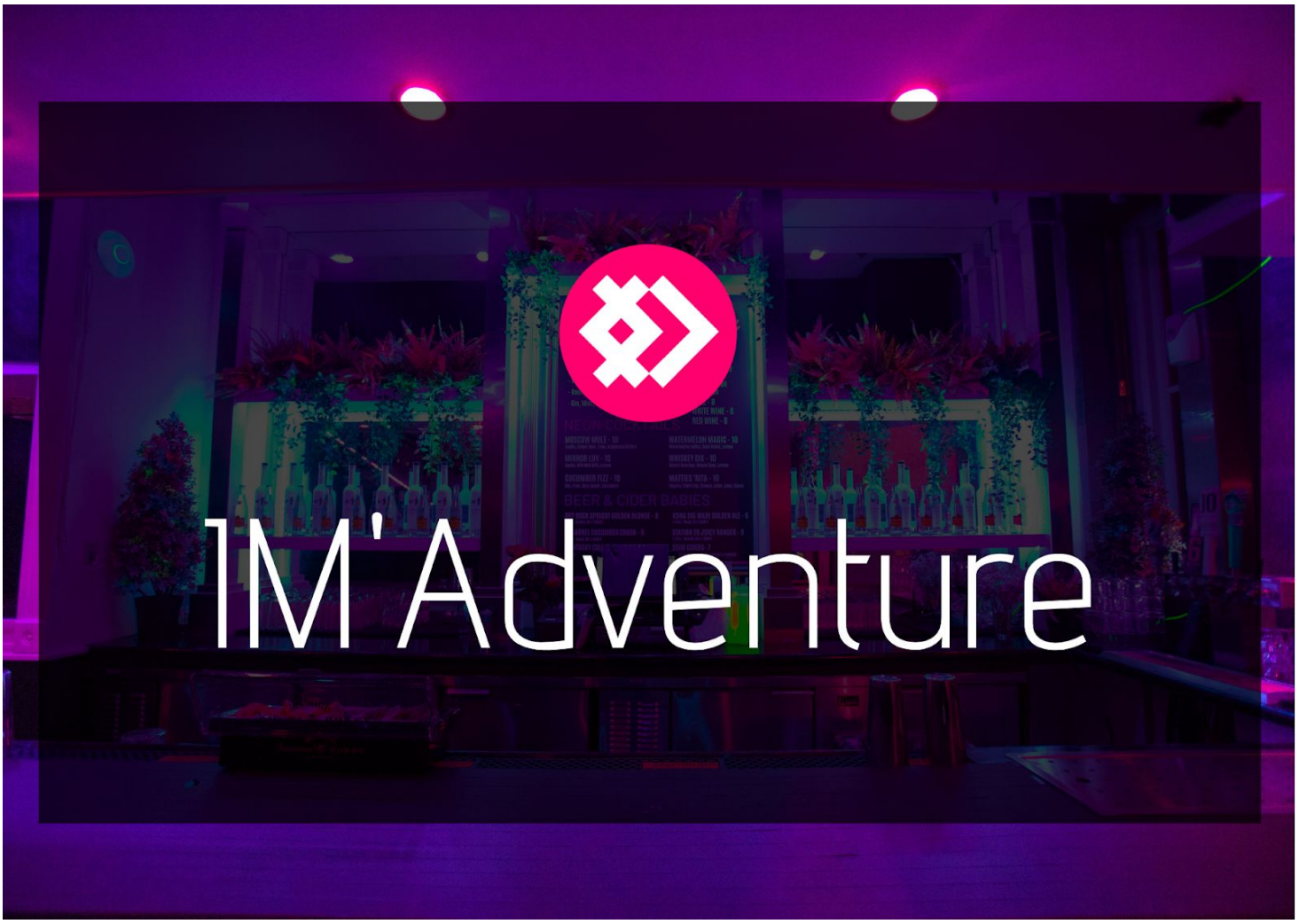
IM'ADVENTURE

Projet de Maths - Probabilités



Laurelenn Sangaré - Margaux Vaillant
IMAC2 - Promotion 2021

Université Gustave Eiffel Esipe INGÉNIEURS PARIS EST

Sommaire

I/ Mise en contexte

Synopsis :

"C'est le début de l'année ! Vous voilà enfin à l'IMAC ! L'école de vos rêves ! Vous allez rencontrer des personnes avec des profils très hétérogènes, et découvrir des matières très différentes, et tout le monde a l'air très sympathique. Ces trois années vont être très enrichissantes ! Mais vous ne perdez pas de vue votre objectif principal : **Avoir une assez bonne moyenne pour partir à l'étranger en troisième année** !
Allez, faites de votre mieux ! "

Règles du jeu :

Sélectionnez le **type d'IMAC** avec lequel vous voulez jouer, et essayer d'obtenir la **meilleure note** possible à chacune des épreuves ! Faites attention aussi à votre pourcentage de **"love"**, c'est important la bienveillance à l'IMAC. Lisez bien la consigne en haut de la zone de jeu, et que la chance soit avec vous !

Le Scientifique

- Sciences : ●●●●○
- Programmation : ●●●○○
- Arts : ●○○○○
- Survie : ●○○○○

L'Artiste

- Sciences : ●○○○○
- Programmation : ●●○○○
- Arts : ●●●●●
- Survie : ●●○○○

Le Polyvalent

- Sciences : ●●○○○
- Programmation : ●●●○○
- Arts : ●●○○○
- Survie : ●●●○○

Le Communiquant

- Sciences : ●○○○○
- Programmation : ●○○○○
- Arts : ●●●○○
- Survie : ●●●●●

II/ Lois et variables aléatoires

1) Les variables aléatoires

Les paramètres que l'utilisateur peut choisir de lui-même et qui sont capables d'influer sur l'ensemble des lois de probabilité sont les suivantes :

- Le **skill en Science**
- Le **skill en Programmation**
- Le **skill en Art**
- Le **skill en Communication**

Ces quatre skill sont définis en fonction du **type d'IMAC** que choisit le joueur.

Le joueur peut également influer sur trois autres variables aléatoires :

- Au Semestre 2 (partiel de signal) : il choisit **sa méthode de révision**
- Au Semestre 3 (love et sel) : il peut **augmenter et diminuer son score en "love"** en fonction de sa performance.
- Au S4 (Gala - bonus) : il peut **choisir combien il sert de verres** à son interlocuteur pour avoir une meilleure note !

Ainsi, le joueur influe sur six à sept paramètres de variables aléatoires sur l'ensemble du jeu.

Les **variables aléatoires** présentes sur le jeu sont les suivantes :

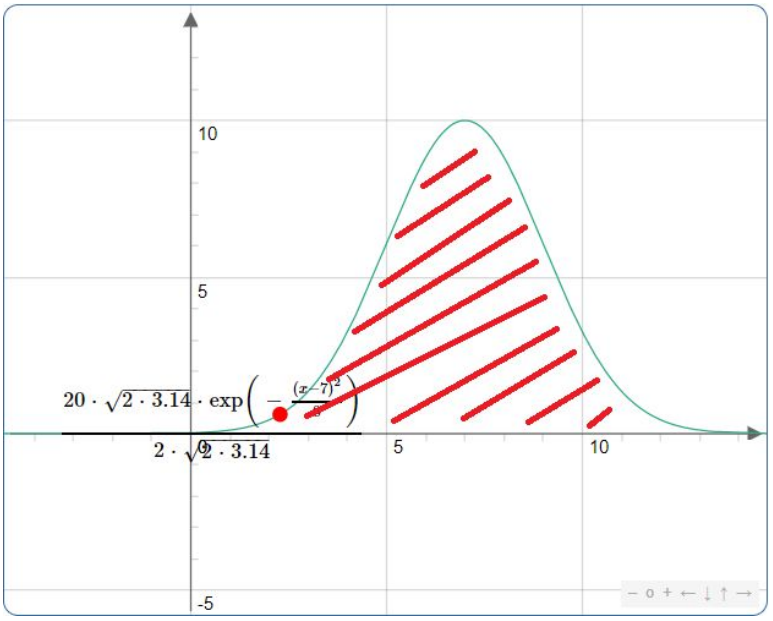
a) Semestre 1 : Matière sélectionnée au Semestre 1

- Modélise la sélection du partiel que va passer le joueur.
- La loi choisie est une **loi uniforme discrète sur des valeurs non-numériques**. Nous l'avons choisi car nous avons **4 possibilités** bien distinctes pour ce tirage aléatoire, chacune d'elle ayant **la même chance** que les autres d'apparaître.
- Voir la partie 2 : "lois de probabilité" pour le détail de l'algorithme.
- Aucun paramètre extérieur pris en compte, si ce n'est le nombre de partiels possible (4).

b) Semestre 1 : Note du Semestre 1

- Modélise la note obtenue par le joueur au partiel du Semestre 1
- La loi choisie est **une fonction Gaussienne mêlée à une méthode de rejet**. En effet, nous voulions que le joueur ait plus de chance de tomber sur une fourchette de notes,

mais puisse également avoir une mauvaise note malgré ses capacités. Une méthode de rejet nous semblait évidente, et une fonction gaussienne illustre bien les probabilités de la vie réelle. La fonction gaussienne est donc centrée sur la fourchette donnée (par exemple si la fourchette donnée est 6-8, la fonction gaussienne est centrée en 7), et est majorée par 10, étant la note maximale.



- Voir la partie 2 : “lois de probabilité” pour le détail de l’algorithme.
- Les paramètres pris en compte pour déterminer cette variable sont **les intervalles de note** possibles, la **matière du partiel**, ainsi que le **type d’IMAC** du joueur. Nous prenons aussi en compte le **centre** μ de la loi (défini en fonction de la matière et du type d’IMAC), **l’écart-type** σ (défini à 2). *Voir la partie III “structures de corrélation” pour plus de détails.*

c) Semestre 2 : Probabilité d’augmentation du skill de Science

- La possibilité d’augmenter son skill en science ou pas (booléen)
- La loi choisie est une **loi uniforme continue**, nous l’avons choisi car elle permet simplement de déterminer où nous nous situons sur le **pourcentage de chance** d’obtenir un bonus de skill.
- Voir la partie 2 : “lois de probabilité” pour le détail de l’algorithme.
- **Aucun paramètre** particulier pour modéliser cette loi. En revanche, le verdict en fonction du résultat de la simulation repose sur le **type d’IMAC** et la **méthode de révision** choisie. *Voir la partie III “structures de corrélation” pour plus de détails.*

d) Semestre 2 : Note du Semestre 2

- La note obtenue au partiel de signal du Semestre 2
- Nous déterminons cette note via **une loi uniforme continue** *(pour le pourcentage de chance d’obtenir une note satisfaisante)*, que nous mêlons à une **loi de Bernoulli**, car déterminons le **nombre de succès** (exercices réussis) sur le nombre de simulation (5).
- Voir la partie 2 : “lois de probabilité” pour le détail de l’algorithme.

- Les paramètres pris en compte pour ces lois sont simplement **les extrémités a et b de notre intervalle** pour **la loi continue** (0 à 1)**.** Ainsi que le **skill en science** du joueur pour la loi de **Bernoulli.** Nous *Voir la partie III "structures de corrélation" pour plus de détails.*

e) <u>Semestre 3 : Position en Y du sel et du love</u>

- La position Y des items love et sel dans le mini-jeu
- La loi choisie est une **loi uniforme continue**, car nous voulons déterminer de manière **équiprobable** à **quelle position Y** nos items (sel/love) tomberont sur la zone de jeu. Ils peuvent apparaître sur toute la zone de l'espace dédié.
- Voir la partie 2 : "lois de probabilité" pour le détail de l'algorithme.
- Les paramètres pris en compte pour cette loi sont simplement **les extrémités a et b (0 et 1) de notre intervalle** fini. Le position est recalculée a posteriori en fonction du résultat de la simulation et de la taille du canva.

f) <u>S4 (si débloqué) : Probabilité d'avoir un bonus</u>

- La possibilité d'avoir un bonus de la part de M.Biri
- La loi choisie est une **loi Gaussienne** centrée sur 6, car il faut que notre interlocuteur soit assez éméché, mais pas trop ! En donnant une valeur autour de 6, il y a plus de chance qu'on nous offre un bonus.
- Voir la partie 2 : "lois de probabilité" pour le détail de l'algorithme.
- Les paramètres pris en compte pour cette loi sont le **centre** μ de la loi (défini à 6), **l'écart-type** σ (défini à 2) ainsi que la valeur du **nombre de verres** de shots donnés par le joueur. *Voir la partie III "structures de corrélation" pour plus de détails.*

g) <u>S4 (si bonus donné) : Valeur du bonus</u>

- La valeur du bonus donné par M.Biri, entre 0.01 et 2 points.
- La loi choisie est une **loi uniforme continue**, car M.Biri peut nous augmenter notre moyenne de manière aléatoire entre 0.01 et 2 points Ce bonus peut prendre n'importe quelle valeur entre ces deux nombres.
- Voir la partie 2 : "lois de probabilité" pour le détail de l'algorithme.
- Les paramètres pris en compte pour cette loi sont simplement **les extrémités a et b (0 et 1) de notre intervalle** fini.

Pour le détail des **<u>algorithmes</u>** utilisés pour modéliser les lois de probabilité, nous vous invitons à consulter la partie ci-dessous.

2) <u>Les lois de probabilité</u>

a) <u>Loi normale n°1 + Méthode de rejet :</u>

Continue ; $\sigma = 2$ & μ = *défini en paramètre grâce aux skills* **;**

```
/*Une fonction gaussienne est sous le forme
f(x)= 1/ sigma*racine(2pi)  * e^{- (x-mu)^2  / 2\sigma^2
On a donc pris sigma = 2 et mu = center
afin que notre courbe gaussienne soit un peu étalée et centrée en la valeur souhaitée
On multiplie le tout par 20sqrt(2pi) afin que le max soit égal à 1
*/
function gauss(center, x){
    return (20*Math.sqrt(2*Math.PI) *Math.exp(- Math.pow(x-center, 2)/8))/(2*Math.sqrt(2*Math.PI));
}
```

```
function algoRejet(centerGauss){
    //Y peut etre compris entre 0 et 10
    let y = Math.random()*10;
    const u = Math.random();
    while(u > (gauss(centerGauss, y)/10)){
        y = Math.random()*10;
    }
    /*On aura plus de chance de tomber sous la courbe aux alentours de la valeur donnée*/

    return y;
}
```

Explication : Nous utilisons cette méthode de rejet lors de la détermination de la note du premier partiel. Cette note dépend du type d'IMAC sélectionné par le joueur. Nous avons défini un intervalle de note possible en fonction du partiel et du type d'IMAC sélectionné. Le centre de la courbe de Gauss dépend du type d'IMAC également.

L'intérêt de la méthode de rejet ici est donc de prendre un Y aléatoire compris entre 0 et 10. On tire ensuite U selon la loi uniforme U(0;1), indépendamment de Y. Dès qu'on trouvera un Y donc f(Y) est inférieur à U, on retourne Y. Ici, les Y sélectionnés ne peuvent donc être que sous la courbe de Gauss codée et nous permet d'échantillonner notre résultat en faisant attention à ne pas avoir une chance équiprobable des notes. La note sur laquelle la courbe est centrée aura bien évidemment plus de surface sous la courbe que le reste des notes, et il sera donc plus probable de tomber sur celle-ci.

b) Loi normale n°2 (GaussBool) :

Continue ; $\sigma = 2$ & $\mu = 6$ (*défini en parametre*) **;**

```
function gaussBool(center, x, a,b){
    var val =(20 * Math.sqrt(2 * Math.PI) * Math.exp(- Math.pow(x - center, 2) / 16)) / (2 * Math.sqrt(2 * Math.PI));
    return (val >a && val<b);
}
```

Explication : On utilise à nouveau une courbe de Gauss, dont les paramètres sont différents ($\mu = 6$) et dont la fonction dans laquelle est incorporée la loi retourne un booléen. Nous définissons en paramètre un min et un max (a ou b) en dehors desquels la fonction nous retourne faux, sinon vrai.

c) **Loi de Bernoulli :**

Discrète - Tirage aléatoire à deux résultats de probabilité respective p et 1-p.

Nous utilisons la loi de Bernoulli pour déterminer en fonction de notre probabilité égale au **Skill en science/5** (le skill allant de 1 à 5). *Voir la partie d pour le code complet.*

d) **Loi Binomiale :**

Discrète - Mixée avec la loi de Bernoulli, retourne le nombre de succès obtenus pour n répétitions.

Notre **loi de Bernoulli** prend en paramètre le nombre de tirage à réaliser (**5** puisque nous avons **5 exercices de 2 points chacun**).

Dans notre cas, **la loi de Bernoulli et la loi binomiale sont mixées** dans la même fonction ci-dessous :

```
const data = bernoulliAndbinom(5, getSkillScience()/5);
```

```
function bernoulliAndbinom(n, p){
    let data = new Array();
    let lastChance= 0;
    for(let i=1; i<6; i++){
        data[i] = lastChance + coefBinomial(n, i)* Math.pow(p, i) * Math.pow((1-p), (n-i));
        lastChance = data[i];
    }
    return data;
}

function coefBinomial(n, k){
    return factoriel(n)/(factoriel(k)*factoriel(n-k));
}

function factoriel(n){
    if (n<0) {
        alert ("Veuillez Saisir Un Entier Positif ou null");
        return "### Erreur ###";
    }else {
        if (n == 0) {
            return 1;
        }else {
            return n * factoriel (n-1);
        }
    }
}
```

Explications : On combine dans la première fonction le calcul de la **loi de Bernoulli**, tandis que nous détournons la **loi binomiale** pour recueillir le **nombre de succès** dans un tableau. En fonction de ce nombre de succès, nous déterminons quelle note est obtenue à la fin du partiel de signal.

e) **Loi uniforme continue :**

Continue - Prend une valeur entre a et b de manière équiprobable.

Cette loi génère un **nombre aléatoire entre a (inclus) et b (exclus) sans interruption de manière équiprobable** entre les deux. Nous l'utilisons régulièrement dans ce projet pour calculer des positions, des notes ou des bonus par exemple.

```
function uniform(min,max){
    return Math.random() * (max - min) + min;
}
```

f) **Loi uniforme discrète :**

Discrète - Sélectionne un des partiels parmi les 4 disponibles au Semestre 1.
Retourne une valeur non-numérique (matière du partiel que le joueur va passer).

Nous générons une loi uniforme qui va nous retourner **des entiers entre min et max** (max étant **le nombre d'issues** pour cette loi). Elle nous retourne ensuite la matière du partiel correspondant à cet entier. Pour cela, nous générons simplement une **loi uniforme continue,** et nous tronquons à l'entier près.

```
function uniformPartiel(){
    //Discrète, retourne le type de partiel à passer (non-numérique)

    partiel = Math.round(Math.random() * (4 - 1) + 1);
    if (partiel == 1) { // HDA
        return "HDA"
    } else if (partiel == 2) { //TECH
        return "TECH"
    } else if (partiel == 3) { //SIGNAL
        return "SIGNAL"
    } else { //PROG
        return "PROG"
    }
}
```

III/ Structures de corrélation

La plupart des structures de corrélation de notre jeu dépendent soit des statistiques du joueur, soit des choix qu'il réalise tout au long de sa partie. En voici la liste :

a. **Semestre 1 - Le type d'IMAC, le partiel du Semestre 1 et la note :**

La matière du partiel que le joueur devra passer au semestre 1 est tiré de manière totalement aléatoire et équiprobable. La **note** qu'il obtient à ce partiel dépend de la matière sélectionnée AINSI que du type d'IMAC qu'il incarne.

Par exemple, si le joueur tombe sur le partiel "traitement du signal" et que son profil est "scientifique", il a plus de chance d'avoir une note entre 7 et 10 qu'entre 0 et 5.

```
function update(type, matiere){
    let note;
    switch(matiere){
        case window.matiere.SIGNAL : note = getSignalResult(type);
        break;
        case window.matiere.PROG : note = getProgResult(type);
        break;
        case window.matiere.HDA : note = getHDAResult(type);
        break;
        case window.matiere.TECH : note = getTechResult(type);
        break;
        default: note = 0; break;
    }

    setNote(Math.trunc(note*10)/10);

    setTimeout(() => {  updateDisplay(matiere); }, 2000);
}
```

Ici, on appelle la fonction qui nous donne la note obtenue au partiel du Semestre 1 en fonction des paramètres **type** (type d'IMAC) et **matière** (matière du partiel). Les fonctions permettant de déterminer cette note ont alors la structure ci-dessous qui nous donne un intervalle de notes possible en fonction de notre type d'IMAC. *(exemple avec le partiel de Signal)* :

```
function getSignalResult(type){
    switch(type){
        case window.type.ART: return getInRange(2, 5);
        break;
        case window.type.SCIENCE: return getInRange(7, 10);
        break;
        case window.type.SURVIE: return getInRange(0, 5);
        break;
        case window.type.POLYVALENT: return getInRange(4, 7);
        break;
        default: break;
    }
}
```

b. **Semestre 2 - Le choix de la méthode de révision, le type d'IMAC, et le gain de skill :**

Au semestre deux, nous pouvons choisir trois modes de révision différents pour préparer le partiel suivant. En fonction de ces **modes de révision** ET de **notre type d'IMAC**, nous avons **plus ou moins de chance** d'**améliorer nos compétences en Science**. Exemple ci-dessous lorsque nous avons choisi d'aller en soirée :

```
function soiree(){
    estSorti = true;
    let probaChanceAugmentée;
    switch(getTypeInt()){
        case 1: probaChanceAugmentée = 0.8; // Science
        break;
        case 2: probaChanceAugmentée = 0.3; // Prog
        break;
        case 3: probaChanceAugmentée = 0.3; // Art
        break;
        case 4: probaChanceAugmentée = 0.45; // Comm
        break;
        default: break;
    }
```

Note : L'IMAC scientifique a plus de chance d'avoir une bonne note. L'IMAC communiquant lui, sait très bien parler et peut donc broder autour du peu de connaissances qu'il a ! ;)

c. **Semestre 2 - La note et le skill en Science :**

En fonction de notre actuel **skill en science** (amélioré ou pas), nous avons p**lus ou moins de chance** d'avoir une bonne note en signal.

```
const data = bernouilli(5, getSkillScience()/5);
```

d. **S4 - Avoir un bonus de M.Biri et le nombre de verres offerts :**

En fonction du **nombre de verres** que nous offrons à M.Biri pendant le gala, il y a plus ou moins de chance qu'il nous donne un petit bonus sur notre moyenne générale. **La loi gaussienne décidant ceci est centrée sur 6.**

```
if (gaussBool(6, nbShot, 5, 8) == true) {
```

IV / Difficultés rencontrées et commentaires

En ce qui concerne les difficultés rencontrées durant le projet, nous avons eu du mal au début à **déterminer comment coder correctement les lois** que nous utilisions. Avec un peu de recherche et de travail nous avons réussi à surmonter ce problème et obtenir de véritables lois aléatoires.

La seconde chose qui nous a posé problème était **les exigences du cahier des charges quant aux types de lois à implémenter** : En effet, nous savons implémenter des lois continues, des lois discrètes, numériques ou non, mais le problème est qu'utiliser **spécifiquement 4 discrètes et une loi continue au minimum** n'est pas très pertinent dans notre sujet.

Comme vous avez pu le voir, nous générons des notes, des probabilités (chance) ou la position de certains éléments la plupart du temps. C'est pourquoi avoir des lois discrètes pour ces variables aléatoires ne correspondait pas du tout à notre jeu, nous avons utilisé plutôt des **lois continues**. **Nous avons donc pris la liberté de ne pas intégrer à notre jeu des lois superflues à son fonctionnement.**

En revanche, vous pourrez retrouver en **annexe** quelques algorithmes supplémentaires que nous avons implémentés. Ils n'ont pas été utilisés dans notre jeu mais nous avons pris le temps de les réaliser pour que vous puissiez **attester de nos compétences** et **de notre compréhension des lois de probabilité.**

Une autre des difficultés rencontrée a été **la réalisation du mini-jeu du sel et du love** qui n'a pas été une mince affaire à coder et à animer, mais nous sommes fières d'avoir pu créer un jeu fonctionnel et nous espérons, agréable !

Enfin, **notre gestion du temps et les circonstances sanitaires actuelles** nous ont évidemment posé problème, puisque beaucoup de rendus ont été décalés à cette période, et que notre avancement n'a pas pu être aussi rapide qu'en présentiel. Cependant, nous avons pu délivrer **un jeu complet,** tout en voyant (enfin !) **la mise en application des mathématiques** dans le domaine du développement et du jeu vidéo.

V/ Annexe : Lois codées non-exploitées

Ici vous trouverez toutes les lois que nous avons codées et étudiées mais que nous n'avons pas ajouté dans notre jeu par manque de pertinence sur les événements que nous souhaitions générer.

a) Loi Géométrique

Discrète

```
function geometrique(n,p,q){
    // Probabilité d'obtenir dans une succession de k épreuves de Bernoulli, k – 1 échecs suivis d'un succès.
    // Les épreuves sont indépendantes, cette probabilité est de qk – 1p.

    // Loi discrète

    return Math.pow(q,  n-1)*p;
}
```

b) Loi de Poisson

Discrète

```
function Poisson(n,m){
    // Probabilité d'avoir n évènements distribués selon une loi de Poisson de paramètre m
    return Math.pow(exp(1), -m)*(Math.pow(m,n)/factoriel(n));

}
```

c) Loi logarithmique

Discrète

```
function logarithme(p,k){
    // Loi de probabilité discrète, dérivée du développement de Taylor
    result = Math.pow(p,k)/k;
    result = result*(-1/Math.log(1-p));

}
```