Université des Antilles
UFR des Sciences Exactes et Naturelles
Master 2 - MIAGE

PROJET
CRAZY M2

Notre objectif sera de développer **Crazy M2,** un jeu de course de voitures en réseau, basé sur des services WEB. L'application globale se compose de deux éléments principaux :

- **Un service WEB de type REST**, qui calcule à chaque requête la position des voitures en fonction des touches saisies par l'utilisateur
- **Un client,** qui sollicite le serveur pour connaitre sa position et celle de ses concurrents sur le circuit, et qui affiche le résultat

Le fonctionnement de notre application peut-être schématisé ainsi :

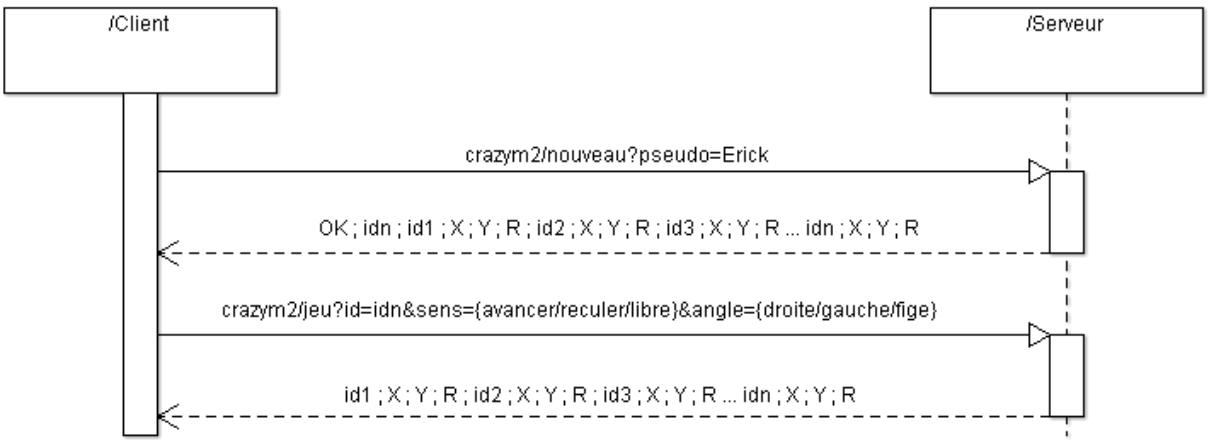

Figure 1. Fonctionnement de l'application

Ce fonctionnement présente deux avantages :

1. Tous les calculs physiques liés aux déplacements sont effectués sur le serveur via un service WEB. Le client ne fait qu'envoyer au serveur les touches sur lesquelles appuie l'utilisateur. Il est ainsi possible de rajouter de nouvelles fonctionnalités au jeu, tels que des dérapages, des bonus ou des collisions, sans avoir à modifier le client.

2. Le client peut-être développé dans n'importe quel langage et sur n'importe quelle plateforme. Il serait d'ailleurs intéressant que chacun développe "*sa version*" du client (C, JavaScript, PHP, ActionScript, une version IPhone, une version Android, etc.).

LE SERVICE WEB REST

Exercice 0 : Discussion

Discussion sur le principe de fonctionnement de l'application

Exercice 1 : La classe Voiture

Ecrivez la classe **Voiture** permettant de gérer une voiture sachant qu'à un instant t, une voiture est définie par :

- Sa position sur X et Y
- Sa vitesse
- Son angle de rotation

Cette classe doit contenir les méthodes fondamentales permettant, à chaque instant du jeu, à une voiture de:

- Tourner à gauche
- Tourner à droite
- Avancer
- Reculer
- Décélérer

Les contraintes sont les suivantes :

1. La vitesse du véhicule doit varier en fonction du temps, mais ne doit pas dépasser une certaine limite **VITESSE_MAX**
2. Le véhicule ne change pas brutalement de direction mais tourne progressivement selon un angle **ANGLE_ROTATION**
3. Si l'utilisateur n'appuie sur aucune touche, le véhicule décélère selon un facteur **DECELERATION** jusqu'à atteindre une vitesse nulle et se stabilise
4. Si le véhicule est déjà en mouvement (avance ou recule), le véhicule ne peut pas reculer ou avancer brutalement, il doit d'abord décélérer progressivement selon un facteur **FREIN** jusqu'à atteindre une vitesse nulle

On se limitera à ces contraintes pour commencer, mais d'autres effets peuvent être ajoutés au cours du développement (dérapage, ralentissement selon la zone, collisions, etc.).

Exercice 2 : La classe Joueur

Ecrivez la classe **Joueur** permettant de stocker un joueur.
Un joueur est défini par :

- Un ID
- Un pseudonyme
- Sa voiture

Dans un premier temps, nous conserverons uniquement ces informations. Toutefois, des informations additionnelles pourront également être ajoutées par la suite, telles que le score, l'heure de connexion au jeu, le temps au parcours, etc.

Exercice 3 : La classe ListeDeJoueurs

Ecrivez la classe **ListeDeJoueurs** permettant de gérer une liste de joueurs (ArrayList<Joueur>).
Cette classe doit naturellement comporter les méthodes fondamentales permettant:

- L'ajout d'un nouveau joueur
- La recherche d'un joueur en fonction de son ID
- La suppression d'un joueur
- Le renvoie d'une chaine de caractères contenant la position de toutes les voitures stockées
- Le test pour savoir si un joueur est déjà présent

Vous ajouterez toutes les fonctions qui vous semblent utiles pour la gestion d'une liste de joeur.

Exercice 4 : Le service WEB REST

Ecrire le service permettant de traiter les différentes requêtes décrites sur la Figure 1. Ce service se chargera de gérer la liste de joueurs (ajout, mise à jour, etc.).

TOUTES les requêtes concernant le jeu seront traitées par une même Servlet !

Comme le montre la Figure 1, le service doit permettre d'effectuer au moins trois opérations:

- **GET crazym2/nouveau**
  Permet de s'inscrire en tant que nouveau joueur
  La requête doit être accompagnée du paramètre *pseudo* pour être validée.
  Ex. crazym2/nouveau?pseudo=Toto
  Le serveur renvoie au client son identifiant, ses coordonnées initiales et son angle de rotation initial dans un format au choix: plain text, html, xml, yaml, json ???
  Les coordonnées des concurrents sont également renvoyées.
  (Voir exemple de réponse du serveur sur Figure 1).

- **GET crazym2/jeu**
  Une fois inscrit, permet au client d'envoyer au serveur les touches enfoncées par l'utilisateur et de recevoir la mise à jour des coordonnées.
  La requête doit être accompagnée des paramètres
  - id (reçu lors de l'inscription),
  - sens = {avancer, reculer, libre) : correspond respectivement a un appui sur touche haut, bas et rien
  - angle = {droite, gauche, fige} : correspond respectivement a un appui sur touche droite, gauche, et rien

  Ex. crazym2/jeu?id=1&sens=avancer&angle=gauche
  Signifie que le joueur 1 appuie sur la touche avancé en tournant à gauche.
  Ex. crazym2/jeu?id=3&sens=libre&angle=droite
  Signifie que le joueur 3 tourne à droite, sans demander d'accélération ou de freinage

  Le serveur reçoit ainsi la requête, la traite, puis retourne au client sa nouvelle position et celle des concurrents. Naturellement, tous les contrôles d'erreur seront effectués. Par exemple, un utilisateur ne peut pas jouer s'il ne s'est pas enregistré ou si les paramètres requis ne sont pas renseignés.

- **GET crazym2/who**
  Permet d'afficher la liste des joueurs connectés au service. Le résultat sera renvoyé sous forme de page HTML.

Exercice 5 : Tests et Validation

Tester le fonctionnement global de votre service à l'aide de votre navigateur (ou d'un plugin) en saisissant, par exemple, les URLs

- crazym2/nouveau?pseudo=Toto
- crazym2/jeu?id=1,sens=avancer,angle=gauche
- crazym2/jeu?id=1,sens=rien,angle=rien
- etc.

Exercice 6 : Implémenter le client

Implémenter le client graphique, dans le langage de votre choix (C, JAVA, HTML, Javascript, etc.), faisant appel à votre service WEB.

Exercice 7 : Compléter le client

Maintenant que vous avez le squelette de votre application et de vos échanges client/serveur.
Faites évoluer votre application pour y intégrer une map, des scores, des éléments de décor qui impactent la vitesse des véhicules, des collisions, des bonus, etc. etc.