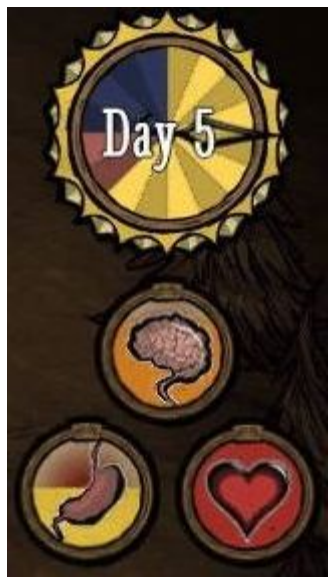
Partie 1 : Note technique

1.1 : Principe général

Dans ce rpg de survie inspiré du jeu Don't Starve développé par Klei Entertainment, le joueur incarne un personnage abandonné sur une île déserte. La carte et les déplacements dépendent d'un ensemble de cases indexées. Toute case appelée par un numéro négatif ou supérieur à 144 sera associée par défaut à une case "dans l'eau", donc inaccessible. L'équilibrage du jeu fut compliqué (tantôt trop facile, tantôt trop dur). On a cependant considéré la difficulté acceptable dans la mesure où ce type de rpg tient du "die and retry", où l'arrêt de la partie peut survenir à tout moment suite à un mauvais choix opéré par le joueur, l'obligeant à recommencer et agir ou réagir d'une manière différente de l'échec précédent afin de progresser.



Le joueur doit tenter de survivre et de s'enfuir en tenant compte des données suivantes : le temps s'écoule (et il vaut mieux être éclairé lorsque la nuit tombe…) ; la santé physique importe (une blessure est si vite arrivée avec toutes les créatures étranges qu'on peut croiser ici…) ; la santé mentale importe également (difficile de garder son calme, surtout en mangeant n'importe quoi…) et surtout, surtout : *il ne faut pas mourir de faim*.

Notre code étant un rpg textuel en français, le code a également été réalisé et commenté en français.

1.2 : Mode d'emploi et commandes d'introduction

Nous proposons une fonction mode_emploi qui suit le même principe que le code fourni en exemple, à ceci près qu'elle a été modifiée pour correspondre à nos prédicats.

Le prédicat compteRendu/0 permet à tout moment d'obtenir un résumé de la situation (temps restant avant la nuit, vie, mental, faim, position sur la carte…). Il est régulièrement appelé automatiquement pour que le joueur ait conscience de sa situation mais il peut aussi être appelé par le joueur, ce sans consommation d'unités temporelles.

Enfin, demarrer/0 affiche une introduction du jeu, ainsi qu'une première description de la case où se trouve le joueur et des indications sur les actions à sa disposition.

1.3 : Prédicats dynamiques du jeu

Le principe des prédicats dynamique fourni dans le code de base (je_suis_a/1, il_y_a/2, vivant/1) a été conservé et élargi en particulier aux différentes variables d'état et de ressources nécessaires dans notre jeu. On définit également nombre/1 qui correspond à un compteur que nous utilisons dans différentes fonctions de dénombrement ou de vérification (comme nbDrop/1 ou caseVide/2). De même, un prédicat dynamique choix/2 à été créé pour les prédicats proposants un choix à l'utilisateur. Ces prédicats affichent les différents choix que l'utilisateur a, et pour chaque choix, on assert(choix(NumeroDeChoix,ChoixCorrespondant)). Ainsi, le joueur peut entrer un numéro grâce au prédicat read/1, et on est en mesure de lier ce numéro au choix du joueur.

Une série de retractall/1 permet de réinitialiser l'ensemble des prédicats dynamiques au chargement du jeu.

1.4 : Initialisation des éléments, des cases, des drops, ...

On initialise par exemple :

- la position du joueur (au centre de l'île, case 54)
- la compétence de combat du joueur et de ses potentiels adversaires selon le principe suivant : combat(combattant,points).
- la position des créatures mobiles (et des arbres, par facilité de code compte tenu que l'un des arbres - arbre5 - est capable de se défendre) selon le principe suivant : estSur(créature , position sur la map).
- le statut (vivant/mort) des créatures selon le principe suivant : vivant(créature).
- la caractérisation de certains éléments (exemple : araignee(araignee1). ou ramassable(carotte). ou brulable(buche).)
- la position des objets ramassables sur la carte, selon le principe suivant : estSur(Objet ramassable , position sur la map, nombre).
- les informations culinaires:
  - apportNutritionnel(element,valeur).
  - toxique(élément,toxicité).
  - cuisinable(élément pré-cuisson,élément post-cuisson).
- les drop, c'est à dire les éléments qui apparaissent sur la carte suite à un combat ou une interaction selon le principe suivant : drop(Entité, Objet ramassable, nombre).
- des caractérisations plus génériques (ennemi/1, mobile/1, equipable/1) pour traiter les différents éléments du jeu
- ...

1.5 : Outils de syntaxe

Cette partie du code nous permet d'appeler (partout ailleurs dans le code) des noms définis ou indéfinis, de gérer le pluriel, ... et plus généralement de faciliter la lecture de notre jeu en évitant les fautes. Ainsi, on voit "une araignée" mais on attaque "l'araignée".

1.6 : Temps, attente, compteurs

Des "retract" et "assert" sont employés pour gérer le temps écoulé, le temps restant avant la nuit, etc. On considère donc des unités temporelles (initialement 100) pour composer une journée. La nuit survient quand ces unités se sont écoulées. La nuit est "instantanée" et le joueur ne joue pas pendant la nuit.

Le temps s'écoule via le prédicat attendre/1 qui (en fonction du temps attendu passé en paramètre) agit sur la faim, les déplacements, et plus généralement sur les différents compteurs.

Les fonctions de calcul calculFaim/3 et calculDeplEnnemis/2 permettent respectivement de gérer la faim en fonction de l'attente et de calculer le nombre de déplacement effectués par les monstres NPC en fonction de l'attente.

1.7 : Gestion des ressources du joueur (pv, faim, moral)

Dans cette partie du code, nous travaillons sur les trois jauges (pv, faim, moral) du joueur. Les fonctions blesser/1 et soigner/1 affectent les points de vie. A chaque perte de points concernant la jauge de faim, on appelle le prédicat famine/1. En cas de “famine” (jauge de faim tombée à zéro ou moins), famine/1 utilise affame/1 pour mettre la jauge à zéro et blesser le joueur.

Le joueur utilise manger/0 lorsqu’il veut manger. Ce prédicat réagit alors en fonction des éléments disposant d’un apport nutritionnel présents dans l’inventaire pour proposer au joueur une liste de choses à manger s’il y en a et -en fonction du choix- appeler manger/1.

Le moral du joueur - ou sa santé mentale - est représenté par la jauge ‘mental’. Certaines actions considérées comme stressantes peuvent retirer du mental au joueur à l’aide du prédicat degenerer/1. Si le mental du joueur tombe à 0, degenerer/1 appelle fou/0, qui explique la situation au joueur, lequel meurt de folie (fou appelle pour cela blesser(100)).

Dans le jeu Don’t Starve dont nous nous sommes librement inspiré, la folie permet de complexifier grandement le jeu : les éléments ne réagiront pas de la même façon si le joueur approche la folie, ce qui participe à l’ambiance du jeu et à sa difficulté. Notre but en ajoutant la jauge de mental était similaire ; changer l’expérience de jeu en fonction de la folie du joueur. Nous n’en avons cependant pas eu le temps, et la folie telle qu’elle est implémentée dans notre projet n’est qu’une jauge supplémentaire qui augmente la difficulté en donnant une nouvelle butée au joueur : il doit s’enfuir avant de perdre la raison. Une version suivante du jeu serait d’ajouter des fonctionnalités liées à la folie. De la même façon que nous appelons nomDefini, nomIndefini ou nomPluriel, il serait, par exemple, possible de modifier les descriptions des objets quand le joueur approche la folie (“Pourquoi est-ce qu’une carotte ressemble à un bureau ?”). Un facteur aléatoire dépendant de la santé mentale du joueur peut aussi être ajouté à certaines actions ; par exemple si le joueur décide de manger une carotte se trouvant dans son inventaire, sa folie peut le faire manger une bûche à la place et se briser les dents.

Notons que les trois jauges disposent d’un nombre de point maximum (en l’occurrence 100) qui ne peut pas être dépassé, ce quelles que soient les actions du joueur.

1.8 : Gestion des combats, nombre d'ennemis, drop

On distingue les notions attaquer, combattre et tuer. Le prédicat attaquer/0 correspond à l'attaque par défaut. Il fonctionne lorsqu'il n'y a pas de doute possible sur la cible (ie : au moins une créature ennemie se trouve sur la case). Ainsi, une série de prédicats attaquer/1 sont définis en fonction du type d'Objet attaqué. Ensuite, combattre/1 permet de gérer la différence de force entre les adversaires, et les éventuelles blessures encaissées à l'issue du combat. Enfin, tuer/1 dépose tous les "drop" au sol et agit sur le statut (vivant/1, estSur/2) de l'ennemi.

Notons que l'emploi de attaquer/0 face à deux ennemis ou plus entraîne une mort systématique..

1.9 : Gestion de l'inventaire

Les règles pour ramasser et déposer un objet ont un principe similaire à celui du code de base mais ont été généralisées pour qu'on puisse utiliser ramasser/0 et pour gérer les objets existant en plusieurs exemplaires. Par ailleurs, jeter/0 permet au joueur (via un choix puis un appel à jeter/2) de jeter des éléments de l'inventaire.

1.10 : Gestion du craft et des recettes

On retrouve trois prédicats principaux dans cette partie :

- recette/1 définit s'il existe ou non une recette pour un objet.
- dansRecette(Objet,Ingrédient,Nombre) permet d'identifier les éléments nécessaires à la réalisation d'une recette.
- dansListeRecettes(objet) définit si la recette pour un objet est - ou non - connue par le joueur.

Ainsi, un appel à “listeRecette.” permet d’obtenir la liste des objets dont la recette est connue et un appel à “consulterRecette(Objet)” permet (si la recette est connue) de voir les ingrédients.

Le predicat “craft(Objet).” vérifie qu’une recette existe pour cet objet, que la recette est connue et que le joueur dispose des ingrédients en nombre suffisant. L’objet fabriqué va alors soit se retrouver dans l’inventaire, soit être équipé via equiper/1 (s’il s’agit d’une lance puisqu’elle influence les points de combat du joueur), soit aller sur la case (dans le cas de l’allumage d’un feu, via allumerfeu/0).

1.11 : Cuisson des aliments

Le prédicat cuire/0 fonctionne comme manger/0 : il vérifie qu’il y a quelque chose à cuire dans l’inventaire du joueur, lui demande ce qu’il veut cuire puis appelle cuire/1 pour procéder à la cuisson. Ce prédicat applique un temps de cuisson (attendre(4).), et soit un changement dans l’inventaire (l’élément cuit change de nom via cuisinable/2), soit un simple retrait de l’inventaire (via brulable/1).

1.12 : Observation, descriptions, interface utilisateur

Le principe de description a été généralisé à décrire/1 qui décrit une case en particulier (à partir d’un indice).

1.13 : Chester, cartes au trésor et fin du jeu

Chester se trouve en case 24, au nord-ouest de la carte. L’ouvrir permet de trouver la recette du bateau qui permettra au joueur de voguer (enfin !) vers son salut.

Ainsi, ouvrir/0 permet au joueur (lorsqu’il est sur la bonne case) d’obtenir une nouvelle recette en la ramassant.

1.14 : Gestion de la map, des déplacements

Le principe de définition de l'environnement fourni dans le code de base a été largement modifié dans notre code : nous utilisons une map codée par un rectangle de 12*12 cases dont certaines - dans l’eau - sont inaccessibles. Quatre fonctions (alest/2, alouest/2, …) nous permettent d’identifier le rapport entre deux indices de cases. La fonction adjacentes/2 permet ensuite non-seulement d’identifier deux cases adjacentes mais également de vérifier qu’on s'intéresse à des cases émergées (et donc bien accessibles sur l’île qui nous intéresse).

Les appels simplifiés proposés dans le code fourni (n/0, s/0, o/0 et e/0) ont été conservés mais

la fonction de déplacement du joueur (aller(Direction)) a été complexifiée dans la mesure où un déplacement influe sur le temps qui s'écoule (et appelle par conséquent la fonction attendre/1).

Le prédicat decrire/1 affiche la liste des éléments présents sur une case donnée et appelle lorsque c'est nécessaire une description plus poussée (exemple : decrireChester/0, decrireChien/1).

Contrairement à l'environnement statique du code de base fourni en exemple, certaines créatures de notre jeu sont mobiles : elles se déplacent dans une direction aléatoire lorsque le joueur consomme un certain nombre d'unités de temps. On utilise alors la fonction deplacer/1 qui va chercher une direction aléatoire (via randomDirection/1) jusqu'à trouver un chemin (chemin(Position,Direction,Case d'arrivée)) possible.

Partie 2 : Notice d'utilisation (mode_emploi.)

Il est à noter que attendre/1 est le seul prédicat appelable par le joueur nécessitant un argument. L'argument attendu est une valeur numérique entière. A titre indicatif, une journée dure 100 unités de temps. Attendre sans rien faire risque d'influencer (en mal !) la jauge de faim, voire les jauges de vie et de moral. Cela étant dit, utiliser attendre/1 avec un paramètre trop grand va se contenter d'appeler tempsEcoule/0 et nuit/0. Le trop-plein ne sera pas pris en compte.

Les autres commandes sont décrites ci-dessous :

```
?- mode_emploi.

Entrez les commandes avec la syntaxe Prolog standard.
Les commandes disponibles sont :
demarrer.                   -- pour commencer une partie.
n. s. e. o.                 -- pour aller dans une direction.
regarder.                   -- pour regarder autour de toi.
inventaire.                 -- pour regarder ton inventaire.
ramasser.                   -- pour ramasser tous les objets a disposition.
jeter.                      -- pour laisser tomber un objet.
listeRecettes.              -- afficher les recettes que tu connais.
consulterRecette(X).        -- pour affiche la recette de l'objet X.
craft.                      -- pour fabriquer un objet de ta liste de recettes.
manger.                     -- pour manger.
cuire.                      -- pour cuire de la nourriture.
attaquer.                   -- pour attaquer un ennemi.
ouvrir.                     -- pour ouvrir quelque-chose ?
mode_emploi.                -- pour afficher ce mode d'emploi de nouveau.
halt.                       -- pour terminer la partie.
compteRendu.                -- pour afficher les stats, la position et le temps restant avant la nuit.
attendre(duree).            -- pour ne rien faire pendant un certain temps
afficherMap.                -- pour afficher la map (avec une croix pour savoir oÃ¹ on est)

true.
```

Bordeaux INP
ENSC

Partie 3 : Extraits de session de jeu

Début de partie :

```
?- demarrer.


Dans ce jeu, tu incarnes Wilson, un gentleman scientifique intrepide et ambitie
ux qui s'est tres recemment retrouve dans un univers sauvage et mysterieux suit
e a une altercation avec un demon denomme Maxwell. Abandonne sur une ile desert
e, il te faudra tenter de survivre, et chercher un moyen de te sortir de ce mau
vais pas... Tu n'as pas l'air au meilleur de ta forme. Tu ferais bien de trouve
r quelque chose a te mettre sous la dent avant que la nuit ne tombe ...

Au cours de la partie, tu devras tenir compte des donnees suivantes : le temps
s'ecoule (et il vaut mieux Ãªtre eclaire lorsque la nuit tombe !), la sante phy
sique importe (une blessure est si vite arrivee avec toutes les creatures etran
ges qu'on peut croiser ici !), la sante mentale importe egalement (difficile de
 garder son calme, surtout en mangeant n'importe quoi...) et surtout, surtout :
 la faim.
Ne meurs pas de faim.


 ## DEBUT DE LA PARTIE ##

Tu es au milieu de nulle part. tu peux voir :
* 2 branches


> pour regarder autour, utilise 'regarder.'.
N'hesite pas a souvent regarder autour de toi, tu pourrais rater des elements i
mportants ou te jeter dans un piege...
> pour avoir une idee de la carte du monde, utilise 'afficherMap.' ;
> pour voir le mode d'emploi, utilise 'mode_emploi.'
true.

?-
```

Regarder autour de soi :

```
?- regarder.

A proximite immediate, tu peux voir :
* une araignee

Au nord, tu peux voir :
* un champignon

Au sud, tu peux voir :
* un lapin

A l'est, on dirait qu'il n'y a rien d'interessant.

A l'ouest, tu peux voir :
* un lapin
* une carotte

Que fais tu ?
> pour avoir une idee de la carte du monde, utilise 'afficherMap.'
> pour voir le mode d'emploi, utilise 'mode_emploi.'
true.

?-
```

Se battre :

```
?- attaquer.
Tu es blesse au cours du combat ;

 Tu as perdu 30 points de vie.
C'etait un beau combat ! Autour du corps de l'araignee tu trouves...
- une toile(*1)
- un steack de monstre (*2)


true.

?-
```

Passer la nuit :

```
?- craft(feu).

Tu allumes un feu, te voila en securite pour la nuit
true.

?- attendre(14).
tu peux voir :
* un feu
* une araignee
* un lapin

La nuit tombe !
Tu te couche a cote de ton feu pour passer la nuit. Au petit matin, tu te revei
lle (plus ou moins) frais et dispo ! Ton feu s'est eteint pendant la nuit.

Une nouvelle journee commence ! Explore l'ile pour trouver une issue, et n'hesi
te pas a utiliser 'regarder.' pour ne pas rater d'indices...

false.

?-
```

Mourir en combat :

```
?- e.
tu peux voir :
* un grand oiseau

Il s'est ecoule 5 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 15
pv : 100
mental : 100
faim : 82
case actuelle : 99


Le grand oiseau a l'air tres agressif, et ne perd pas une seconde pour t'attaqu
er. Tu n'as plus le temps de fuir, le combat est inevitable.
Tu es blesse au cours du combat ;

 Tu as perdu 100 points de vie.



TU ES MORT !!! Partie terminee.
Tape la commande halt
% Execution Aborted
?-
```

Craft un objet :

```
?- craft(hache).

Tu as reussi a craft une hache
on dirait qu'il n'y a rien d'interessant.

Il s'est ecoule 2 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 23
pv : 100
mental : 100
faim : 85
case actuelle : 25

true.

?-
```

Faire un feu, cuire un aliment et le manger :

```
?- craft(feu).

Tu allumes un feu, te voila en securite pour la nuit
true.

?- cuire.

Que veux tu cuire ? (entre le numero de ton choix)

1 : une toile
2 : un steack de monstre
3 : un champignon
4 : une carotte
5 : une branche
6 : une buche

|: 4.

la carotte cuit sur le feu. Tu obtiens une carotte cuite .tu peux voir :
* un feu

Il s'est ecoule 4 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 30
pv : 70
mental : 100
faim : 71
case actuelle : 50

true.

?- manger.

Que voulez vous manger ? (entrez le numero de votre choix)
1 : un steack de monstre
2 : un champignon
3 : une carotte cuite

|: 3.

tu as recupere 20 points de faim en mangeant la carotte cuite , ta faim est mai
ntenant a 91

Manger te soigne.
Tu as recupere 10 pv. Tu as maintenant 80 pv
```

Je voudrais couper cet arbre…

```
?- e.
tu peux voir :
* un arbre

Il s'est ecoule 5 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 60
pv : 100
mental : 100
faim : 84
case actuelle : 26


true.
```

… Mais je n'ai pas de quoi fabriquer une hache…

```
?- craft(hache).

Faut pas vendre la peau du boeuf avant de l'avoir vole !
Il te manque des elements pour fabriquer une hache  !
> pour consulter la liste des recettes a ta disposition, utilise 'listeRecettes
' ;
> pour verifier ton inventaire, utilise la commande 'inventaire.'
> pour voir le mode d'emploi, utilise 'mode_emploi.'

Ici, tu peux voir :
* un arbre

Il s'est ecoule 1 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 59
pv : 100
mental : 100
faim : 84
case actuelle : 26

true.
```

…Il me faut donc du silex, je vais en chercher…

```
?- regarder.

A proximite immediate, tu peux voir :
* un arbre

Au nord, on dirait qu'il n'y a rien d'interessant.

Au sud, on dirait qu'il n'y a rien d'interessant.

A l'est, on dirait qu'il n'y a rien d'interessant.

A l'ouest, tu peux voir :
* 4 silex

Que fais tu ?
> pour avoir une idee de la carte du monde, utilise 'afficherMap.' ;
> pour voir le mode d'emploi, utilise 'mode_emploi.'
true.

?- o.
tu peux voir :
* 4 silex

Il s'est ecoule 5 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 54
pv : 100
mental : 100
faim : 82
case actuelle : 25


true.

?- ramasser.

Tu ramasses 4 silex
true.
```

… Je peux maintenant revenir couper l'arbre.

```
?- e.
tu peux voir :
* un arbre

Il s'est ecoule 5 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 49
pv : 100
mental : 100
faim : 80
case actuelle : 26


true.

?- craft(hache).

Tu as reussi a craft une hache
tu peux voir :
* un arbre

Il s'est ecoule 2 unite(s) de temps.
 ## petit compte-rendu ##
temps restant avant la nuit : 47
pv : 100
mental : 100
faim : 79
case actuelle : 26

true.

?- couper.

Autour de l'arbre fraichement coupe, tu trouves...
- une buche
- 5 branches
- 2 buches


tu peux voir :
* 5 branches
* 3 buches
```