc parmi ceux-là sera exécuté.

Le deuxième bloc contient plusieurs lblMulti et active un flag "Fail". Il faudrait que l'exécution ne passe pas par ce bloc.

Le troisième bloc n'a qu'un lblMulti, et il active un flag "OK A2". Il faut que ce soit exécuté.

Le dernier bloc active un flag "Fail". Il ne doit pas être exécuté.

Il faudrait donc pouvoir désactiver tous les lblMulti du milieu, mais laisser activé le premier et celui du troisième bloc.

Ce n'est pas faisable directement, mais ça l'est en deux étapes, grâce à l'ordre particulier de traitement des labels par les instructions "zap" et "restore".

- Poussez 7 fois le premier curseur vert. Les 7 premiers lblMulti sont désactivés.
- Poussez une seule fois le second curseur vert. Le premier lblMulti (celui qui définit "OK A1") est restauré.
- Touchez l'objet GlobCheck.

Vous pouvez maintenant toucher le point d'interrogation vert. Il affichera le texte suivant :

```
"flag Fail A not set"
"flag OK A1 is set"
"flag OK A2 is set"
"flag 1 Once is set"
```

Le "Fail" n'est pas activé, mais les "OK" sont activés. C'est bon pour cette sous-partie. **Les curseurs verts sont sur les numéros 8 et 2.**

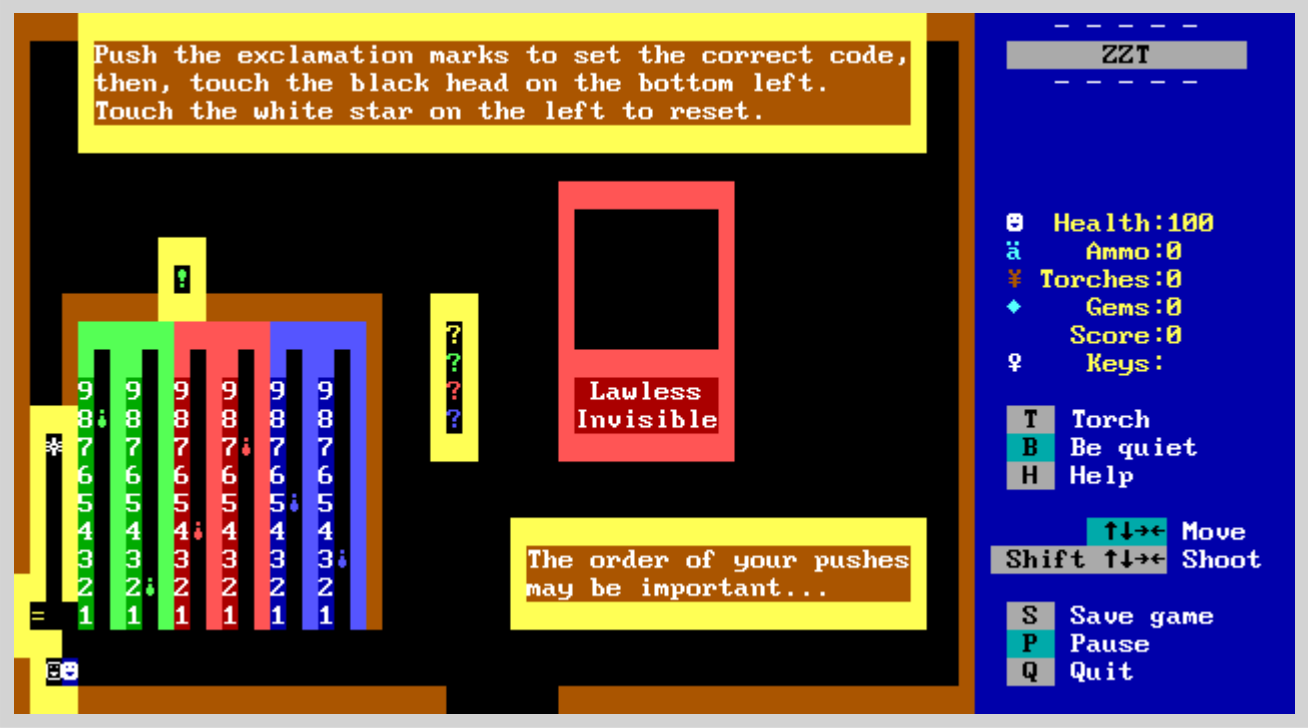
Conclusion

Il ne vous reste plus qu'à faire un reset, et repousser les curseurs où il faut. Attention à l'ordre, en particulier pour les curseurs rouges.

- Poussez 7 fois le premier curseur vert.
- Poussez une fois le second curseur vert.
- Premier curseur rouge, une fois.
- Deuxième curseur rouge, quatre fois.
- Premier curseur rouge, deux fois.
- Deuxième curseur rouge, deux fois.
- pousser 4 fois le premier curseur bleu.
- poussez deux fois le deuxième curseur bleu.
- Touchez l'objet GlobCheck.

C'est gagné, le message confirmant que vous avez trouvé le bon code s'affiche, et il y a même une musique de victoire (disons plutôt, un bruitage de victoire).

Le flag final est donc : **THCon21{Z824753}**



Et voilà, j'espère que ça vous a plu !

Réchèr.