Algorithmique

DOSSIER DE CONCEPTION DE PROJET

–

Modélisation de la propagation d'une rumeur

GARDIES Samuel
LIONG-WEE-KWONG Jade
MONFRAIX Laure
STEPHEN Valentine

TD3

TABLE DES MATIERES

1. *Objectifs du projet*

2. *Conception du projet*

3. *Structures de données et algorithmes principaux*

4. *Description et mode d'emploi de l'interface*

5. *Exemples de tests*

Préambule : *Ce document et le listing commenté du code sont complémentaires, certains détails non explicités ici le sont dans le listing du code commenté et inversement.*

1. Objectifs du projet

On souhaite simuler la propagation d'une rumeur dans une population constituée de personnages représentés par un pixel dans une matrice 2D.

Pour ce faire, un niveau de croyance à cette rumeur, représenté par une saturation d'une couleur, est attribué à chacun des personnages.

A chaque fois que la rumeur se propage, chaque personnage convaincu par celle-ci échange avec ses huit voisins et fait évoluer son niveau de croyance en fonction des paramètres suivants :

- le niveau de croyance ;
- la force de persuasion ;
- le niveau de crédulité.

On souhaite définir une loi, intégrant éventuellement une part d'aléatoire, pour que la propagation de la rumeur prenne en compte ces paramètres.

De plus, on pourra gérer la grille de personnages de manière classique ou torique, initialiser la simulation interactivement ou aléatoirement, proposer différentes vitesses de propagation de la rumeur et visualiser des informations statistiques intéressantes pour l'expérimentateur.

La finalité de ce projet est d'explorer expérimentalement, en interagissant sur différents réglages de la simulation, des comportements intéressants lors de la propagation de rumeurs.

2. Conception du projet

Tout d'abord, nous avons décidé de commencer par la création des structures fondamentales du programme.

Nous avons donc défini la structure du personnage et ses paramètres : le niveau de croyance - déterminant si le personnage est convaincu ou non par la rumeur -, la crédulité, la force de persuasion ainsi qu'une position dans la grille. Pour le choix de l'échelle, nous avons d'abord pensé à utiliser des pourcentages qui auraient parfaitement représenté le niveau de croyance ; cependant, nous avons très vite réalisé qu'une palette de cent couleurs - pour bien distinguer chacun de ces niveaux de croyance - aurait été trop compliquée à réaliser et aurait été peu visible. Ainsi, une échelle de 1 à 10 nous a semblé plus adéquate. Au vu de l'échelle choisie, les personnages seraient donc naturellement convaincus s'ils possédaient un niveau de croyance strictement supérieur à 5.

Quant à la structure de la rumeur, nous avons défini son seul paramètre, la crédibilité, de la même façon sur une échelle de 1 à 10.

Avant d'entamer l'initialisation du programme, nous avons voulu nous projeter et prévoir à l'avance certains éléments que nous voulions implanter dans le projet. Ainsi, il nous a paru intéressant de pouvoir visualiser toutes les informations caractérisant un personnage en un simple clic sur ce qui allait le représenter.

En gardant cela en tête, nous nous sommes alors lancés dans la création de l'interface du projet. Celle-ci nous est apparue comme étant divisée en trois parties : une grille contenant les cent personnages qui allaient se transmettre la rumeur, une partie "profil" permettant de visualiser les caractéristiques du personnage sélectionné et une partie permettant de paramétrer la rumeur et d'agir sur la simulation.

Les parties "paramétrage de la rumeur" et "profil du personnage" ne posaient pas de problèmes : nous avions seulement besoin de boutons et de *textbox*, voire de *picturebox*. D'un autre côté, la partie "grille de personnages" était moins évidente : nous voulions représenter chaque personnage par une *picturebox*, pour pouvoir par la suite associer à chacune de ces *picturebox* une couleur de fond en fonction du niveau de croyance du personnage lui correspondant. Cela impliquait donc de créer cent *picturebox* et de les aligner de façon à former une grille acceptable esthétiquement parlant. Heureusement, après avoir effectué quelques recherches et parcouru la boîte à outils, nous avons trouvé un objet correspondant parfaitement à nos exigences : le *TableLayoutPanel*, qui aligne automatiquement les objets présents à l'intérieur.

Ainsi, ayant déclaré nos structures et réalisé une interface basique mais complète, nous sommes passés à l'initialisation du programme.

Dans un souci de réalisme de la simulation et de simplicité de son utilisation, nous avons décidé d'attribuer, lors de l'initialisation, les paramètres des personnages de façon aléatoire, tout en limitant leurs valeurs possibles ; notamment le niveau de croyance, compris dans l'intervalle [1, 5], afin que les personnages ne soient pas convaincu avant que la rumeur ne soit lancée.

Vint ensuite l'initialisation de la position des personnages, qui allait nous poser la plus grande difficulté rencontrée durant la création de ce projet. En effet, nous avions d'un côté un tableau de cent personnages, numérotés de 0 à 99 et de l'autre une grille de cent *picturebox*, caractérisées par leurs coordonnées X et Y dans le *TableLayoutPanel*. Nous devions alors faire le lien entre ces deux éléments, en faisant correspondre le paramètre "position" des personnages à celle des *picturebox* dans la grille. Pour ce faire, nous voulions parcourir les *picturebox* grâce à une boucle de ce type :

```
For i = 1 To 100
    PictureBox&i
Next
```

Malheureusement, nous nous sommes vite rendu compte qu'il n'était pas possible de le faire ainsi. Comme nous ne trouvions pas d'autre moyen d'y parvenir, malgré le temps passé à faire des recherches, nous nous sommes finalement contentés d'une solution plus longue, mais dont le fonctionnement était garanti.

Néanmoins, nous avions certes réussi à nous passer d'une boucle parcourant les *picturebox* pour l'initialisation des paramètres des personnages ; mais la prochaine étape étant de définir la couleur des *picturebox* selon le niveau de croyances des personnages qu'elles représentent , nous avons alors réalisé que nous n'avions fait que repousser le problème et qu'il nous serait cette fois impossible de le contourner.

S'ensuivirent de nombreuses recherches et de nombreux tests dans le but de réaliser cette boucle. Malgré le temps que ces recherches commençaient à prendre, nous avons persévéré car nous étions conscients de l'importance de celle-ci. En effet, une fois cette dernière obtenue, nous allions pouvoir la réutiliser tout au long de la programmation du reste du projet qui allait s'en trouver fortement simplifiée.

Finalement, c'est en s'intéressant à la page décrivant le *TableLayoutPanel* que nous avons trouvé une commande permettant de réaliser la boucle que nous désirions. En effet, il est possible d'accéder à un élément d'un *TableLayoutPanel* à partir d'une position à l'aide de la commande suivante :

[Nom du TableLayoutPanel].GetControlFromPosition([position X], [Position Y])

Il ne nous restait alors plus qu'à parcourir les personnages et à insérer leur position dans ladite commande, ce que nous avons fait ainsi :

```
For i = 0 To 99
    TableLayoutPanel1.GetControlFromPosition(LePersonnage(i).pos.x, LePersonnage(i).pos.y)
Next
```

Étant enfin en mesure de parcourir les *picturebox*, nous avons alors sélectionné dix nuances de bleu bien distinguables les unes des autres que nous avons classées de la plus claire à la plus foncée (Cf. *image* 1), et associées aux dix niveaux de croyance, terminant ainsi l'initialisation du programme.

Niveau de croyance	1	2	3	4	5	6	7	8	9	10
Couleur de la *picturebox*										

Image 1 : couleurs des picturebox *selon le niveau de croyance du personnage qu'elles représentent*

S'ensuivit la création du bouton permettant de transmettre la rumeur à un personnage choisi via sa position, qui fut réalisée assez facilement grâce à la boucle évoquée plus haut. Pour déterminer le niveau de croyance du personnage après avoir reçu la rumeur, nous avons décidé de prendre en compte son niveau de croyance initial, sa crédulité et la crédibilité de la rumeur, l'idée étant alors d'additionner ces trois paramètres, pour obtenir un résultat prenant ses valeurs dans l'intervalle [3, 25]. Nous voulions ensuite diviser cet intervalle en cinq parties correspondant chacune à un niveau de croyance du personnage, niveau qui serait alors compris entre 6 et 10 inclus, pour que le personnage soit bien convaincu. Ceci nous amena à réduire les valeurs possibles pour la crédibilité de la rumeur à sept, afin que la division de l'intervalle en cinq parties ne pose pas problème.

Néanmoins, ces décisions ne s'avérèrent que temporaires puisque nous allions être amenés à revenir dessus lors de la création de la loi régissant la propagation de la rumeur.

Il ne nous restait alors plus qu'à programmer la propagation de la rumeur pour avoir la base du projet. Toutefois, avant de nous y mettre, nous avons décidé de finaliser la visualisation des informations des personnages que nous avions en tête depuis le début. Ainsi, nous avons programmé de façon similaire les cent *picturebox* - en changeant seulement leur numéro - pour que, lorsque l'on clique sur l'une d'entre elles, toutes les caractéristiques du personnage qu'elle représente soient affichées dans la zone de l'interface dédiée.

Finalement, nous sommes passés à la propagation de la rumeur. Dans un premier temps, nous voulions réussir à sélectionner tous les personnages qui allaient recevoir la rumeur, à partir de la position de ceux déjà convaincus au moment de la propagation. Cela ne fut pas si simple puisque nous avions décidé de gérer la grille de personnages de manière torique, comme suggéré dans le sujet. Ceci impliquait de distinguer deux cas : celui des personnages ne se trouvant pas sur les bords de la grille (Cf. *image* 2), qui allaient simplement transmettre la rumeur aux huit personnages les entourant et celui des personnages se trouvant sur les bords de la grille (Cf. *image* 3), qui allaient faire de même, mais en propageant en plus la rumeur à leur "voisins" se trouvant de l'autre côté de la grille.

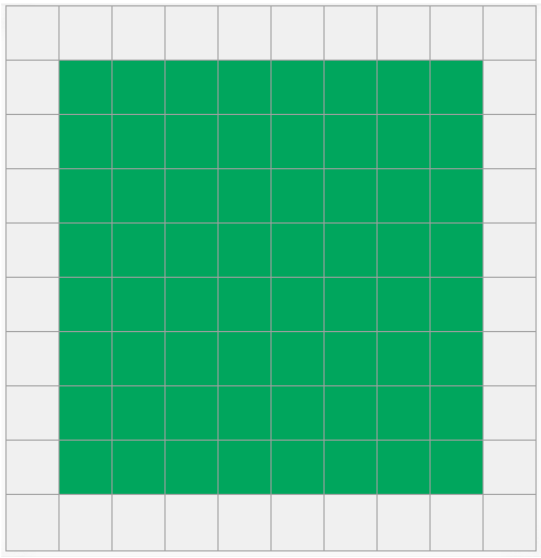


Image 2 : personnages au centre de la grille
(extrait du listing commenté du code)

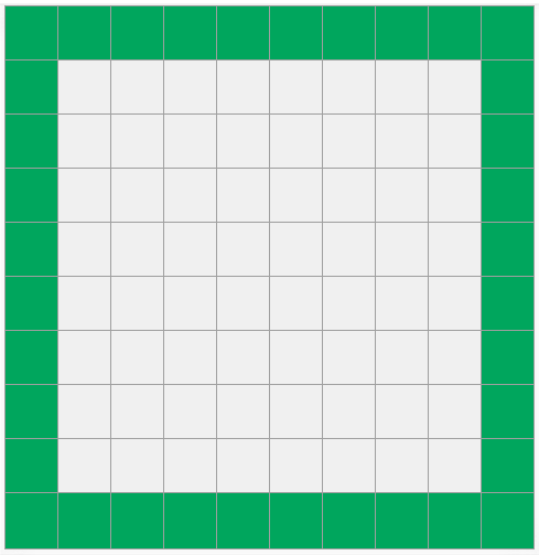


Image 3 : personnages sur les bords de la grille
(extrait du listing commenté du code)

Étant habitués, grâce au projet du semestre 3 et au premier TP du semestre 4, à sélectionner les huit cases entourant une case précise à partir de la position de cette dernière, le cas des personnages au centre de la grille ne posait pas de problème. En revanche, celui des personnages sur les bords nous a, quant à lui, pris plus de temps et a nécessité d'être à son tour divisé en quatre cas - un pour chaque côté de la grille -, mais nous avons fini par y arriver.

Une fois les personnages, qui allaient recevoir la rumeur, sélectionnés, nous sommes passés à la deuxième étape : la création d'une règle pour définir les nouveaux niveaux de croyance après propagation de la rumeur.

Nous avons quelque peu hésité sur la nature de l'échange entre le personnage transmettant la rumeur et celui la recevant : était-ce un débat auquel les deux participaient ou bien le personnage recevant la rumeur se contentait-il d'écouter puis de juger s'il était convaincu ?

Le choix le plus réaliste nous a semblé être le premier. Néanmoins, nous avons décidé que, bien que les deux personnages débattaient et pouvaient donc se convaincre mutuellement, seul le niveau de croyance du personnage recevant la rumeur allait être amené à changer.

Nous avons ensuite listé les variables qui allaient influencer le nouveau niveau de croyance du personnage recevant la rumeur, que ce soit positivement ou négativement et qu'elles appartiennent à l'un ou à l'autre des personnages. Les variables que nous avons retenues ainsi que leur influence - positive ou négative - et leurs intervalles de valeurs sont récapitulées dans le tableau suivant :

	Niveau de croyance	**Force de persuasion**	**Crédulité**
Personnage propageant la rumeur	- Impact : positif - Échelle : 1-10	- Impact : positif - Échelle : 1-10	- Impact : négatif - Échelle : 1-10
Personnage recevant la rumeur	- Impact : positif - Échelle : 1-5	- Impact : négatif - Échelle : 1-10	- Impact : positif - Échelle : 1-10

	Crédibilité
Rumeur	- Impact : positif - Échelle : 1-7

Influence des différents paramètres sur le niveau de croyance du personnage recevant la rumeur

***Remarque**: le personnage recevant la rumeur n'étant pas convaincu, l'échelle de son niveau de croyance est donc 1-5 et non pas 1-10.*

Nous voulions alors procéder de la même façon que lors de la transmission de la rumeur à un personnage choisi, c'est-à-dire additionner l'ensemble de ces paramètres, puis diviser les valeurs possibles pour ce résultat en dix intervalles correspondants au nouveau niveau de croyance du personnage recevant la rumeur.

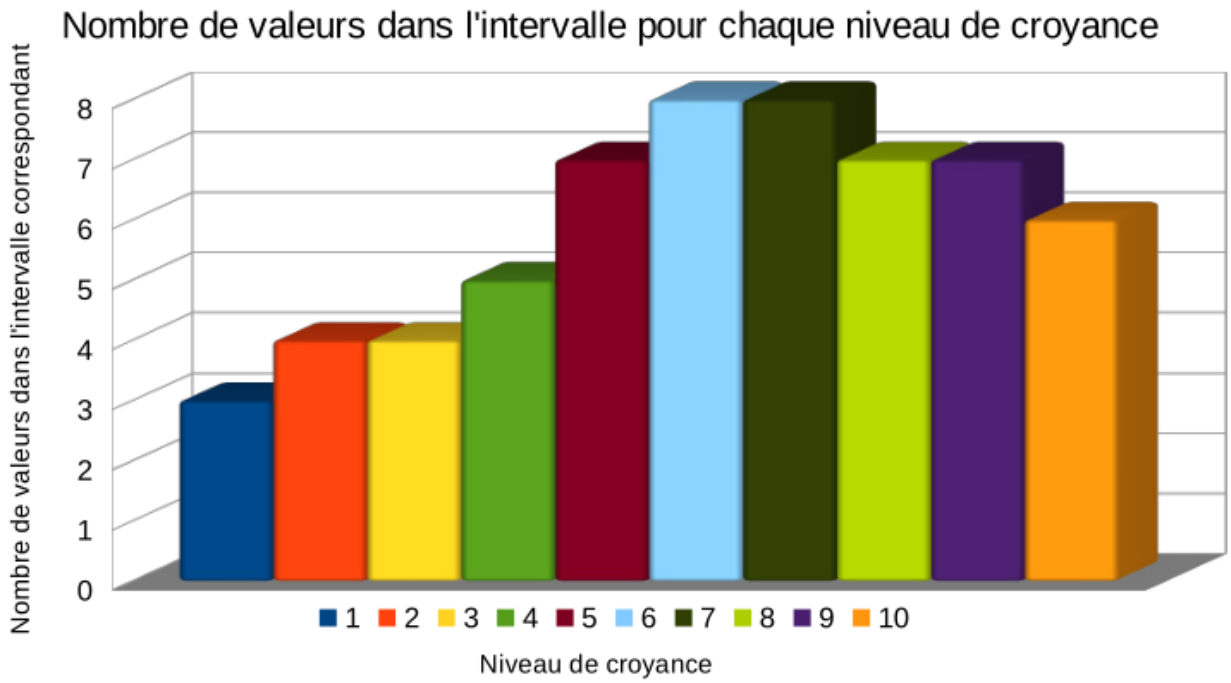
Plusieurs problèmes se sont alors posés à nous : l'impact de la crédibilité de la rumeur était trop faible proportionnellement au reste des paramètres, la division en dix intervalles ne pouvait se faire en l'état et une modification éventuelle de l'échelle de la crédibilité de la rumeur pour régler ce problème se serait répercutée sur la transmission de la rumeur à un personnage.

Nous avons donc modifié certains paramètres pour harmoniser les deux programmes tout en restant le plus réaliste possible et en faisant en sorte que les divisions en intervalles soient réalisables. Ainsi, nous avons rétabli l'échelle de la crédibilité de la rumeur à 1-10, supprimé l'impact du niveau de croyance du personnage avant qu'il ne reçoive la rumeur dans la transmission de la rumeur à un personnage précis et supprimé l'impact de la crédibilité de la rumeur dans sa propagation.

Tous les problèmes semblaient alors réglés. Toutefois, lorsque nous avons testé la propagation de la rumeur, nous nous sommes vite rendu compte que quelque chose n'allait pas. En effet, la rumeur se propageait comme il faut. Néanmoins, elle ne durait que très peu de temps : même avec un niveau de crédibilité élevée, elle arrêtait rapidement de se propager. La source du problème était claire : la propagation que nous avions mise en place prenait en compte trois paramètres positifs allant de 1 à 10 et trois paramètres négatifs allant, pour deux d'entre eux, de 1-10 mais pour le troisième, de 1 à 5. Ceci, couplé au fait que les intervalles correspondant au nouveau niveau de croyance contenait chacun le même nombre de valeurs, entraînait une tendance de la rumeur à disparaître rapidement.

Pour résoudre ce problème, nous devions donc modifier ces intervalles, de façon à ce qu'ils ne soient plus équilibrés, mais que ceux correspondant à des niveaux de croyances élevés contiennent plus de valeurs que les autres.

Ainsi, après avoir testé la propagation de rumeur avec différents paramétrages de ces intervalles, nous avons retenu la répartition suivante :



La propagation de la rumeur programmée, nous avions terminé la base du projet. Nous nous sommes donc consacrés à l'ajout d'extensions et à la finalisation de l'interface.

En effet, l'interface que nous avions à ce moment-là était fonctionnelle mais peu plaisante visuellement puisqu'il ne s'agissait que d'une grille entourée de *textbox* et de boutons. Nous avons donc placé une *picturebox* à l'arrière-plan de l'interface qui contiendrait notre image de fond (Cf. *image* 4) ; puis nous avons ajusté la disposition des

composants de l'interface pour correspondre à ladite image. Enfin, nous avons modifié l'aspect des boutons et des *textbox* en jouant sur la taille, la mise en forme des écritures, ou bien encore la couleur et l'image de fond.

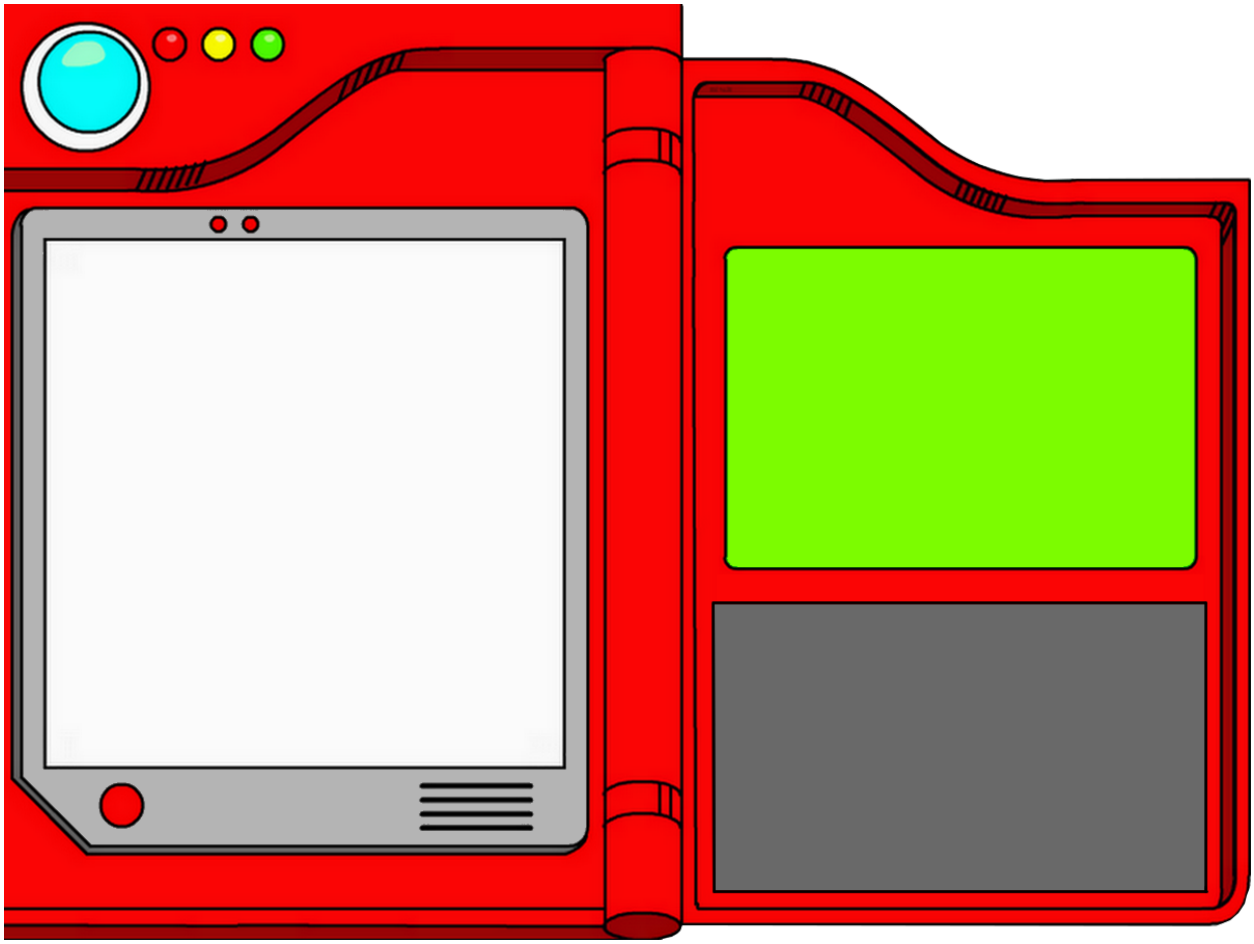


Image 4 : image de fond de l'interface

Pour compléter cette nouvelle interface plus esthétique que la précédente, nous avons rajouté quelques détails comme le changement de curseur au survol des *picturebox* en fonction de si le personnage correspondant est convaincu ou non (Cf. *image* 5), le changement de bordure lors du clic sur une des *picturebox* pour la démarquer des autres (Cf. *image* 6), ou encore l'ajout d'une *picturebox* dans la zone de profil du personnage, reprenant la couleur de la *picturebox* correspondant au personnage sélectionné pour la rendre plus visible.

Image 5 : apparence du curseur au survol d'un personnage convaincu ou non par la rumeur

Image 6 : apparence d'une picturebox avant et après avoir cliqué dessus

Tout ceci nous permet d'avoir un aperçu plus rapide, pratique et ergonomique de l'état d'un personnage, évitant ainsi de devoir cliquer sur une case à chaque fois que l'on veut savoir si un personnage est convaincu ou non.

Une fois l'interface terminée, nous avons implémenté une *trackbar* pour contrôler la vitesse de la propagation de la rumeur, ainsi qu'un bouton pour arrêter cette propagation.

Pour mettre en place différentes vitesses de propagation, nous voulions modifier le bouton de propagation de la rumeur pour qu'il répète la fonction propageant la rumeur toutes les *x* secondes.

Avant de commencer à chercher un moyen de faire un programme qui attendrait avant de répéter une boucle, un test du bouton nous a révélé un problème : en répétant plusieurs fois la fonction propageant la rumeur, cette dernière ne se propageait en fait qu'une fois. Comme nous pensions que ce problème était dû au fait que la fonction propageant la rumeur partait, à chaque fois qu'elle était répétée, de la même position de départ, nous avons alors pensé à répéter un clic sur le bouton propageant la rumeur au lieu de répéter directement la fonction propageant la rumeur. Heureusement, nous avons rapidement trouvé qu'il était possible de faire une telle chose grâce à la commande suivante

```
Select Case TrackBar1.Value
    Case 0
        ButtonRépandre_Click(ButtonVRAI, e)
    Case 1
```

Nous avons donc créé un nouveau bouton qui cliquerait sur le bouton utilisé précédemment pour répandre la rumeur, que nous avons ensuite rendu invisible sur l'interface pour éviter toute confusion.

Il ne nous restait plus qu'à trouver un moyen de faire en sorte que le programme attende avant de passer au clic suivant. Après de nombreux tests, ainsi qu'une longue et vaine tentative de comprendre et d'utiliser un timer, nous avons fini par découvrir une solution beaucoup plus simple et efficace : la commande "*Await Task.Delay()*" qui permet de faire exactement ce que l'on souhaitait en choisissant le temps d'attente, en milliseconde, avant de répéter la boucle. Nous avons donc mis en place trois vitesses différentes : "pas à pas", "lente" - propagation toutes les deux secondes - et "rapide" - propagation toutes les secondes.

Une dernière chose manquait alors au projet : nous voulions proposer un graphique permettant de visualiser l'évolution de la propagation de la rumeur au cours du temps.

N'ayant jamais utilisé cette fonctionnalité de *Visual Basic*, nous ne savions pas vraiment ce qu'il allait être possible ou non de faire. Heureusement, nous avons pu

rapidement apprendre tout ce que nous avions besoin de savoir, notamment grâce à cette **vidéo**.

Nous avons donc rajouté une partie initialisation du graphique pour le bouton initialiser, ainsi qu'une partie actualisation du graphique pour chaque bouton modifiant la propagation de la rumeur. Ainsi, le graphique ne s'affiche pas qu'à la fin pour récapituler l'évolution de la rumeur, mais s'actualise automatiquement tout au long de la simulation à chaque fois celle-ci se répand.

Concernant l'interface, comme ajouter une quatrième zone pour le graphique l'aurait au mieux surchargée, au pire était impossible, nous avons décidé de superposer le graphique et la grille des personnages, et de créer un bouton - deux en réalité, voir le listing commenté du code - permettant de passer de l'affichage de l'un à l'autre, en envoyant simplement l'un ou l'autre à l'arrière-plan.

Pour terminer, nous avons mis en place des *tooltip* (Cf. *image* 7) s'affichant au survol de certaines parties de l'interface, pour détailler leur fonctionnement sans la rendre illisible.

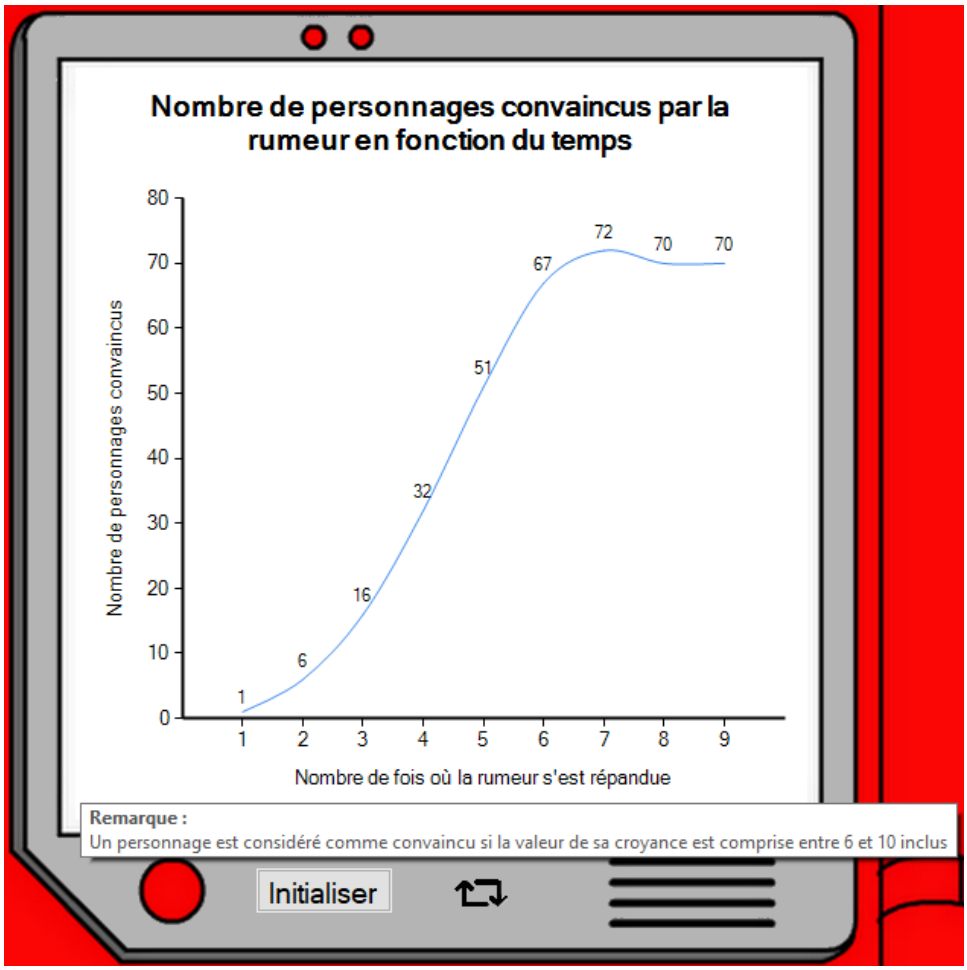


Image 7 : exemple d'un tooltip *rappelant dans quel cas un personnage est considéré convaincu*

3. Structures de données et algorithmes principaux

Liste des structures utilisées dans le projet :

- Coordonnées (*Structure coord*): définies par une coordonnée X en abscisse et Y en ordonnée ;
- Personnage (*Structure personnage*): défini par une position qui renvoie à la *structure coord*, son niveau de croyance, sa force de persuasion et sa crédulité, mais aussi un *booléen* pour savoir si oui ou non il est convaincu ;
- Rumeur (*Structure rumeur*): définie par sa crédibilité.

Liste des procédures utilisées dans le projet :

- Initialiser : utilisée dans l'initialisation du tableau de personnage ;
- Répandre la rumeur : permet de répandre la rumeur parmi tous les personnages.

Liste des Objets utilisées dans le projet :

- Boutons :
    - Initialiser: initialise l'ensemble de la simulation ;
    - Lancer la rumeur: transmet la rumeur à un personnage voulu ;
    - Répandre la rumeur: propage une fois la rumeur (bouton invisible sur l'interface) ;
    - Répandre la rumeur Vrai: utilise le bouton précédent pour lancer la propagation de la rumeur ;
    - Stop: suspend la propagation de la rumeur ;
    - Graphique: affiche le graphique à la place de la grille de personnages ;
    - Grille: affiche la grille de personnages à la place du graphique.

- *Textbox :*
    - Profil du personnage, Gestion de la rumeur, Vitesse, Répandre la rumeur : servent de titres aux zones/boutons de l'interface correspondant ;
    - Position X : affiche la position en abscisse du personnage sélectionné ;
    - Position Y : affiche la position en ordonné du personnage sélectionné ;
    - Niveau de croyance: affiche le niveau de croyance du personnage sélectionné ;

    - Force de persuasion: affiche la force de persuasion du personnage sélectionné ;
    - Niveau de crédulité: affiche la crédulité du personnage sélectionné ;
    - Convaincu ?: explicite si le personnage sélectionné est convaincu ou non par la rumeur ;
    - Crédibilité: permet de choisir la crédibilité de la rumeur ;
    - Position X: permet de choisir la position en abscisse du personnage qui va recevoir la rumeur ;
    - Position Y: permet de choisir la position en ordonnée du personnage qui va recevoir la rumeur.

Remarque **:** *les textbox précédentes sont toutes accompagnées de textbox leur servant de titre sur l'interface.*

- Divers :
    - Picturebox: cent *picturebox* représentant les personnages, une *picturebox* servant à afficher l'image de fond de l'interface et une *picturebox* reprenant la couleur du personnage sélectionné ;
    - Trackbar: permet de choisir la vitesse de propagation de la rumeur ;
    - Chart: contient le graphique représentant le nombre de personnages convaincu par la rumeur en fonction du nombre de propagation de cette dernière.

4. Description et mode d'emploi de l'interface

Le design de l'interface a été conçu sur *Photoshop*. Nous avons repris une image d'un pokédex - tirée du célèbre dessin animé *Pokémon* - que nous avons modifié pour nos besoins (Cf. *image* 8) de façon à délimiter trois zones présentant chacune une partie de l'interface (Cf. *mode d'emploi de l'interface* - p.15).

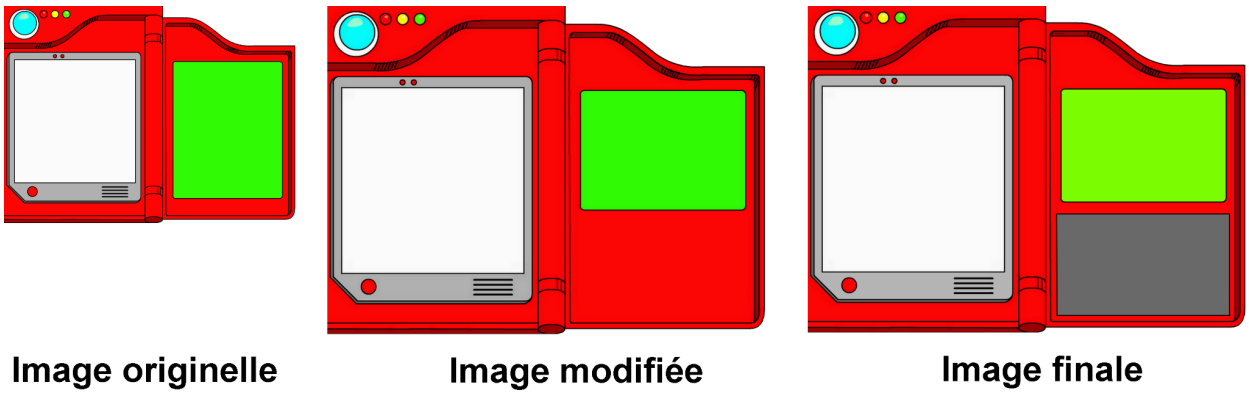
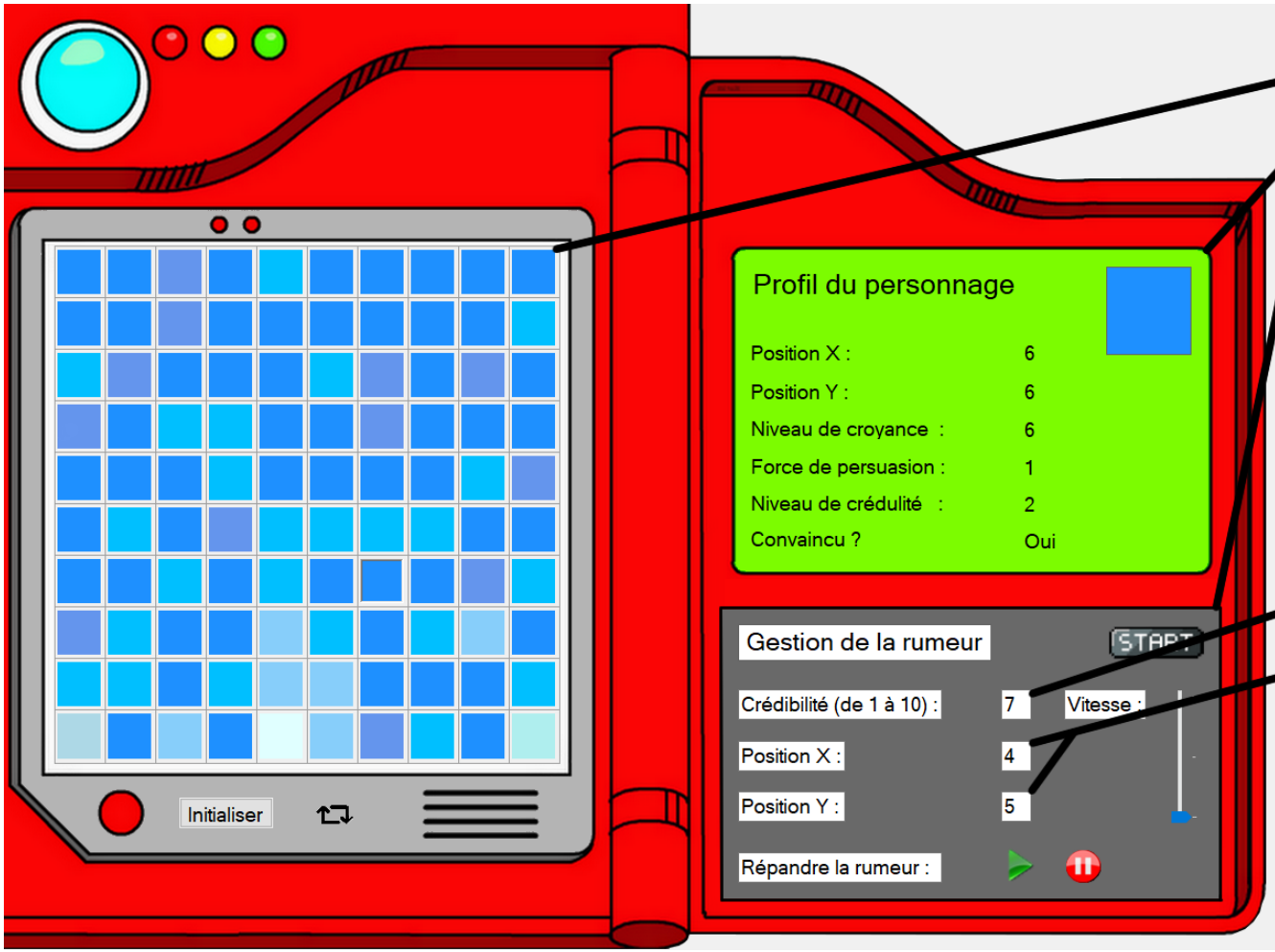


Image 8 : différentes étapes de la création de l'image de fond de l'interface

Concernant les objets présents sur l'interface, nous avons modifié leurs tailles et leurs couleurs de fond pour bien les intégrer à l'image de fond. De plus, toujours sur *Photoshop*, nous avons modifié l'apparence de la majorité des boutons pour éviter d'avoir de simple carré avec un texte dessus (Cf. *image* 9).

Image 9 : images utilisées pour changer l'apparence des boutons



Description et mode d'emploi de l'interface :

Zone d'affichage N°1 :
Affiche les 100 personnages ou le graphique

Zone d'affichage N°2 :
Affiche toutes les informations du personnage voulu

Panneau de controle :
Contient les boutons et les boîtes de dialogue permettant de paramétrer la rumeur

Initialiser — **Initialise (ou réinitialise) la simulation**
Remarque : toujours appuyer sur ce bouton avant de faire quoi que ce soit

Change ce qui est affiché dans la zone N°1 (graphique ou personnages)

START — **Transmet la rumeur au personnage choisi**

Permet de choisir la valeur de la crédibilité de la rumeur

Permet de choisir la position du personnage qui va recevoir la rumeur

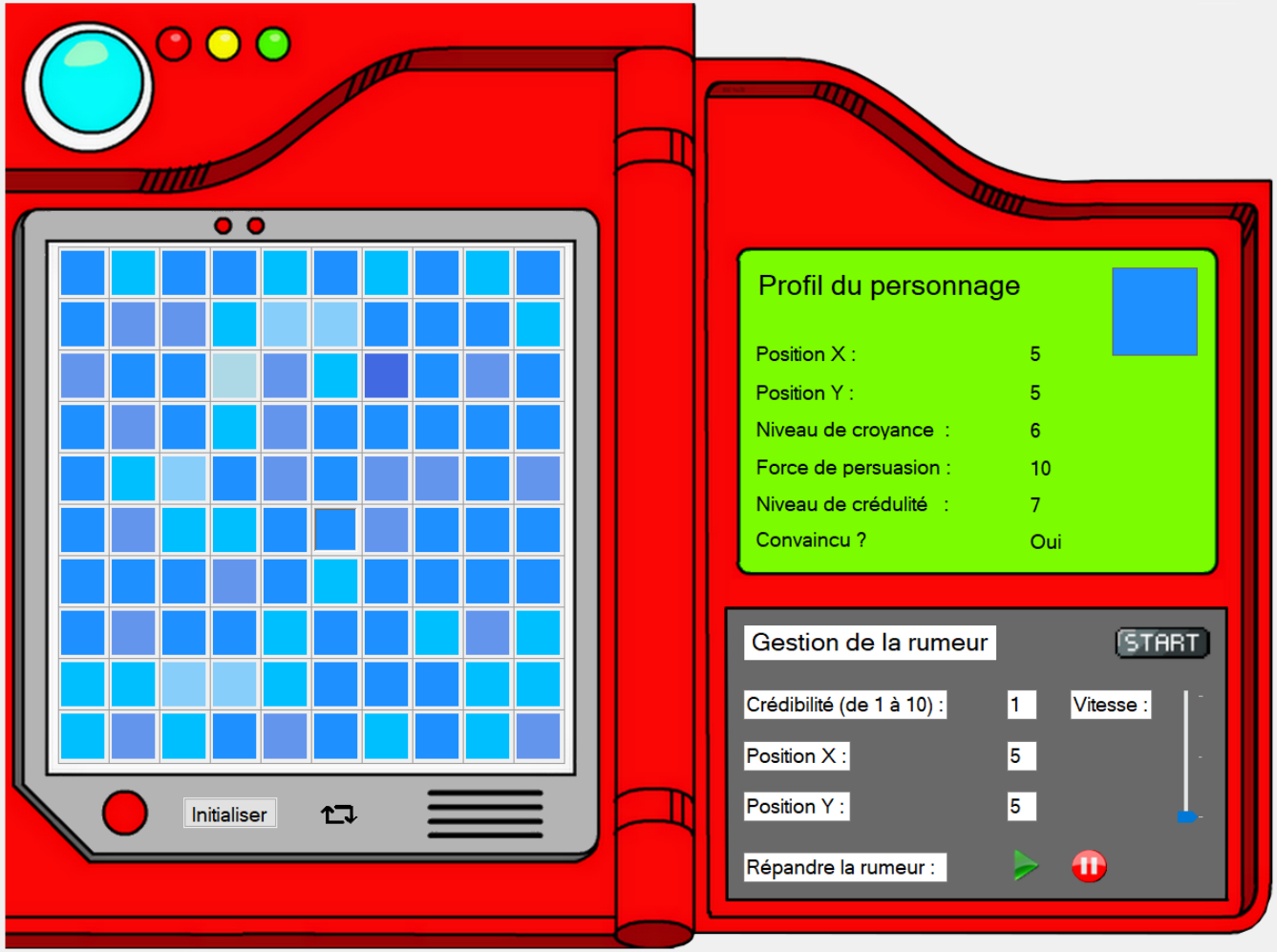
Permet de choisir la vitesse à laquelle la rumeur va se propager

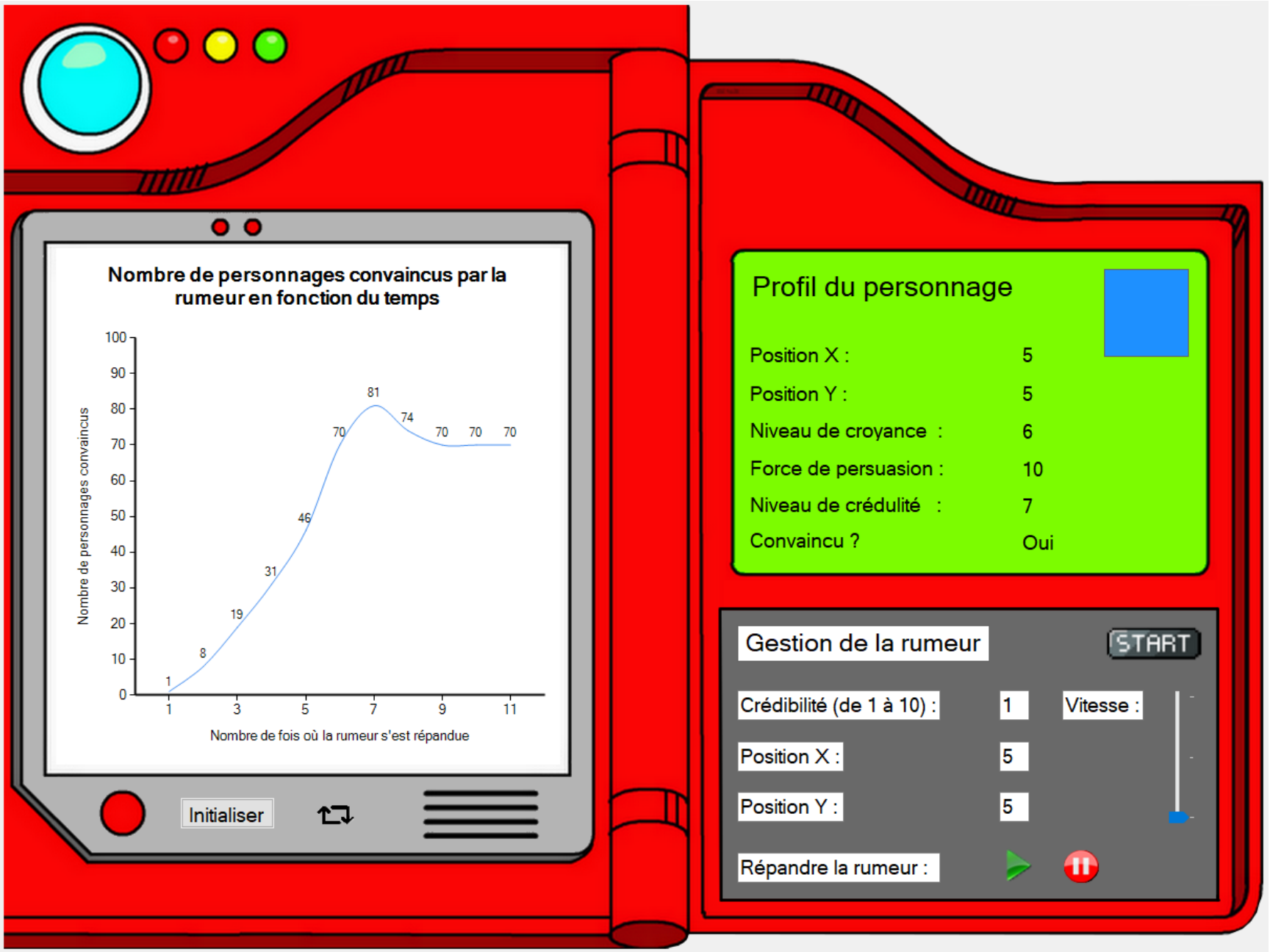
Lance la propagation de la rumeur
Remarque : la rumeur se propage à partir de chaque personnage convaincu et de façon torique

Suspend la propagation de la rumeur

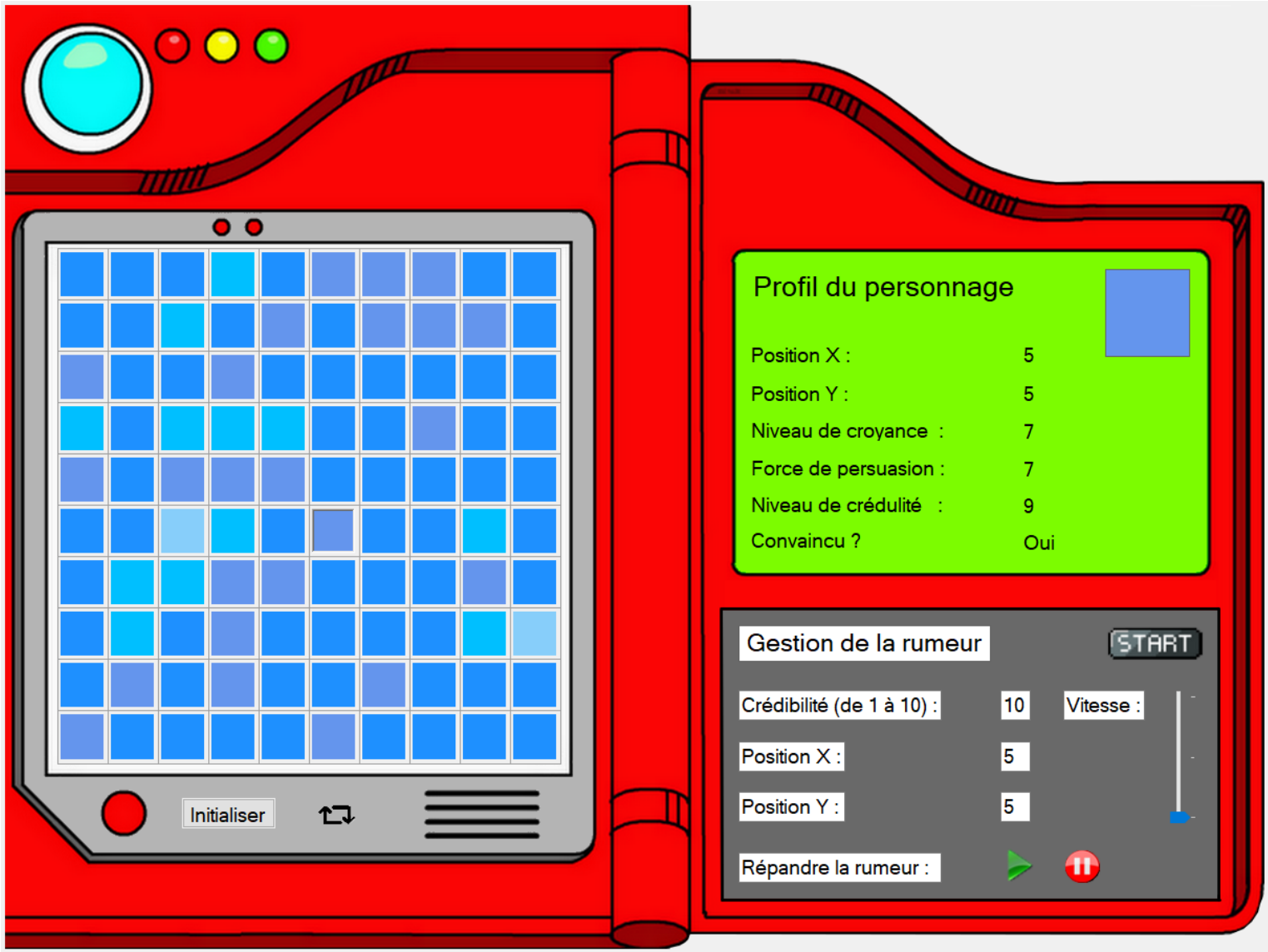
5. Exemples de tests

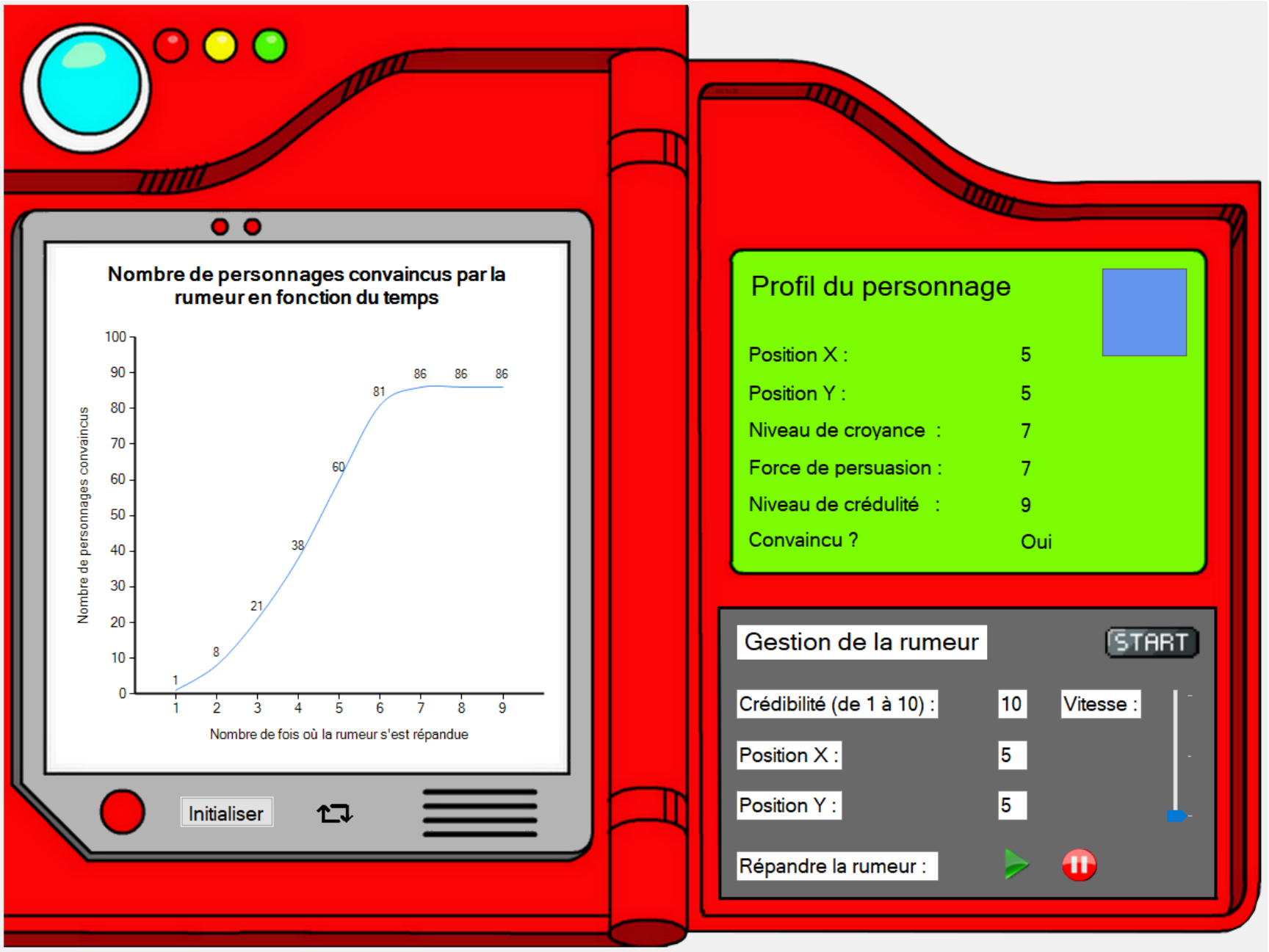
Test N°1 : simulation de la propagation d'une rumeur ayant un niveau de crédibilité de 1




Nombre de personnages convaincus par la rumeur en fonction du temps
Nombre de personnages convaincus
Nombre de fois où la rumeur s'est répandue
1
8
19
31
46
70
81
74
70
70
70
Initialiser
Profil du personnage
Position X : 5
Position Y : 5
Niveau de croyance : 6
Force de persuasion : 10
Niveau de crédulité : 7
Convaincu ? Oui
Gestion de la rumeur
START
Crédibilité (de 1 à 10) : 1
Vitesse :
Position X : 5
Position Y : 5
Répandre la rumeur :

Test N°2 : simulation de la propagation d'une rumeur ayant un niveau de crédibilité de 10




Nombre de personnages convaincus par la rumeur en fonction du temps
Nombre de personnages convaincus
Nombre de fois où la rumeur s'est répandue
1
8
21
38
60
81
86
86
86
Initialiser
Profil du personnage
Position X : 5
Position Y : 5
Niveau de croyance : 7
Force de persuasion : 7
Niveau de crédulité : 9
Convaincu ? Oui
Gestion de la rumeur
START
Crédibilité (de 1 à 10) : 10
Vitesse :
Position X : 5
Position Y : 5
Répandre la rumeur :

La comparaison des tests N°1 et 2 permet de bien se rendre compte que l'impact du niveau de crédibilité de la rumeur dans la simulation n'est pas négligeable, mais est loin d'être le seul paramètre pris en compte. En effet, bien que l'on observe une plus faible propagation de la rumeur dans le cas d'une crédibilité de 1 que dans le cas d'une crédibilité de 10, la différence entre les deux tests est modérée. Les caractéristiques des personnages avant l'apparition de la rumeur jouent donc un rôle important dans la propagation de celle-ci.

Cet aspect du projet nous a paru très intéressant, car il illustre bien l'influence décisive que peut avoir le choix de la population lors d'une simulation.

Test N°3 : simulation de la propagation d'une rumeur en vitesse "lente"

Test N°4 : simulation de la propagation d'une rumeur en vitesse "rapide"

Les tests N°3 et 4 illustrent les différentes vitesses possibles pour la propagation de la rumeur.