# Daleks

# Jeu de Daleks

Les Daleks, robots envahisseurs de l’espace intersidéral, cherchent à supprimer le docteur Who. Leur seul point faible, les collisions entre eux ou avec les tas de ferrailles laissés par de précédentes collisions. Le bon docteur se défend comme il peut en créant des situations de collisions, mais ses adversaires reviennent à la charge, inexorablement, vague après vague, de plus en plus nombreux. Ce n’est qu’une question de temps avant que le docteur ne soit rejoint et qu’il ne succombe. Sa seule consolation, laisser un héritage fabuleux : les crédits cosmiques attribués par le grand chancelier galactique pour chaque Dalek détruit.

If (dalek entre en contant avec un autre dalek sa créer une ferraille )

If (dalek rentre en contact avec la ferraille d une colission de deux dalek bah il meur)

## Détails

* L’aire de jeu consiste en une surface divisée rangées et colonnes. Les cases, intersections des rangées et des colonnes, peuvent être occupées par un Dalek, un tas de ferraille ou le Docteur.
* L’aire de jeu peut-être variable, N colonnes par M rangées: commencez par de petites dimensions, mettons 8x10. (moins long pour les tests)
* Le jeu consiste en des niveaux comprenant de plus en plus de Daleks, suivant la progression suivante : 5, 10, 15, 20, etc.
* Les Daleks apparaissent à des positions aléatoires, sur la surface de jeu à chaque début de niveau.
* Le joueur contrôle les déplacements du Docteur.
* Le docteur peut se déplacer d’une seule case à la fois, dans n’importe quelle direction.
* Dès que le Docteur à jouer son tour, qui inclut la capacité à demeurer au même endroit, chaque Dalek fait un pas à son tour, vers le docteur.
* Les Daleks avancent vers le Docteur, une case à la fois, se dirigeant vers lui, tentant de l’atteindre.
* Les Daleks peuvent être détruits sous deux conditions :
* s’ils entrent en collision l’un avec l’autre, ce qui donne un tas de ferraille,
* s’ils entrent en collision avec un tas de ferraille.
* Pour chaque Dalek détruit, compter 5 points (hummm, crédits cosmiques, je veux dire…)
* Les tas de ferraille ne tuent pas le Docteur, qui ne peut aller sur une case occupée par un tas de ferraille, ni se téléporter dessus…
* Les tas de ferrailles disparaissent à la fin du niveau.

Le docteur dispose de deux outils:

un téléporteur , utilisable à volonté, qui peut lui permettre d’échapper à une mort certaine en se déplaçant vers une case aléatoirement

**en mode facile** - le téléportage se rend à une case vide, ayant au moins deux cases de distance des Daleks le plus proche afin que ceux-ci ne puissent immédiatement le tuer.

en mode ordinaire, idem mais on ne vérifie pas la proximité de Daleks

**en mode difficile** - le téléportage est complètement aléatoire et donc on peut atterrir sur un Dalek

NOTE : on ne téléporte jamais sur un tas de ferraille

**un zappeur**, acquis à chaque niveau, mais cumulable d’un niveau à l’autre, qui volatilise les Daleks dans sa portée immédiate. La partie se termine lorsque le docteur est mangé par les robots.

On conserve les noms et les points des parties, et on peut afficher cette liste triée, à la demande

### **En Options**

Run (to end), sans bouger le docteur attire tous les Daleks, jusqu’à la fin de la vague

Run -1, idem le précédent mais arrête si un Daleks est en position de tuer le docteur Architecture, objets et Interface

**NOTES**

**Les Objets (OOP)**

On exploitera au maximum les objets: à titre d'exemple il n’y aura pas de matrice de jeu, mais seulement des objets qui seront à une certaine position dans un tableau doté de propriété de longueur et de largeur.

L'application disposera d'une Vue de type Console, utilisant les fonctions print et input seulement en termes d'entrée/sortie.

En orientant l'analyse sur les objets du domaine.

L'idée est que ce sont les objets qui sont les acteurs capables d'actions à travers leurs fonctions.

Je ramène souvent l'idée de jeux de table.

Le fait de faire des classes permet de générer les objets-acteurs.

Reste à hiérarchiser la chaîne de commande lors du lancement de l'application, avec des objets qui se passent des requêtes d'actions.

# **MVC**

L'application aura une architecture MVC (Modèle-Vue-Controleur).

La **Vue** représente ce côté du logiciel chargé de présenter les données et information à l’usager, en général sur un écran mais d’autres modes sont possibles (sons, voix, imprimante, etc.). La Vue s’occupe aussi d’écouter l’usager; ses commandes, ses données.

ATTENTION la Vue ne fait de traitement autre

Le **Modèle** lui, s’occupe du traitement d’information, en vertu des exigences du domaine d’affaires propre à ce logiciel.

Le **Contrôleur** sert, comme son nom l’indique, à contrôler le programme. Le Contrôleur est LA chose dont s’occupe la fonction “main” (si elle existe ;-) )