

HAndroid

Projet informatique

Réalisé par : Mathilde BRUN, Alexis FRADIN, Victoria HOPMANN, Amine RFIG et Anouar MABROUKI

2^{ème} année Eurinsa, 2014/2015

I. Mise en contexte :

- La plupart des applications Android proposées sur le Web utilise le son et les vibrations comme éléments secondaires. De plus, l'interaction entre l'utilisateur et le jeu se base essentiellement sur les images proposées. Mais la technologie des téléphones actuels donne accès à plus de fonctionnalités que celles généralement exploitées. Ainsi, nous souhaitons créer un jeu Android employant le son et les vibrations comme informations vers l'utilisateur, en leur donnant la même importance que les images. Le challenge serait en partie d'utiliser certains sens d'une manière plus poussée que d'ordinaire.

II. Objectifs généraux :

- Créer une application Android pouvant se jouer à l'aveugle, ou tout du moins avec peu de visibilité et ainsi permettre au joueur de se projeter dans la situation d'une personne malvoyante. Le challenge serait en partie d'utiliser ses sens d'une manière inhabituelle et ainsi d'acquérir des compétences qu'il n'a pas à l'origine, tel un aveugle qui doit optimiser son utilisation des autres sens (ici principalement l'ouïe et le touché). Tout ceci se déroule dans le cadre d'un jeu interactif.
- Utilisation des technologies récentes proposées par les smartphones (accéléromètre, vibration, son stéréo) et ainsi se détacher de l'utilisation conventionnelle de l'écran passant par l'image et les touches. L'application amènerait les utilisateurs à faire face à de nouveaux défis, ces éléments étant pour certains encore peu mis en valeur dans les applications actuelles.

III. Description fonctionnelle :

Fonction principale : Jeu basé sur des déplacements dans un espace virtuel.	
Description	<p>L'utilisateur est modélisé par un personnage se trouvant dans un parcours (de difficulté variable), modélisé par une map virtuelle, dans lequel son personnage a un objectif à atteindre. Les déplacements seront gérés via l'accéléromètre. L'utilisateur est guidé par d'autres moyens que l'image : les sons et les vibrations du téléphone lui indiquent les obstacles proches ou à distance.</p> <p>La relation entre le personnage et l'objectif à atteindre pourra modéliser des besoins quotidiens (lieux à atteindre dans un certain délai, nourriture à chercher ...). Le jeu est gagné si le joueur réussit à atteindre son objectif.</p>
Contraintes	<p>Le jeu sera réalisé en 2 dimensions, et devra pourtant présenter un environnement expressif.</p> <p>Le personnage pourra bouger en fonction de l'inclinaison de l'appareil. Le joueur devra donc maîtriser l'orientation du personnage grâce à l'accéléromètre.</p> <p>Une gestion du son en stéréo pourra être envisagée afin de mettre en relief le monde auditif de l'utilisateur (usage d'un casque nécessaire).</p>
Niveau de priorité	Priorité supérieure

Fonction 1 : Menu proposant plusieurs options.	
Descriptions	<p>Après avoir lancé son application, l'utilisateur a accès à un menu lui présentant les choix suivants :</p> <p>1. Lancer une partie</p> <p>L'utilisateur pourra alors créer un profil en rentrant son identifiant et en choisissant un personnage (défini par une couleur notamment) ou reprendre le personnage correspondant à son profil si celui-ci est déjà rentré dans l'application. Un second menu lui est ensuite proposé, dans lequel il peut décider de commencer une nouvelle partie ou de reprendre une partie sauvegardée. Le lancement d'une nouvelle partie implique que l'utilisateur choisisse le niveau de difficulté avec lequel il</p>

	<p>veut jouer, parmi ceux qu'il a déjà débloqués. Si le joueur décide de démarrer une nouvelle partie alors qu'une autre partie avait été sauvegardée à son nom, la sauvegarde de sa nouvelle partie remplacera la précédente sauvegarde.</p> <p>2. Historique des meilleurs scores: Les meilleurs scores sont présentés selon un classement global, présentant les meilleurs scores pour chaque niveau, tous joueurs confondus.</p> <p>3. Visualiser son profil L'utilisateur peut également visualiser ses scores pour chaque niveau déjà joué et modifier les paramètres de son profil.</p> <p>4. Quitter l'application</p>
Contraintes	La gestion des données sauvegardées
Niveau de priorité	Priorité supérieure.

Fonction 2 : Création d'une map avec obstacles empêchant l'atteinte de l'objectif.	
Description	Le plan de la map sera doté de certains obstacles de types variés (modélisation de murs, de trous, de meubles, d'escaliers...). La distribution et le nombre d'obstacles dans l'espace dépendra du niveau de difficulté choisi auparavant.
Contraintes	Créer des obstacles expressifs en 2D.
Niveau de priorité	Priorité supérieure

Fonction 3 : Annonce de l'approche ou du heurt d'un obstacle grâce aux effets sonores et aux vibrations	
Descriptions	Le jeu pouvant être joué à l'aveugle, l'approche d'un ou plusieurs obstacles est signifiée par des émissions de sons, modulées en intensité selon la distance à l'obstacle, en présence des écouteurs, dans l'oreille droite et/ou gauche selon la position du ou des obstacles à proximité du personnage dans la map. La nature de ces sons dépendra de la nature de l'obstacle.

	Selon le type de l'obstacle, l'appareil émettra des vibrations et des sons modélisant le type de l'obstacle heurté.
Contraintes	Trouver des sons et les diffuser et gestion du stéréo. Mettre un place un monde sonore et des vibrations modélisant l'environnement du personnage.
Niveau de priorité	Priorité supérieure

Fonction 4 : Déclenchement d'un compte à rebours	
Description	La difficulté du jeu se caractérise en partie par le fait que le joueur est pressé par le temps et doit atteindre son objectif avant la fin du compte à rebours déclenché une fois le jeu lancé. La durée dépend de la difficulté choisie auparavant.
Contrainte	Le minuteur ne doit pas être gênant : il ne doit pas cacher une partie du labyrinthe (par exemple des chiffres dans un angle de l'écran).
Niveau de priorité	Priorité moyenne

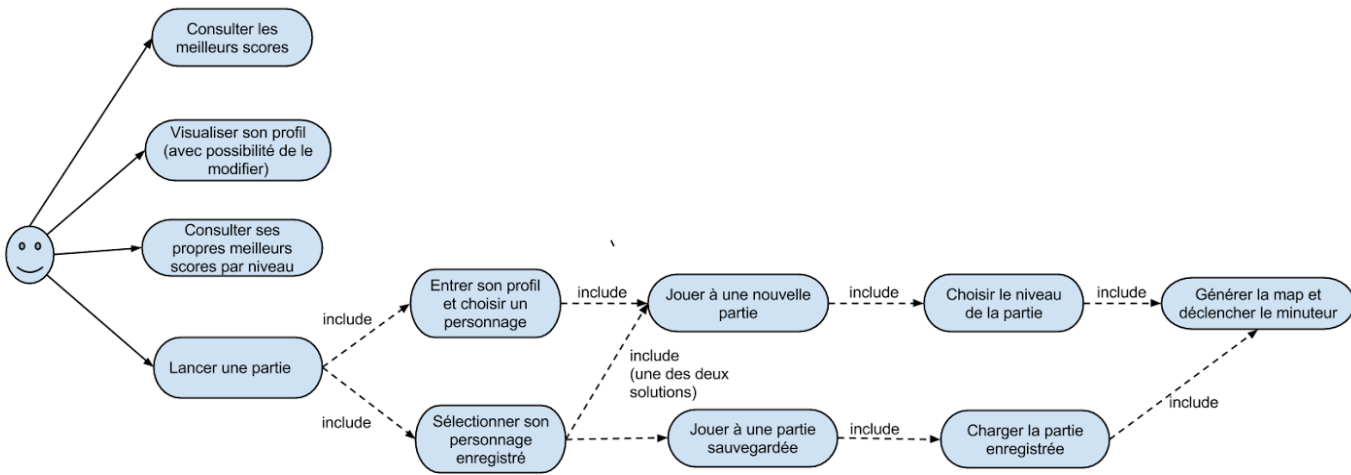
Fonction 5 : Faire perdre le joueur	
Description	Différents cas de défaite sont possibles. Si le personnage rencontre un obstacle létal, alors il perd la partie et recommence sa partie au début du niveau perdu. Au bout d'un certain nombre d'obstacles heurtés, lorsque la vie du personnage est épuisée, il perd également. Enfin, si le joueur n'atteint pas l'objectif dans le temps imparti, il perd. Après avoir perdu, le joueur est ramené au début du niveau qu'il vient d'échouer.
Contrainte	Avoir un compteur d'obstacles rencontrés par le joueur. Faire apparaître la vie restante par une barre de vie ou un autre moyen.
Niveau de priorité	Priorité moyenne

Fonction 6 : Faire une pause	
Description	En touchant la touche pause, le minuteur s'arrête et les déplacements ne sont plus possibles. En touchant à nouveau la touche le jeu est relancé. Lorsque l'utilisateur éteint l'écran, le jeu est également mis en pause.
Contraintes	Arrêter le minuteur et le relancer là où il en était. Mettre certains écouteurs (notamment celui de l'accéléromètre) en pause.
Niveau de priorité	Priorité inférieure

Fonction 7 : Animation modélisant la victoire ou la défaite à la fin du jeu	
Description	A la fin de la partie, on assiste au lancement d'une animation (de victoire ou défaite).
Contraintes	On verra la possibilité de créer une animation 2D ou 3D.
Niveau de priorité	Priorité inférieure

Fonction 8 : Quitter le jeu	
Description	Lors de la partie, le bouton pause donnera également accès à la possibilité de sauvegarder sa partie et de quitter le jeu.
Contraintes	
Niveau de priorité	Priorité inférieure

Diagramme de cas d'utilisations



Ce diagramme présente les actions directes que l'utilisateur peut effectuer dans l'application.

Il peut d'une part lancer une partie. Ceci implique que le joueur entre tout d'abord son profil (en donnant son identifiant et en choisissant un personnage, qui serait notamment caractérisé par une couleur) ou qu'il reprenne son personnage si l'utilisateur a déjà son profil dans l'application. Le joueur peut ensuite choisir de commencer une nouvelle partie ou de reprendre sa partie sauvegardée. Si le joueur choisit la première option, il devra également sélectionner le niveau de difficulté de sa partie, parmi ceux qu'il a déjà débloqués. Une fois ces étapes réalisées, la map correspondant aux différents choix est générée.

Enfin, l'utilisateur peut consulter les meilleurs scores, classés par niveau, tous joueurs confondus, ou bien visualiser son profil, où il peut modifier ses données et avoir accès à ses scores, ordonnés par niveau déjà joué.

Diagramme d'interactions (voir annexe 1)

Ce diagramme représente les trois interactions principales entre l'application et l'utilisateur.

La première action est celle d'initialiser la partie, l'utilisateur, en cliquant le bouton « jouer » va lancer le chargement de l'écran de sélection du profil. Une fois que l'utilisateur a repris un profil existant ou en a créé un nouveau, l'écran du choix de la partie (nouvelle partie ou reprise d'une partie sauvegardée) est chargée. Dans le cas où l'utilisateur décide de commencer une nouvelle partie, un menu de sélection du niveau de jeu est ensuite affiché, menu que nous n'avons pas représenté ici.

Les deux choix de partie (sauvegardée ou nouvelle) amènent à l'interaction de jeu. La map est chargée par la machine, qui va ensuite charger les sons de l'environnement du personnage et écouter l'accéléromètre.

Si le téléphone n'est pas incliné, l'application va seulement jouer les sons et continuer à écouter.

S'il est incliné, l'application va alors regarder ce qui se trouve à l'emplacement où le personnage veut aller.

- S'il y a un obstacle létal, la partie est finie, et un écran de défaite est affiché. Le score est enregistré et l'application demande de choisir entre un retour au menu principal ou un nouvel essai (non modélisé par soucis de clarté du diagramme).
- S'il y a un obstacle normalement non létal, un certain nombre de vie(s) est retiré au personnage. S'il est mort, on effectue les mêmes actions que lorsque un obstacle létal. Autrement, le son à émettre lors du heurt de l'obstacle est joué et le téléphone vibre, puis l'application écoute à nouveau l'accéléromètre et jouer les sons de l'environnement.
- Si le déplacement amène à l'objectif à atteindre, on a chargement de l'écran de victoire, affichage de celui-ci et enregistrement des scores. Le joueur a ensuite le choix entre revenir au menu ou jouer à nouveau.

Si l'emplacement où le personnage doit aller est vide, il va pouvoir se déplacer, l'affichage est donc actualisé avec la nouvelle position. Les sons joués par l'application sont également actualisés en fonction de l'environnement. L'application écoute à nouveau l'accéléromètre.

La dernière action possible est d'afficher les scores. Le jeu charge le tableau des meilleurs scores et va les afficher à l'utilisateur.

Diagramme de classes (voir annexe2)

La classe principale est la classe Partie. Elle sera créée par le MenuPartie, soit lors du lancement d'une nouvelle partie (son constructeur prenant en paramètres le profil de l'utilisateur et le niveau de jeu choisi par celui-ci), soit lors du chargement d'une ancienne partie, dont les caractéristiques sont sauvegardées dans un fichier xml. En effet, lorsque l'utilisateur quittera la partie, ces attributs (profil et niveau) seront sauvegardés et rattachés à la partie sauvegardée.

Partie sera l'activité principale.

Les sons sont référencés en tant qu'attribue par leur identifiant, de type int.

Selon la position d'un obstacle proche du personnage, un même son (correspondant à l'obstacle) sera joué, avec un volume différent, dans chaque oreille, afin de permettre au joueur de repérer la position de l'obstacle. Par ailleurs, en fonction de la distance entre chaque obstacle environnant et le personnage, un volume global est défini pour les sons "émis" par l'obstacle (plus l'obstacle est proche, plus le volume global, à gauche et à droite, est élevé).

La gestion du son est gérée par la classe Sons, qui est un SoundPool. Seul l'attribut obstaclesProches (faisant référence à un tableau d'obstacles suffisamment près du personnage pour générer des sons de repérage) est fourni par la classe Partie, par l'intermédiaire de la méthode chercherObstaclesProches(). Cette dernière modifie les attributs volumeGlobal, volumeDroite et volumeGauche des obstacles concernés, pour que la classe Sons puissent émettre les bons sons au bon volume.

Plusieurs extensions d'obstacles pourront exister (obstacle létal ou non, mobile...). L'objectif pourra être une extension d'obstacle.

L'affichage des meilleurs scores (classement global) se fera à partir d'une ArrayList des meilleurs scores. Pour afficher la valeur de ces scores dans un tableau, on utilisera la méthode toString() de chaque score.

Des fichiers xml permettront de sauvegarder les informations d'une partie, d'un profil ainsi que les meilleurs scores. Les méthodes sauvegarder() des classes Profil et

MeilleursScores et la méthode sauvegarderPartie() de la classe Partie permettent de conserver les données de jeu dans ces fichiers. Ces données pourront être récupérées à partir de méthodes se trouvant dans les classes qui les utilisent.

La classe MenuNiveau, qui permet de gérer l'affichage du quatrième menu permettant le choix du niveau de jeu avant le début d'une nouvelle partie afin d'alléger le diagramme.

Les types ...Tab, par exemple booleanTab signifient que l'attribut est un tableau de booléens pour l'exemple choisi (les crochets ne peuvent être insérés dans le type des attributs).