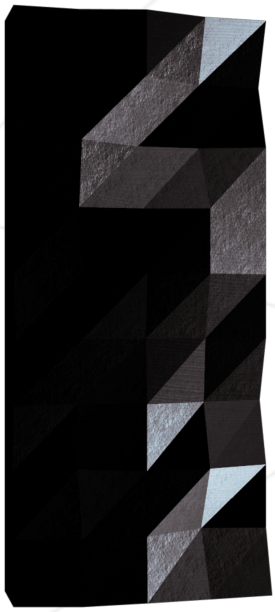

Formation Ruby on Rails - Rush 00

English version coming soon !

Résumé: Ceci est le premier projet complexe (tout est relatif) que vous avez a faire .

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Si aucune information contraire n'est explicitement présente, vous devez assumer les versions de langages suivantes :
  - `Ruby >= 2.3.0`
  - pour `d09 Rails > 5`
  - mais pour tous les autres jours `Rails 4.2.7`
  - `HTML 5`
  - `CSS 3`
- Nous vous interdisons FORMELLEMENT d'utiliser les mots clés `while, for, redo, break, retry, loop et until` dans les codes sources Ruby que vous rendrez. Toute utilisation de ces mots clés est considérée comme triche (et/ou impropre), vous donnant la note de -42.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas, nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépot GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra, ou Slack, ou IRC...

- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Par pitié, par Thor et par Odin ! Réfléchissez nom d'une pipe !

Chapitre II

Règles spécifiques de la journée

- Vous devez faire une application `stateless`
- Vous ne devez pas utiliser de `database`
- Vous ne devez pas utiliser `ActiveRecord`
- Cela revient à dire :
  - Votre classe dans le / les modèle(s) n'heritent de `rien`
  - Vos / votre controlleur(s) heritent (comme par defaut) de ApplicationController
  - Vous avez droit (l'app n'est pas destinée a etre multi-user) à `4 variables globales` : $view, $selected, $game et $player
- Les sauvegardes seront stockées en format JSON valide.

Chapitre III

Préambule

Intro scene from the movie La_Classe_Américaine

V12 — *V12 appelle le capitaine George Abitbol, V12 appelle le capitaine George Abitbol. Quelqu'un vous demande sur le pont.*
George — *Qui ?*
V12 — *Un dénommé José.*
George — *OK, j'arrive, V12.*
José — *Ah, voilà enfin le roi de la classe !*
José — *L'homme trop bien sapé, Abitbol !*
José — *Alors comme ça tu as été élu l'homme le plus classe du monde ! Laisse-moi rire !*
José — *Style le grand play-boy des fonds marins, genre qui fait rêver les ménagères. Sauf que moi je les baise, moi, les ménagères, non ? C'est pas vrai ?*
George — *Écoute-moi bien, mon petit José. Tu baises les ménagères, bien, tu dois avoir le cul qui brille. Mais c'est pas ça qu'on appelle la classe. Je te dis ça en qualité d'homme le plus classe du monde.*
José — *Eh, je t'arrête tout de suite. La classe, c'est d'être chic dans sa manière de s'habiller.*
José — *Rien de tel que d'aller chez Azzedine Alaia... ... ou même de s'acheter des sous-pulls chez Yohji Yamamoto !*
George — *Excuse-moi de te dire ça, mon pauvre José, mais tu confonds un peu tout.*
George — *Tu fais un amalgame entre la coquetterie et la classe. Tu es fou. Tu dépenses tout ton argent dans les habits et... accèssoires de mode... mais tu es ridicule. Enfin si ça te plaît... C'est toi qui les portes. Mais moi, si tu veux mon opinion, ça fait un peu... has been.*
José — *La vache ! Moi, j'ai l'air has been ? J'en ai pour plus d'une barre de fringues sur moi. Alors, va te faire mettre !*
George — *Tu n'es vraiment pas très sympa. Mais le train de tes injures roule sur le rail de mon indifférence. Je préfère partir plutôt que d'entendre ça plutôt que d'être sourd.*
José — *Bien ! Considère qu'on n'est plus amis, Abitbol !*

Chapitre IV

Partie obligatoire

IV.1 Introduction - FAQ

Ce rush à pour objectif de vous faire coder un petit jeu solo doté d'une interface Web.

`Quel est l'objectif de ce jeu ?`
Ce jeu se nomme `Poke..` heuu attendez non, ce jeux s'appelle `MovieMon` et son but est de capturer tous les ... -Movie-mons- qui se cachent sur une grille de jeu en se servant de -Movie-balls-.

`Qu'est ce qu'un Moviemon ?`
Un `Moviemon` est un film disponible sur une base de données comme IMDB ou themoviedb. Idéalement un film de monstre.

`Comment attrape-t-on un Moviemon ?`
En faisant descendre ses points de vie à 0 avant vous. Le rating d'un film d'un `Moviemon` correspondra à sa force. Un rating élevé rend le `Moviemon` plus difficile à attraper qu'un rating bas. La force du joueur, qui correspond au nombre de `Moviemons` en sa possession, augmente ses chances d'en attraper.
`Comment se déroule une nouvelle partie typique ?`
Lorsqu'un joueur démarre une nouvelle partie, le jeu requête sur la base de donnée tous les films nécéssaires avant de l'envoyer sur la 'Worldmap', la page principale du jeu.
Sur cette page, le joueur se déplace librement et allègrement de case en case, sur une grille de taille fixe. Au hasard des cases sur lesquelles il marche, il débusque un `Moviemon`. Deux choix s'offrent alors à lui : s'il estime avoir ses chances, le joueur tente de le capturer. Sinon, il fuit lâchement.
Si son energie descend en dessous de 0 avant le moviemon, celui ci s'échappe et sort du jeu. Si, au contraire, le moviemon est à 0 avant le joueur, alors le moviemon est capturé ! Le joueur peux consulter alors fièrement son `Moviedex` qui répertorie tous les `Moviemons` capturés, avant de repartir en chasse pour tous les attraper ! !

IV.2 Consignes

IV.2.1 Rulez

Votre jeu doit obéir à quelques règles :

- le design du front doit être celui d'une station de jeu video portable grise a écran vert (cf. les screenshots).
- les boutons doivent être des zones cliquable de l'écran principal du layout image-map
- La taille de la grille de jeu doit faire un minimum de dix cases de haut et dix cases de large.
- le jeu doit commencer par un écran titre
- le jeu se déroule sur l'écran de la grille (map) où doit figurer une image de carte et une image de joueur (definissant sa position).
- sur l'écran de jeu, le bouton 'start' va sur le "Movie_dex" et on retourne sur l'écran de jeu via le bouton 'start'.
- si des fonctionalités via des touches sont présentes, une legende doit les indiquer (ex : "Press [A] to skip").
- cliquer sur un bouton autre que ceux définis par la légende ne fait rien (pas meme une erreur).
- les "movies_mons" doivent etre chargés au debut de la partie .
- le "movie_dex" est une liste des "movie_mons" capturés, les boutons droite et gauche feront defiler les pages de cette liste de manière circulaire : aller a gauche depuis le premier item de la liste doit vous emmener à la fin de cette même liste.

Voici des databases de bibliothèques de films : OMDB, qui est une API officieuse d'IMDB, ou The movie database.

IV.2.2 Données de jeu

Vous allez avoir besoin de conserver des données entre les différentes pages lors d'une scéance de jeu. Un site web typique utiliserait des cookies, ou encore un système de sessions coté serveur. Mais ici, il n'est pas question de site web typique.

Vous devrez stocker les données dans des variables globales. Vous en avez quatre a votre disposition 'view', 'game', 'player' et pour les menus 'selected'

Vous devez donc également créer dans votre projet, la logique nécessaire à la mise à jour et à l'utilisation de ce fichier qui doit contenir les informations suivantes :

- La position du joueur sur la map.
- Les noms (ou identifiants) de tous les `Moviemons` du `Moviedex`.
- Les informations complètes de tous les `Moviemons` de la partie, tels qu'obtenus sur la base de données de votre choix.

IV.2.3 Gestion des données

Vous devez également créer une (ou plusieurs) classe(s) qui a pour mission de gérer ces données de jeu. Cette(Ces) classe(s) doi(ven)t à minima contenir les méthodes suivantes :

- 'initialize'
- 'save' : Ecrit les données de jeu dans un JSON valide .
- 'load' : Lit les données de jeu depuis un fichier et les utilise pour assigner ces données aux variables du jeu .
- 'get_movie' : Retourne un array ou hash contenant tous les détails du nom de `Moviemon` passé en paramètres et nécessaires à la page `Detail`.

Vous pouvez ajouter à vo(tre)s classe(s) autant de méthodes et d'attributs que nécessaire.

Sur l'écran titre, le bouton 'select" affiche les celèbres 3 'save slots' et de charger une sauvegarde. Sur l'ecran de jeu (la map), le bouton 'select" affiche aussi les trois 'save slots' et on peux y sauvegarder la partie en cours. Sur ces deux ecrans de 'save slots', on navigue avec les fleches haut et bas .

IV.2.4 Esthétique du jeu

L'affichage du jeu se fera naturellement sur votre navigateur via du `HTML` et du `CSS`. Vous n'avez pas la permission d'utiliser de `Javascript`.

L'affichage du jeu est scindé en deux parties qui doivent être visuellement parfaitement distinguable :

- `L'écran` : Affiche ce qui se passe dans le jeu. Aucune interaction n'est possible à cet endroit qui ne doit jamais contenir ni lien ni aucun formulaire.

- `Les contrôles` : Situés en dessous ou de part et d'autre de l'écran, ils permettent d'interragir avec le jeu et sont contextuels, c'est à dire qu'ils changent de comportement en fonction de l'état du jeu. Cela signifie également qu'ils ne sont pas nécessairement tous actifs systématiquement. En revanche, même inactifs, ils doivent être `visibles et garder la même place en permanence`.

  Il doit y avoir neuf 'boutons' :

  - Quatre `directions`, comme celles qu'on pourrait trouver sur une une croix directionnelle de manette de jeu :
    Haut, droite, bas, gauche
  - Un bouton `select`.
  - Un bouton `start`.
  - Un bouton `Power` (la petite led rouge)
  - Un bouton `A`
  - Un bouton `B`

Vous n'avez pas la permission de rajouter/supprimer de 'boutons' ni d'afficher d'informations, en dehors du nom des 'boutons', dans cette zone.
Au delà de cette distinction minimum, l'esthétique n'a pas d'importance dans la partie obligatoire du sujet.
Le comportement des boutons pour chaque vue est décrit dans la section suivante.

IV.2.5 Les pages

Vous devez créer les pages/vues/comportements listées si après. Un 'bouton' non mentionné dans une page est un 'bouton' inactif. De plus, si la destination n'est pas précisé pour un contrôle, c'est que celui-ci renvoye la même page, potentiellement modifiée.

`TitleScreen`

- Description : Ecran d'accueil
- Ecran : Doit afficher le nom du jeu ainsi que 'Start - New Game' et 'Select - Load'.
- Contrôles :
  - `Start` accès au `Jeu`
  - `Select` accès aux `Save slots`
  - `Power` "reboot" le jeu, reset les variables globales et retourne au `Title screen`.

`Worldmap`

- Description : carte du jeu, où le personnage se déplace, et débusque des `Moviemons`.
- Ecran : Une grille dont la taille est celle définie dans les settings. Sur la case correspondant à la position actuelle du joueur doit se trouver une représentation (image, caractère, etc ...) clairement identifiable du personnage.
- Contrôles :
  - `Directions` : Chaque direction doit déplacer le personnage d'une case dans la même direction. Le joueur ne doit pas pouvoir sortir de la map.
    Chaque déplacement a une chance de débusquer un `Moviemon`
  - `start` : accès au `Moviedex`.
  - `select` : accès aux `Save slots`.

`Rafraîchir cette page ne doit pas modifier la position du personnage sur la carte.`

`Battle`

- Ecran : Affiche le poster le nom du `Moviemon`, le nom du directeur, son energie et la votre
  En cas de capture, vous devez également afficher une phrase du type "You caught it" pour marquer l'evenement.
  En cas de fuite, vous devez également afficher une phrase du type "You coward !!" .

- Contrôles :
  - `A` : Hit `moviemon`
    le joueur ici lance un hit et soustrait son hit_point à l'énergie du moviemon, qui le hit aussi à son tour avec son hit_point à lui qui est son 'Rating'.
    En cas de succès (l'énergie du moviemon arrive à 0 avant la votre), le `Moviemon` est capturé puis stocké dans le `MovieDex`. Vous afficherez un message adéquat, et remettez l'énergie du joueur au max, mais aussi augmenter son hit point (et oui, l'experience ça paie).
    En cas d'échec, vous afficherez un message adequat, et vous supprimez le moviemon de la liste disponible et vous redirigerez le joueur sur la map.

  - `B` : Fuite vers `Worldmap`
    Dans ce cas le moviemon est remis en liberté, l'énergie du joueur est restaurée et on retourne sur la map .

`Moviedex`

- Ecran : Un `Moviemon` capturé avec ses infos :
  Année, Genre, Directeur, Rating, Synopsis ET Poster

- Contrôles :
  - `Droite et Gauche` : Les directions permettent de sélectionner un film différent. Vous devez utiliser au moins deux directions : gauche et droite .

  - `start` : Retour vers la page `Worldmap`

`Par défaut en arrivant sur la page, le premier film de la liste est sélectionné.`

`Save slot`

- Description : Permet de faire des sauvegardes du jeu dans le fichier 'save.json' ou depuis le title screen de 'load' une sauvegarde .
- Ecran : Affiche les trois slots de sauvegarde / chargement
- Contrôles :
  - `Haut et Bas` : Selectionne le slot à manipuler
  - `Select` : retour à l'ecran precedant
  - le reste : `AD LIB`

Chapitre V

Partie Bonus

Une fois votre partie obligatoire parfaitement réalisée, vous pouvez implémenter des fonctionnalités supplémentaires afin de gagner des points bonus.

Pour que votre correcteur considère ces fonctionnalités comme réussies, vous devrez le convaincre de leur bonne réalisation.
Une erreur non gérée invalidera la fonctionnalité en question.

Uniquement dans le cadre des bonus, l'utilisation de Javascript est tolérée tant que celle-ci n'altère pas le fonctionnement de la partie obligatoire (qui doit fonctionner sans javascript). L'AJAX ou les Websockets ne sont pas autorisés.

Voici quelques idées de bonus que vous pourriez implémenter :

- Associez les contrôles à des touches du clavier afin de ne pas avoir à utiliser la souris pour jouer.
- Afin de varier le jeu, faites que chaque nouvelle partie charge une selection différente de monstres.
- Ajoutez de la variété à la `Worldmap` avec elements infranchissables.

Chapitre VI

Rendu et peer-évaluation

Vous devez rendre un projet `Rails` parfaitement configuré.

Hormis de ce qui vous est imposé dans le sujet, vous êtes libre d'organiser ce projet comme vous l'entendez.

Aucune erreur serveur ne sera toléré. Testez bien les comportements de vos vues.

Vous devez fournir un fichier requirement.txt contenant toutes les librairies nécessaires au fonctionnement de votre projet.

`Vos modeles ne doivent hériter de rien, vous pouvez avoir des migrations générées par rails mais rien dans la db`

Verification facile avec la commnade suivante executee dans la console :

```
ActiveRecord::Base.subclasses.map { |forbid| forbid.name }
```

Chapitre VII

Exemple de rendu

FIGURE VII.1 – Your titlescreen could look like that

FIGURE VII.2 – This Moviemon game appears to happen in A familiar map

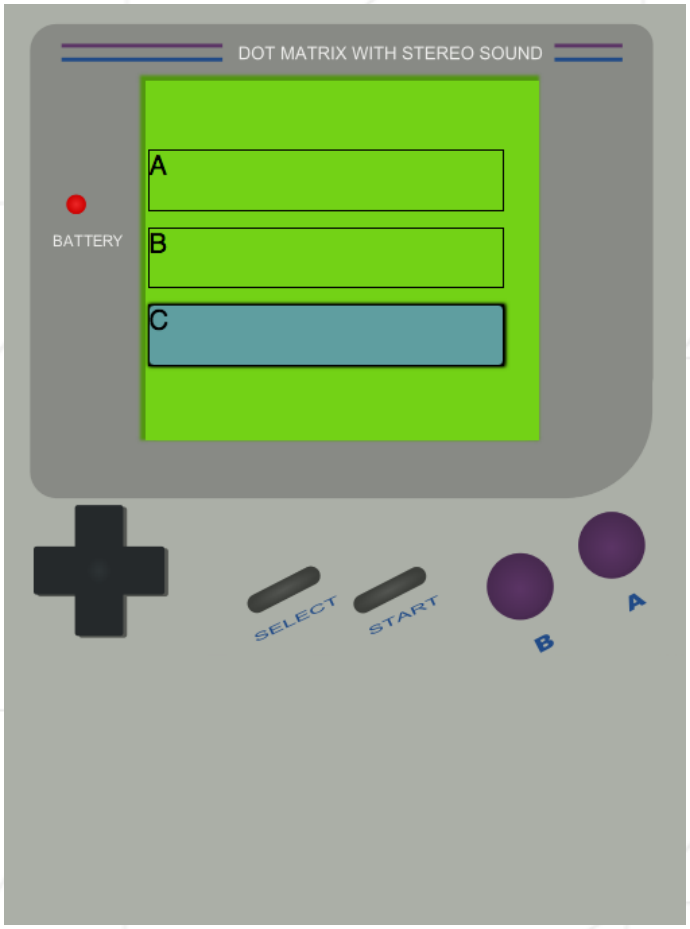

FIGURE VII.3 – Save slot with 'c'selected

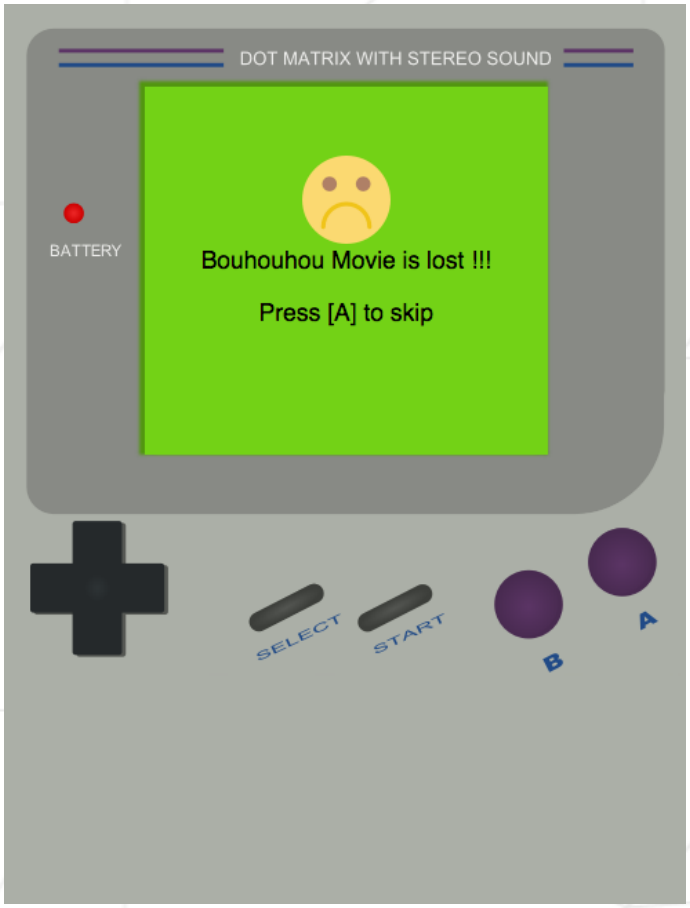

FIGURE VII.4 – A lost combat message

FIGURE VII.5 – A fight screen

FIGURE VII.6 – Your moviedex could look like that