ICT 306  
Projet de classe avec gestion AGILE

===

Nom provisoire: BMW (Bene-Male World)  
Simulation d'un monde peuplé   
de créatures "autonomes"

1. Introduction module ICT 306:

Comme indiqué dans le document <https://www.modulbaukasten.ch/module/306/4/fr-FR?title=R%C3%A9aliser-de-petits-projets-dans-son-propre-environnement-professionnel> , le module 306 doit permettre d'expérimenter:

* (1) Les ressources nécessaires pour un projet (notamment en termes de collaborateurs)
* (1) Les manques de formation des collaborateurs
* (2) Les limites de temps (volume horaire et délais limités)
* (3) La planification dans une grande équipe (ici AGILE iceScrum), le travail en parallèle
* (4) (5) Le suivi de l'avancement d'un projet avec des outils pour voir où on en est.
* (6) Les discussions avec le "client" (ici Product Owner)
* (7) la collaboration, les feed-back…etc
  + ici sprint review (focus sur le produit)
  + et sprint retrospective (focus sur le processus)

1. Introduction BM World:

Le projet choisi comporte les particularités suivantes:

* Un projet nouveau où il faut tout créer.
* Un projet choisi par le groupe classe par votation parmi plusieurs possibilités.
* Un projet pour lequel il n'y a pas encore toutes les compétences dans le groupe classe. (nécessité de prévoir des temps de formation)
* Un projet dont on ne sait pas à l'avance comment il va se terminer. (seul le sprint 1 est assez bien défini)

BM World vise à créer un monde simplifié, dans lequel des créatures vont évoluer de manière autonome avec des comportements plus ou moins éthiques, plus ou moins corrects ou plus ou moins crapuleux, en fonction de leur tendance. Les actes qu'ils choisiront de faire auront un impact sur leur état (et éventuellement sur leur tendance).

1. Description BM World:

Dans ce monde, réalisé en 3 phases (sprint) de plus en plus complexes:

* On aura une partie **géographie du monde** (init: un immense domaine plat dans lequel les individus se déplaceront et se rencontreront).
* Les créatures :
  + Auront un **profil** (tendance) qui déterminera une probabilité pour certaines actions
  + Sont capables de diverses **actions seul** (étudier, jouer, travailler, faire du sport, consommer des substances psy…)
  + Sont capables de diverses **interactions deux par deux** (discuter, jouer, voler…), à définir la distance à partir de laquelle ils peuvent interagir.
  + Seront capables **d'évoluer** en fonction des expériences qu'ils font (apprentissage)
* La **société** du monde aura établi des règles, avec des sanctions possibles. Elle aura un certain fonctionnement (par exemple dictature, anarchie, démocratie sociale, démocratie libérale…etc)

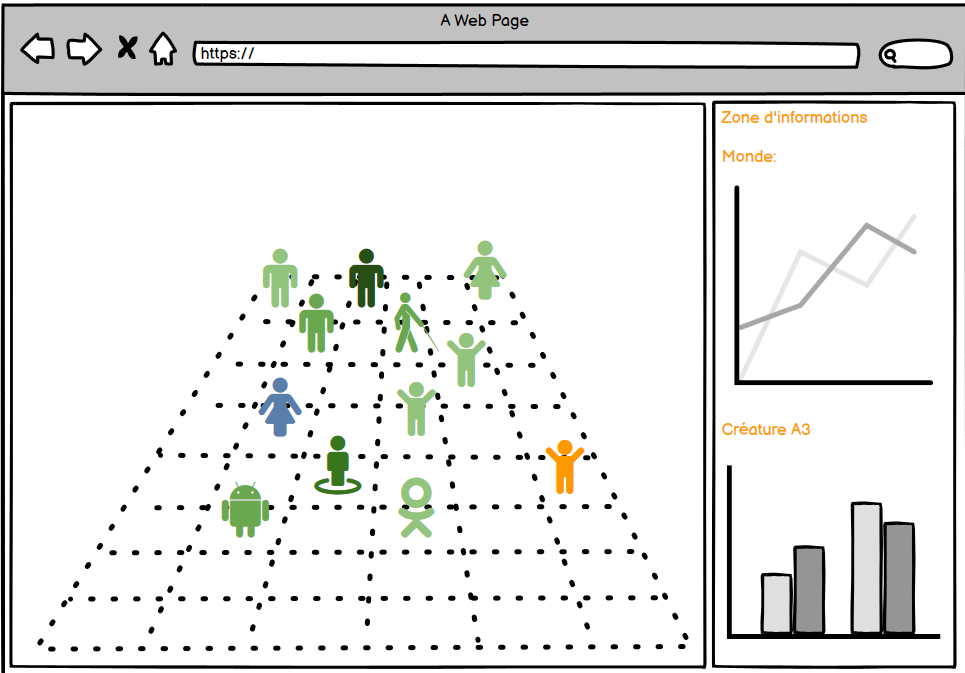
1. Organisation BM World:

En principe on partagera en 4 équipes (+ 1 coordinateur github et scrummaster). L'enseignant JCY restera le Product Owner et également une ressource par rapport à la programmation dans le langage choisi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Equipe "moteur" du monde  3x | Equipe représentation du monde  2-3x | Equipe "design"  1-2x | Equipe "comportement"  2x | ScrumMaster et github coordinateur  1x |
| Construire les fonctions/tableaux/mécanismes qui permettent de faire évoluer les créatures (bouger, agir, interagir) et le monde.  Créer un jeu de données de base. | Créer les fonctions affichant le monde + créatures d'après le tableau mémoire du monde. Permettre de suivre le monde et chaque créature (courbes) | Propose 2 design du monde et des créatures.  Après choix, on affine et l'équipe "représentation" devra utiliser ce design au sprint suivant. | Modéliser un comportement des créatures de plus en plus complexe, ainsi que plusieurs modèles de "société" (dictature, anarchie, …) Tester les modèles, exploiter les résultats | Assurer la coordination des équipes et la circulation de l'info.  Tenir icescrum à jour. Assurer la disponibilité des sources par github |
| engine.js  (javascript) | BMWorld.html  display.js  maquette  (html, css, javascript) | \image, modèles 2D ou 3D  (photoshop…) | Descriptions, propositions  Tableau "actions", "monde" | Icescrum github |

1. Description sommaire du sprint 1 (3 semaines max – 23 fév):

* Le monde sera un plateau simple (exemple un carré de 25x25 "cases". A une fréquence variable (1s ou accéléré…), les personnages se déplacent et peuvent choisir une action.
* Sa représentation sera à choisir (2D, 2D et demi, 3D…), voir point 10.



* **Les créatures** (par ex. 100) seront de simples bonhommes, un corps, une tête,... Une partie de leur état sera visible par couleur, mais on pourra accéder à plus d'informations en cliquant sur lui ou sur la case.

Pour simplifier (dans le sprint 1) et travailler juste sur le "comportement individuel et social", les créatures sont simplifiées. On les imagine à un âge encore malléable (par ex. 15 ans):

* + Pas besoin de sommeil, pas de cycle jour/nuit
  + Pas besoin de manger
  + Pas de famille
  + Pas de sexe masculin ou féminin
  + Pas d'âge
  + Pas d'appartenance à un groupe
  + Toutes les actions peuvent se passer n'importe où (pas de notion d'école, de terrain de jeu…etc), toutes les actions se font sur la durée de base (ex: 1 jour), simulée par exemple 1 fois par seconde.
* Les créatures auront un **profil** de départ, en particulier une tendance plus ou moins forte sur des axes, par exemple (de 0 à 1) :
  + I = immédiat, envie de plaisir immédiat (par. Ex jouer)
  + F = future, envie d'investir pour le futur (p. exemple travailler, étudier)
  + R = relation, envie d'investir dans des relations
  + P = possession, envie de posséder un max d'argent (AG) et d'objets (OB)
  + E = Ethique, plus le niveau est haut, moins la créature se permet des actions douteuses
  + Le profil est susceptible d'être modifié en fonction des expériences, des relations…
* Les créatures auront un **état** (par exemple de 0 à100), avec plusieurs composantes, par exemple :
  + FC = Force, quel état de force (dépend du sport)
  + CP = Compétences, dépend des études
  + RA = Richesse sous forme argent
  + RP = Richesse sous forme possession
  + BE = bien-être, qui dépend des activités et du profil.
  + RE = qualité/quantité de relations
* Les **actions** possibles (à compléter) (la probabilité peut aussi dépendre de l'état). Chaque individu fait au max 1 action par période (typique 1 jour).
  + ETU = Etudier (probable suivant 2\*F + E, conséquence CP+)
  + TRA = Travailler (probable suivant F + E, conséquence RA+ (prop au CP))
  + JOS = Jouer seul (probable suivant 2\*I – R, conséquence BE+, FC- ?)
  + SPS = Sport seul (probable suivant F, conséquence FC+ BE+)
  + VOS = Voler seul (probable suivant 2\*I-E, aléatoire RP+, RA+)
  + COS = Consommer des produits stupéfiants (prob 2\*I, conséquence FC-, CP-, -2RA, BE+)
  + JO2= Jouer à 2 (probable suivant I + R, conséquence RE+, BE+)
  + VO2 = Voler l'autre (probable suivant 2\*I-E, si FC + que l'autre, RP+, RE-, dépend si l'autre est riche)
  + DI2 = Discuter (probable suivant I+R, conséquence CP+, BE+, RE++…
  + SP2 = Sport à deux (probable suivant I+F+R, conséquence FC++, BE+, RE+)

Pour l'effet et la probabilité des actions, on peut utiliser un tableau comme ci-dessous. Il faudra encore imaginer un effet des actions et leur conséquence sur le profil.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Effet des actions** | | | | | | | **Probabilité des actions** | | | | |
|  | **FC  force** | **CP Compé-tences** | **RA  Argent** | **RP Posses-sions** | **BE bien-être** | **RE relations** | **Sanction** | **I Immédiat** | **F futur** | **R rlations** | **P posséder** | **E Ethique** |
| **ETU** |  | +2 |  |  |  | +1 |  |  | 2x |  |  | 1x |
| **TRA** |  | +1 | +100\*CP |  |  | +1 |  |  | 1x |  |  | 1x |
| **JOS** |  |  |  |  | +2 |  |  | 2x |  | -1x |  |  |
| **SPS** | +2 |  |  |  | +1 |  |  |  | 1x |  |  |  |
| **VOS** |  |  |  | +2 |  | -1 |  |  |  |  |  |  |
| **COS** | -1 | -1 | -2 |  | +1 |  | Aléa suivant monde | 2x |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| **JO2** |  |  |  |  | +2 +2 | +2 +2 |  | 1x |  | 2x |  |  |
| **VO2 (réus.aléa.)** |  |  |  | +0.5\*RP autre  -0.5 RP |  |  | (RP - 2\*gain | 1x | -1x |  | 2x | -2x |
| **SP2** | +2 |  |  |  |  | +1 |  | 1x | 1x | 1x |  |  |
| **DI2** |  | +1 |  |  | +1 | +2 |  | 1x |  | 2x |  |  |

* Le **monde** (société) sera aussi décrit par des règles, par exemple
  + DICT = dictature (très sévère envers les actions mauvaises (grosse prob d'être pris), mais aussi taxe beaucoup les richesses)
  + ANAR = Anarchie (pas de sanctions envers les actions mauvaises (faible prob d'être pris), quasi pas de taxes)
  + DEMS = Démocratie sociale (sévère avec les actions mauvaises, taxes importantes, aide aux plus pauvres)
  + DEML = Démocratie libérale (moyennement sévère avec actions mauvaises, taxes moyennes, aide faible aux pauvres)
  + etc…

1. MCD:

creatures

ID  
position (x,y)  
profile (I/F/R/P/E)  
status (FC, CP,RA, RP, BE, RE)

actions

ID  
activity  
prob  
effect

Society

ID  
type (ex: anarchie)  
sanctions  
taxes  
aides

(fait) n,n

A pour réponse n,n

1. MLD (json):

Ci-dessous un exemple de quelques structures json pour stocker le monde (la société), les créatures et les actions.

*//Le monde (exemple à affiner)*

*//Les aides et les taxes ne jouent que sur la fortune (RA)*

*//L'aide est une somme qui s'ajoute à RA, la taxe est un pourcentage qui doit être multiplié par RA et qui vient en déduction de RA.*

world = {**"name"**:**"anarchie1"**,**"date"**:**"24.01.2023"**,**"niter"**:1130,

"helps":[{0,1000,100},{1000,5000,50},{5000,10000,20}]

"taxes":[{0,10000,0},{10000,30000,0.001},{30000,100000,0.003},{100000,300000,0.006},[300000,1000000,0.012],

"sanctions":{"strength":50,"police":50}//50% de dureté, 50% chance être pris.

*// les créatures (possible de les générer aléatoirement*

*// Profil I=Plaisir immédiat, F=réussite future, Relation, Possession, Ethique  
// Status FC=force (0-100), CP=compétences(0-100), RA=richesse en argent (0-1'000'000), RP=richesse portables(0-100'000), BE=bien-être (0-100), RE=relations (0-100)*

creatures = [

{**"ID"**:0,"name":"Jo",  
 **"pos"** : [{**"x"**:25},{**"y"**:33}],  
**"profile"**: [{**"I"**: 0.15},{**"F"**: 0.15},{**"R"**: 0.15},{**"P"**: 0.15},{**"E"**: 0.15}],  
 **"status"** : [{**"FC"**:50},{**"CP"**:50},{**"RA"**:10},{**"RP"**:8},{**"BE"**:50},{**"RE"**:50}]

},  
 {**"ID"**:1,**"pos"**:[{**"x"**:26},{**"y"**:33}],**"profile"**: [*/\*..\*/*], **"status"**:[*/\*..\*/*]},  
 {**"ID"**:2,**"pos"**:[{**"x"**:40},{**"y"**:48}],**"profile"**: [*/\*..\*/*], **"status"**:[*/\*..\*/*]}  
]

*//les actions: type 1= indiv, type 2= à deux, type=4 avec gagnant ou perdant (suivant quoi ?)*

actions=[

{**"ID"**:**"ETU"**,**"type"**:1,**"prob"**:[0,2,0,0,1],**"effect"**:[0,2,0,0,0,0]},  
{**"ID"**:**"TRA"**,**"type"**:1,**"prob"**:[0,2,0,1,0],**"effect"**:[0,1,CP\*100,0,0,1]},

{**"ID"**:**"JOS"**,**"type"**:1,**"prob"**:[2,0,-1,0,0],**"effect"**:[0,0,0,0,2,0]}

{**"ID"**:**"VOS"**,**"type"**:1,**"prob"**:[2,0,0,0,-2],**"effect"**:[0,0,0,2,0,-1]}

{**"ID"**:**"SPS"**,**"type"**:1,**"prob"**:[1,1,0,0,0],**"effect"**:[2,0,0,0,1,0]}

{**"ID"**:**"JO2"**,**"type"**:2,**"prob"**:[1,0,1,0,0,1,0,1,0,0],**"effect"**:[0,0,0,0,2,2,0,0,0,0,2,2]}

{**"ID"**:**"SP2"**,**"type"**:2,**"prob"**:[1,1,1,0,0,1,1,1,0,0], **"effect"**:[2,0,0,1,0,-1,0,0,0,-1,0,-1]},//ici les deux sont gagnants  
{**"ID"**:**"VO2"**,**"type"**:4,**"prob"**:[2,0,1,0,-1,2,0,1,0,-1], **"effect"**:[0,0,0,1,0,-1,0,0,0,-1,0,-1]}//ici le premier est le gagnant, le second le perdant

1. Diagramme pour le moteur de calcul de l'état suivant:

Déplacements  
(aléatoires ?)

Actions possibles par ind. + duo

creatures.x,.y

actions.conditions

Probabilité de chaque action

creatures.tendance

Tirage action (aléa. pondéré)

Effet de chaque action

creatures.etat

Sanction de chaque action + taxes + aides + salaire + …

société.sanctions

1. Actions possibles pour une créature (indiv) ou un duo :

Exemple si les créatures 2 et 4 sont à proximité et également 0 et 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Indiv. | Creature 0 | Creature 1 | Creature 2 | Creature 3 | Creature 4 |
| Creature 0 | Actions i |  |  |  | Actions duo |  |
| Creature 1 | Actions i |  |  |  |  |  |
| Creature 2 | Actions i |  |  |  |  | Actions duo |
| Creature 3 | Actions i |  |  |  |  |  |
| Creature 4 | Actions i |  |  |  |  |  |

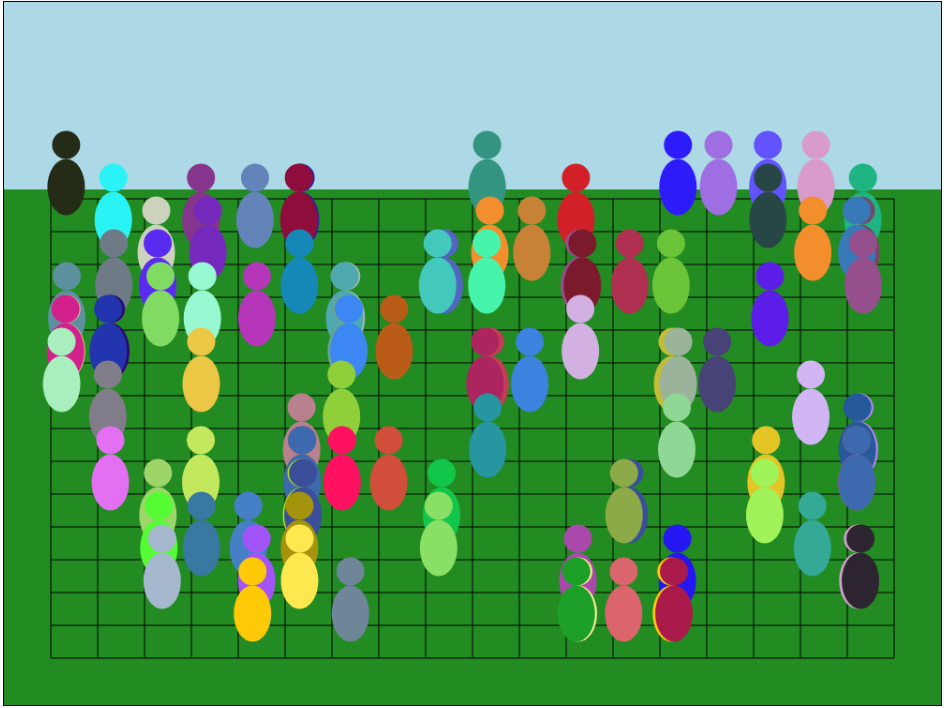
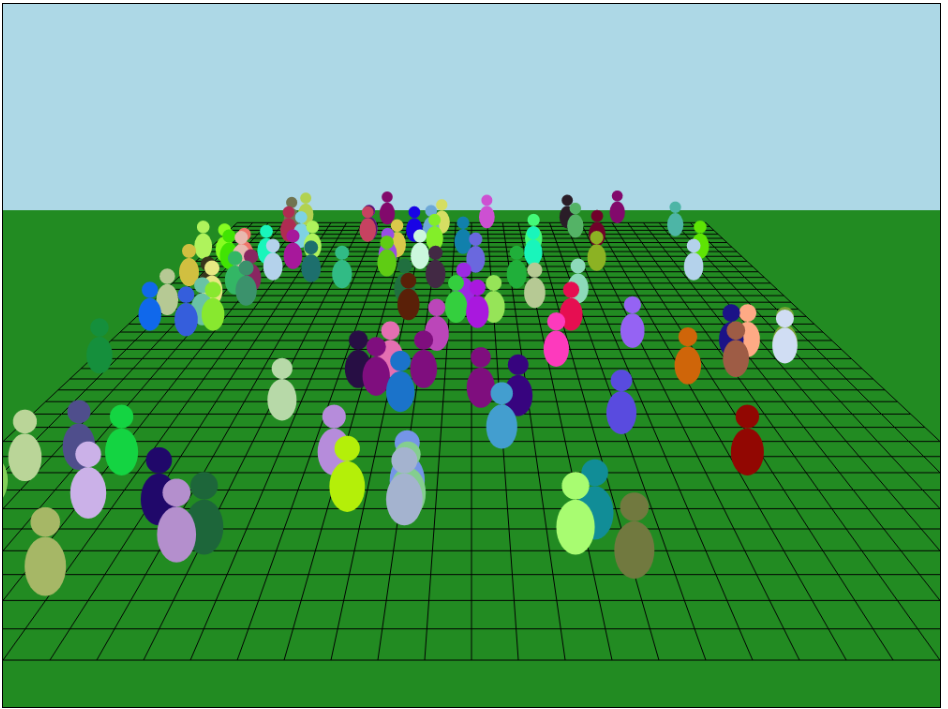
Pour l'interaction entre 2 et 4, on ne traitera qu'une fois (et non pas aussi 4 et 2).

1. Graphismes possibles (à décider) :

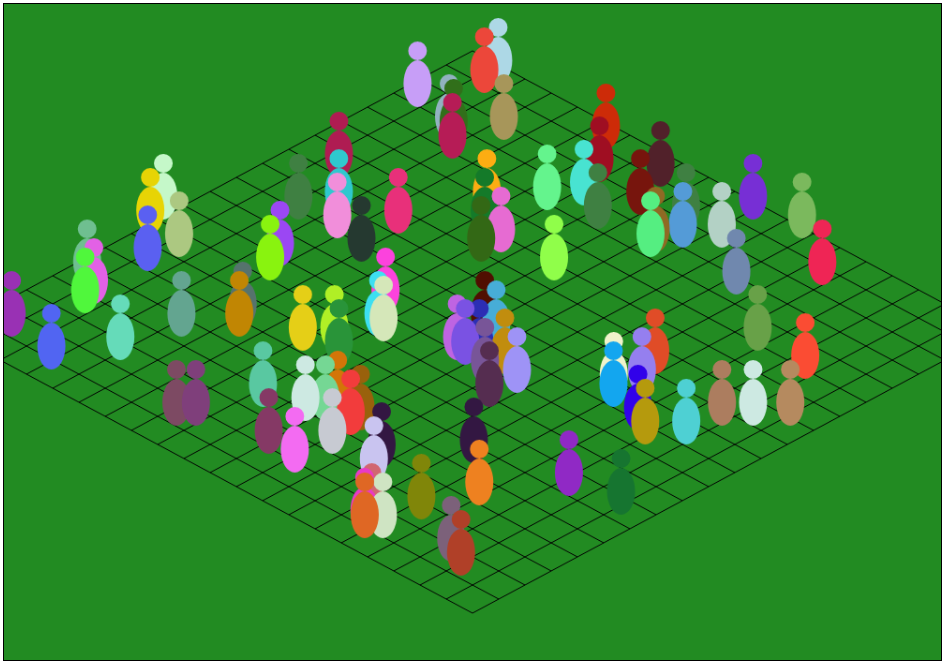
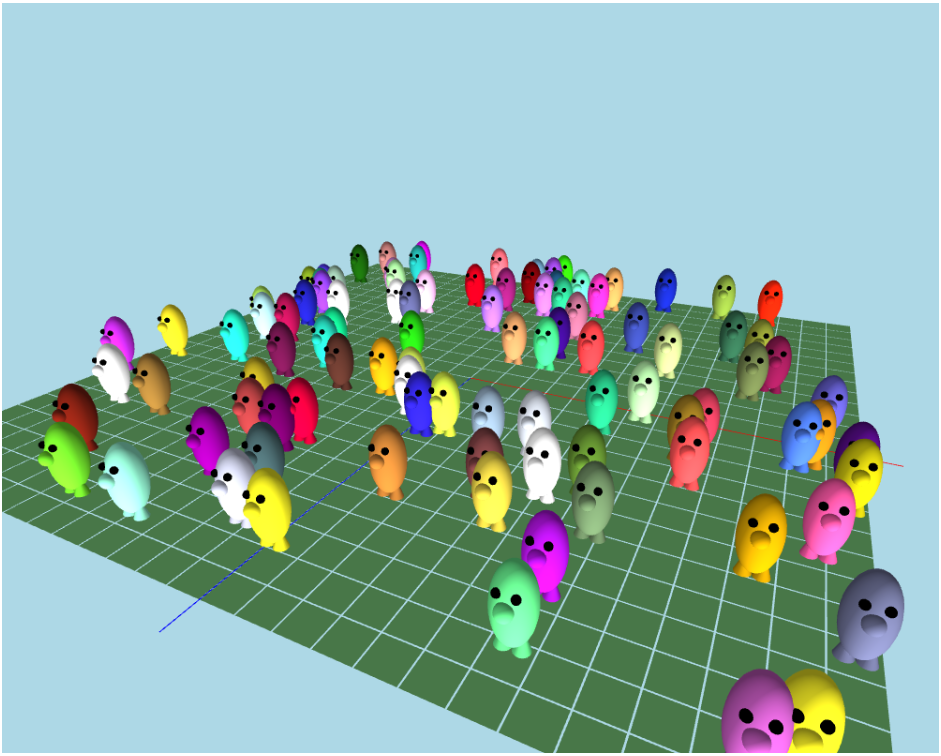
On concevra l'affichage du monde comme indépendant des calculs, on peut donc choisir librement la "perspective". Pour les 3 premières, on n'a pas besoin d'une librairie particulière. Si on prend la représentation 3D, alors il faudra utiliser une petite librairie (ici p5js).

Ci-dessous des "maquettes" avec 100 créatures très simplifiées.

Sans perpective, 2D (ex:14x18): Perpective avec point de fuite (ex:28x40):

Perspective isométrique (ex:20x20): Perspective vraie 3D (ex: 25x25), pilotage /souris.

1. Choix de software:

Le langage (environnement) choisi sera une page html avec des scripts javascript. L'environnement de développement phpstorm est conseillé mais pas obligatoire.

Avantages:

* Facilité de distribuer le soft (par ex. sur un serveur Web).
* Facilité/fluidité pour le graphisme/animation grâce à l'objet canvas.
* Possibilité de passer en 3D facilement (par ex. p5js).
* Compatibilité avec des données en format json.

Limites:

* Difficile de sauvegarder quelque chose (partie en cours) sur le disque local. On pourra travailler avec localStorage.setItem() et localStorage.getItem().

Organisation des fichiers:

* Dans un répertoire images, le groupe "graphisme" déposera tout ce qui sera utile pour la suite, en principe des png pour pouvoir être chargé
* Un fichier world.js décrira le monde de départ sous forme d'un tableau en json
* Un fichier BMWorld.html, chargera les diverses parties (canvas, div…)
* Un timer appellera
  + Un fichier display.js qui offrira une fonction display() qui affichera le monde et les créatures conformément au tableau json
  + Un fichier engine.js qui offrira les fonctions de calcul, par exemple
    - Move(), possible\_actions(), proba\_actions(), action\_choice(), action\_consequences(), retroaction().

1. Dates des sprints
   1. Sprint 1, (v0.1, 3 semaines ouvrables = 30 jan – 23 fév )
   2. Sprint 2: (v0.2 , 2 semaines = 27 fév – 9 mars max)

Le sprint 2 utilisera la représentation créée par l'équipe de graphisme pendant le sprint 1.

On aimerait pouvoir mieux tracer l'évolution d'un individu.

En fonction du sprint 1, on décidera lors du sprint review comment faire évoluer le monde.

* 1. Sprint 3: (v1.0 + 3 semaines = 13 - 30 mars)

(à décider après le sprint 2)

* 1. Réserve, présentation, (3+6 avril)