AJAX - Projet 2020

D.Moreaux

13 décembre 2020

Table des matières

1 Vue d'ensemble

Le projet permettra de se déplacer dans un labyrinthe en pseudo 3D. Le labyrinthe sera affiché en empilant des images contenant les différentes sections de mur, les images en avant plan cachant celles en arrière plan.

Les déplacements se feront case par case et par rotations de 90 degrés. Le labyrinthe sera prévu sur plusieurs étages.

Le programme permettra que plusieurs joueurs soient présents simultanément et qu'ils puissent communiquer.

1.1 Système d'affichage

La personne verra les cases situées devant lui et sur les côtés. Une image de fond contient le sol et le plafond et 26 images permettront d'afficher les murs qui font face aux joueurs et ceux qui sont de côté.

Les fichiers sont nommés en fonction du thème (BLUE, BRICK, DROW, GREEN et XANATHA), de la position (de A à Q), de l'orientation (F pour face au joueur, S pour de côté) et du type de mur (normal, escalier,...)

A	B	C	D	E	F	G
	H	I	J	K	L	
		M	N	O		
		P	↑	Q		

La flèche indique la position du joueur et la direction dans laquelle il regarde.

Les positions N, J et D correspondent aux cases en face du joueur. Ces positions ne proposent donc que des murs de face (F)

Les positions P et Q sont sur les côté du joueur et ne proposent que les images de côté.

Pour les positions extrèmes (A, G, H et L), les images de face n'existent pas.

Les autres positions ont les deux fichiers disponibles

Il y a donc 17 positions et 26 images.

Les images font toutes 320x200 et devront être mises à l'échelle (x2 ou x3 selon la mise en page). L'empilement part de l'image de fond (BACK), commence par les images de côté de la ligne la plus éloignée (A-G), continue avec les images de face de la même ligne (B-F), puis les images de côté de la ligne intermédiaire (H-L), les images de face (I-K), les images de la première ligne (côté puis face M-O) et enfin le deux images sur les côté du joueur (P et Q).

1.2 L'interface de jeu

6 boutons sont prévenus pour être mis en deux lignes de 3 : tourner à gauche, avancer, tourner à droite et glisser à gauche, reculer, glisser à droite.

Des images permettent d'afficher une boussole indiquant la direction dans laquelle on regarde. Elles sont nommées compass-E, compass-N, compass-W et compass-S.

Ces images pourront être affichées sur l'écran de jeu ou à côté. A noter que si un canvas est utilisé, il faudra déterminer la position de la souris dans le canvas pour déterminer sur quel flèche on a cliqué.

Lorsque l'on charge le programme, une interface doit permettre de se connecter ou de créer un compte. La création de compte demandera un identifiant, une adresse E-Mail et un mot de passe.

Une zone devra être prévue dans l'écran de jeu pour l'interface de discussion. Cette zone doit proposer une zone où afficher les messages et une zone d'entrée pour entrer son propre message. Plusieurs types de messages seront prévus : chuchoter, parler et crier. La différence sera la distance à laquelle le message sera entendu (1 case, 3 cases ou tout l'étage).

1.3 Les cartes

Les cartes auront une taille fixe et seront composées de valeurs de 0 à 6. Une valeur de 0 indique un passage et une valeur de 1 à 5 correspondra à un "mur".

Afin de ne pas nécessiter de gérer les 4 côtés d'une case, les types 3 à 6 (passage type porte ouverte, escalier ou portail) ne seront visibles que d'un seul côté (encadré de murs). Les types 1 et 2 correspondent à un mur plein, deux images sont présentes pour casser la monotonie des murs.

Les escaliers (type 4 et 5) permettront d'accéder à l'étage supérieur (numéro d'étage +1) ou inférieur (numéro d'étage -1).

Les passages ouverts (type 3) ne bloqueront pas les déplacements. Pour éviter les problèmes d'affichage, quand on franchira un passage, on avancera automatiquement de deux cases.

Les cartes et la position du joueur ne seront connus que du serveur. Le client ne pourra a aucun moment connaître les coordonnées du joueur (étage, X, Y) ni les cartes complètes.

Les cartes seront fournies sous la forme d'un fichier PHP à inclure et seront des tableaux à 3 dimensions : étage,x,y.Le fichier est généré à partir d'un fichier qui reprend ces cartes dans un format textuel. A côté de cela, le thème de la carte sera sauvé dans la DB dans une table qui reprendra le numéro de l'étage et le nom du thème.

Pour les escaliers, quand on avance sur un escalier, on est téléporté devant l'escalier de l'étage suivant (le déplacement en avant n'est pas effectué), en étant "retourné" de 180° (l'escalier de retour dans son dos). Si on essaie de reculer ou de faire un déplacement latéral vers l'escalier, ce sera refusé comme si c'était un mur.

1.4 Le programme

Si il est possible d'utiliser des divisions pour réaliser l'affichage du labyrinthe, on préférera néanmoins l'utilisation d'un canvas qui permettra de redimensionner les images au vol.

Le programme devra être composé d'un fichier HTML unique (pas de liens, actions de formulaires, iframe ou redirections à partir du Javascript), de un ou plusieurs fichiers javascript et d'un ensemble de fichiers PHP qui ne retourneront pas de contenu HTML (texte pur, CSV ou XML).

Les fichiers PHP devront être impérativement documentés dans un fichier texte. Cette documentation devra décrire l'utilité et l'interface de ces fichiers (type de requêtes, données en entrée et en sortie) mais pas les algoritmes utilisés en interne.

Les différentes ressources devront être réparties dans des sous-répertoires : CSS, JS, Images, points d'accès PHP.

2 Etape 1 : l'affichage de base

Le programme sera composé de 3 *écrans*. L'écran de connexion, l'écran de création de compte et l'écran de jeu [1].

Chaque écran sera une division dans le fichier HTML. L'écran de jeu contiendra de plus un canvas qui permettra d'afficher le labyrinthe.

2.1 Squelette HTML

La première étape consistera donc à réaliser ces trois écrans ainsi que les CSS qui permettent de contrôler leur apparence. A chaque fois, une des divisions sera visible et les deux autres cachées (à l'aide de `display: none;` au niveau CSS).

Pour le canvas, il peut potentiellement être remplacé par une balise IMG qui affiche une des images `XXX.BACK.png` mais si vous le faites, assurez vous

1. rien ne vous interdit d'ajouter des fonctionnalités qui demandent des écrans supplémentaires mais les fonctionnalités exigées doivent se faire sur ces trois écrans

de supprimer bordures et autres pour éviter un espacement parasite [2].

2.2 Javascript

Une fois cette partie correcte, prévoir au niveau javascript les fonctions suivantes :

- Une fonction pour chaque écran qui switche sur l'écran en question (vous serez amenés à y ajouter du code plus tard)
- Des fonctions qui géreront les boutons login, nouveau compte, créer compte, annuler et logout. Pour le moment, ces fonctions se contenteront d'appeller les fonctions ci-dessus.
- Une fonction qui prend en paramètre un nom de thème et qui charge les images correspondant au thème dans des tableaux globaux [3].
- Une fonction qui prend un tableau d'entiers contenant les cases en face du joueur (au choix, un tableau à une dimension qui contiendra les 17 cases d'intérêt ou un tableau de 7x4 dont les cases d'intérêt vaudront les valeurs de 0 à 6 correspondant au type de *mur*) et qui affichera dans le canvas l'image de fond puis les images de murs correspondant aux données du tableau.
- Une fonction de test qui sera lancée quand on accèdera à l'écran de jeu et qui appellera la fonction précédente avec un tableau *fixe*
- une fonction d'initialisation qui mettra en place les fonctions de gestion de bouton et lancera le chargement d'un thème de votre choix (peut-être éviter XANATHA)

3 Etape 2 : affichage du labyrinthe

3.1 Ping pong avec le serveur

Dans un premier temps, les fonctions d'affichage et de déplacement renverront une chaîne contenant la position du joueur. Aucun test de mur ou d'escalier ne sera mis en place.

Lors de l'appui sur le bouton de connexion, on enverra une requête sur le serveur qui permettra de mettre une position de départ dans la session. Cette fonction retournera une valeur indiquant un login correct [4]. A la réception de

2. vous pouvez également utiliser une DIV avec une image de fond et les dimensions voulues mais dans ce cas, l'image ne sera pas mise à l'échelle
3. déclarés en dehors de fonctions
4. vous pouvez choisir une valeur numérique ou texte

cette valeur, vous appellerez la fonction qui bascule sur l'affichage de l'écran de jeu.

Les 6 boutons directionnels enverront une requête au serveur qui indiquera le déplacement. Le serveur mettra les coordonnées à jour dans la session. Un appel sera alors fait à la fonction chargée de retourner la portion de carte visible qui retournera une chaîne avec position et direction. Vous afficherez cette chaîne dans la zone destinée aux messages de chat.

Niveau serveur, vous vous assurerez durant les déplacements que les coordonnées restent dans les limites de la carte courante (niveau 1).

3.2 Avec les images

Dans un second temps, la routine qui retourne la portion visible de la carte devra envoyer le tableau des positions visibles. Cela sera fait sous la forme d'une chaîne texte et peut être fait de diverses manières. La fonction qui traitera le retour devra générer le tableau de 17 ou 7x4 éléments et appeler la fonction de mise à jour du canvas.

3.3 Et les murs

Dans un troisième temps, on ajoutera la gestion des murs au niveau de la fonction de déplacement. On peut commencer en gérant toute valeur non-nulle comme un obstacle pour ensuite gérer correctement portes et escaliers.

4 Etape 3 : Authentification

On ajoutera la création d'un compte et l'authentification.

Pour créer un compte, la valeur de retour devra signaler s'il y a une erreur : compte déjà existant, identifiant invalide, mot de passe vide,...

Pour l'authentification, on récupérera dans la DB la position du joueur. Cette position devra être mise à jour dans la DB (en plus de la session) à chaque mouvement.

5 Etape 4 : Gestion complète des déplacements

Jusqu'à maintenant, le thème était fixe. On ajoutera une fonction qui chargera un nouveau thème et appellera une fonction une fois le thème chargé. Pour cela, il faudra suivre les `onload` des différentes images chargées et appeler la fonction une fois le dernier exécuté.

Lors d'un déplacement, on retournera une valeur indiquant :

— si le déplacement a pu être fait
— si on a changé d'étage (ainsi que le nouveau thème)
— la direction dans laquelle on regarde

Ces informations permettront d'afficher un message, jouer un son ou autre, de mettre à jour la boussole et éventuellement de charger un nouveau thème si ce dernier a changé.

6 Etape 5 : Le chat

On implémentera le chat. Lorsqu'un message est envoyé, au niveau serveur, on vérifiera quels sont les destinataires et on ajoutera le message pour ces derniers.

Pour afficher les messages en temps réel, on utilisera une fonction appelée toutes les 2-3 secondes qui enverra le dernier timestamp reçu et récupérera les messages reçus depuis avec leur timestamp. La liste de message peut être vide. Les nouveaux messages seront ajoutés dans la fenêtre de texte de chat. En absence de timestamp, on enverra un nombre limité de messages.

Pour sélectionner la cible, on peut soit utiliser des contrôles HTML (radio, select), soit utiliser des commandes comme /w (whisper), /s (say), /y (yell) avant le message (choisir une commande *par défaut* quand qucune commande n'est utilisée.

7 Etape 6 : Les émoticons

On ajoutera un bouton qui permettra de choisir des émoticons dans une grille. Chaque icone dans la grille insérera un code textuel dans la zone d'entrée (en fin de texte). Utiliser une image "grille d'icones" et des closures pour implémenter cela.

Dans l'affichage des messages, remplacer les codes par les icones correspondantes.

Les images font 24x24 et sont dans une grille de 38x38 avec un offset de 8 pixel dans les deux directions. Les codes textuels correspondant sont repris dans le fichier texte emoticons.txt.