**Manuel technique "Pixters"**

**Généralités**

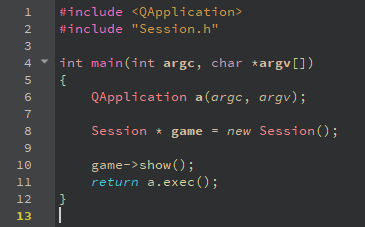
L'application est codée en C++ et utilise le framework Qt. Les scores des utilisateurs, la liste des espèces, des informations sur la session en cours et les informations des comptes admin sont stockés dans une base de données MySQL (version 8.0.27) gérée par phpMyAdmin 5.1.1 et accessible avec le nom d'utilisateur "root" et le mot de passe "AT2jgTMEx-cHeIT9". L'environnement de développement est WampServer 3.2.6. Le versioning du logiciel est fait via GitHub.

Les graphismes ainsi que la musique et les bruitages ont été créés sur mesure respectivement dans Krita et FL Studio.

**Les classes**

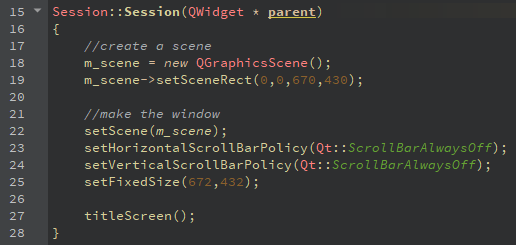
**Démarrage de l'application** (main.cpp)

Lors du démarrage de l'application, la fonction "*main()*" crée uniquement un objet "*Session*" qui gère les sessions de jeu, puis affiche la session en cours avec *show()*.

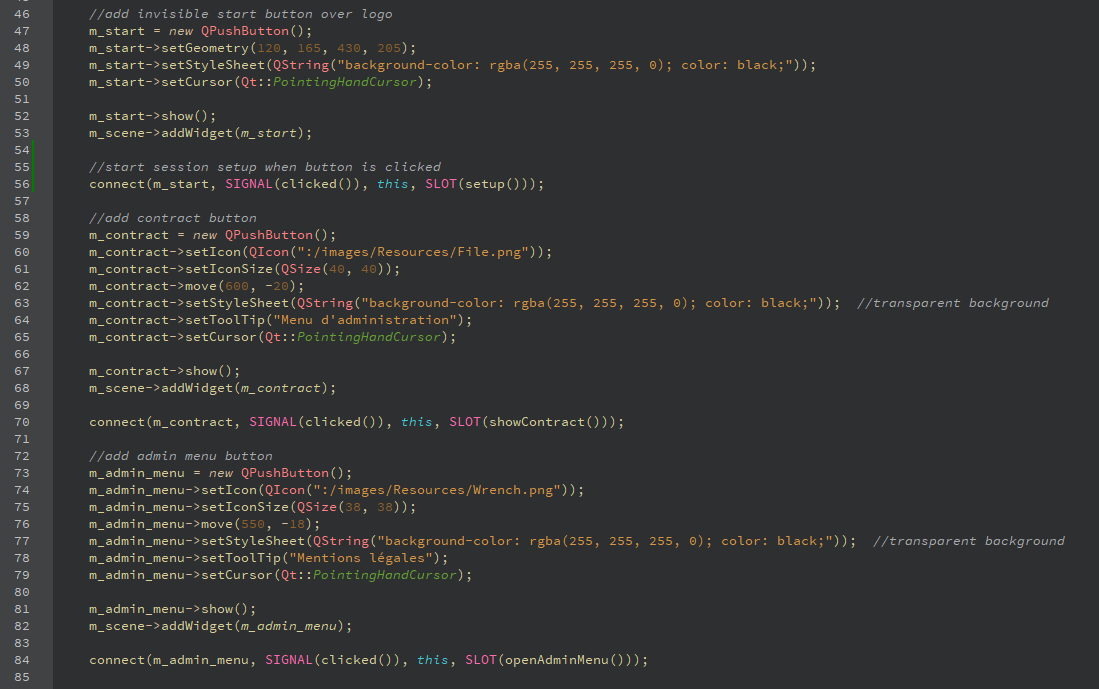


**La session** (Session.cpp/h)

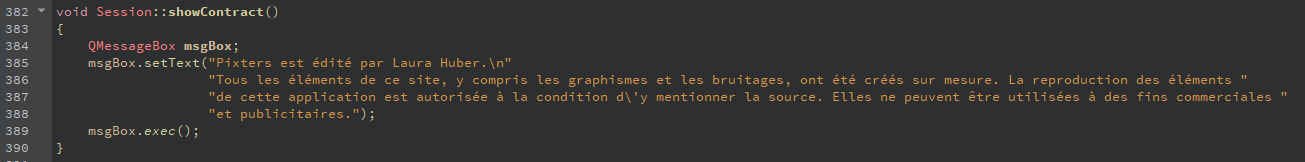
Le constructeur de la classe "*Session*" crée une scène destinée à contenir les élements graphiques et configure la fenêtre d'affichage. Il appelle ensuite la fonction "*titleScreen()*".



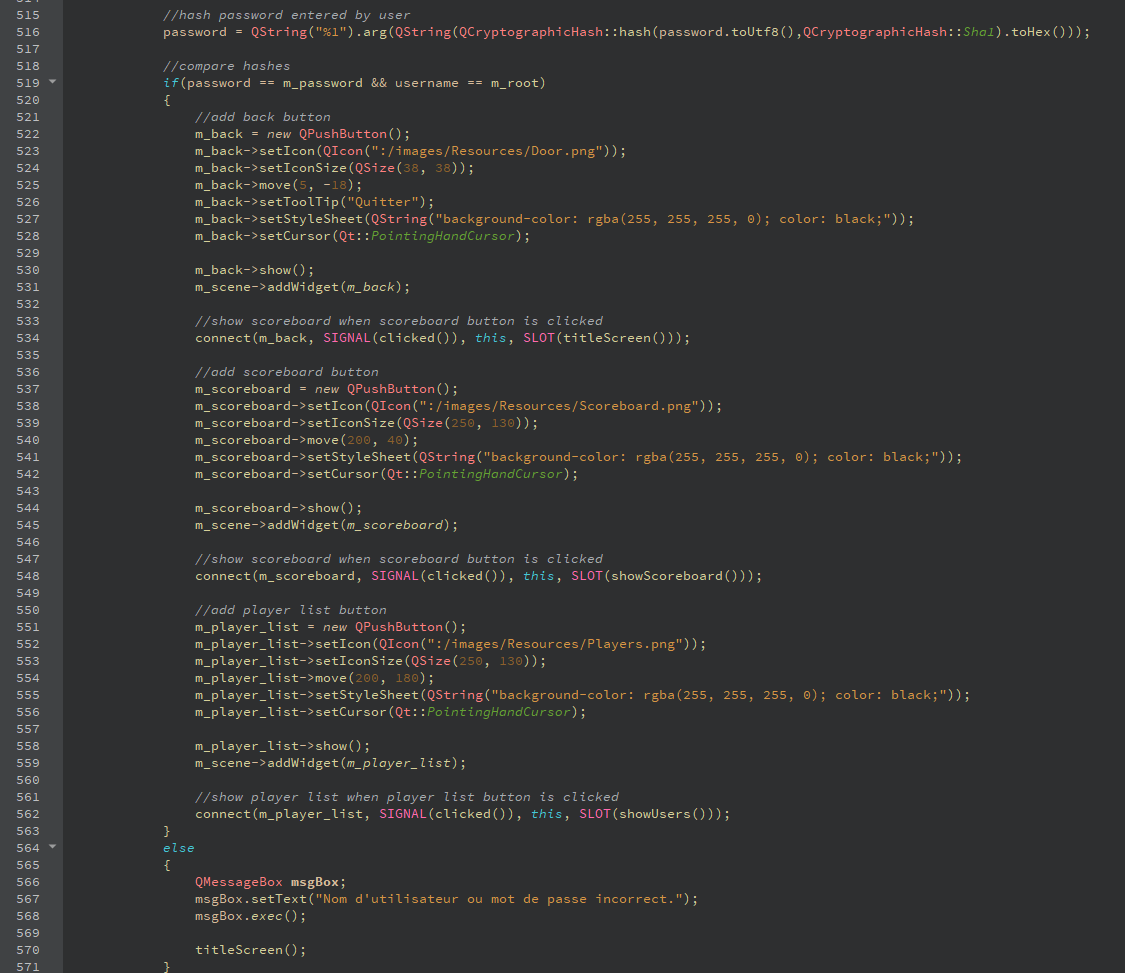
La fonction "*titleScreen()*" affiche l'arrière-plan et met en place les boutons ainsi que la musique de l'écran titre. Le bouton "*m\_contract*" appelle la fonction "*showContract()*", le bouton "*m\_admin\_menu*" appelle la fonction "*openAdminMenu()*" et le bouton "*m\_start*" appelle la fonction "*setup()*".



La fonction "*showContract()*" affiche les mentions légales dans une fênetre pop-up.



La fonction "*openAdminMenu()*" affiche l'arrière-plan et les boutons du menu d'administration après avoir vérifié les identifiants entrés par l'utilisateur. Le bouton "*m\_back*" appelle la fonction "*titleScreen()*", le bouton "*m\_scoreboard*" appelle la fonction "*showScoreboard()*" et le bouton "*m\_player\_list*" appelle la fonction "*showUsers()*".



La fonction "*showScoreboard()*" affiche la liste des scores, ainsi que la date et l'utilisateur qui y sont associé, triés par ordre décroissant dans une nouvelle fênetre.



La fonction "*showUsers()*" affiche la liste des utilsateurs dans une nouvelle fênetre.



La fonction "*setup()*" affiche des fênetres pop-up permettant d'entrer un nom d'utilisateur et de choisir l'espèce et le nom à donner à son pixter. Elle se connecte ensuite à la base de données et y insère les données préalablement fournies. Finalement, elle appelle la fonction "*start()*".



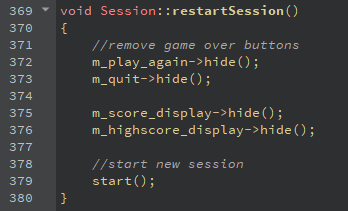
La fonction "*start()*" affiche l'arrière-plan de la session et change la musique de fond. Elle crée un objet "*Pet*" et un objet "*Meter*" et affiche le nom du pixter. L'objet "*Satisfaction*" de "*Meter*" appelle la fonction "*gameOver()*" lorsque sa valeur change ("*m\_death*" devient *true*).



La fonction "*gameOver()*" affiche l'arrière-plan de fin de jeu et change la musique de fond, met en place les boutons "*m\_play\_again*" et "*m\_quit*" et affiche le score de l'utilisateur ainsi que le meilleur score obtenu par l'ensemble des joueurs. Elle stocke également le score de l'utilisateur dans la base de données. Le bouton "*m\_play\_again*" appelle la fonction "*restartSession()*" et le bouton "*m\_quit*" appelle la fonction "*close()*" qui ferme l'application.



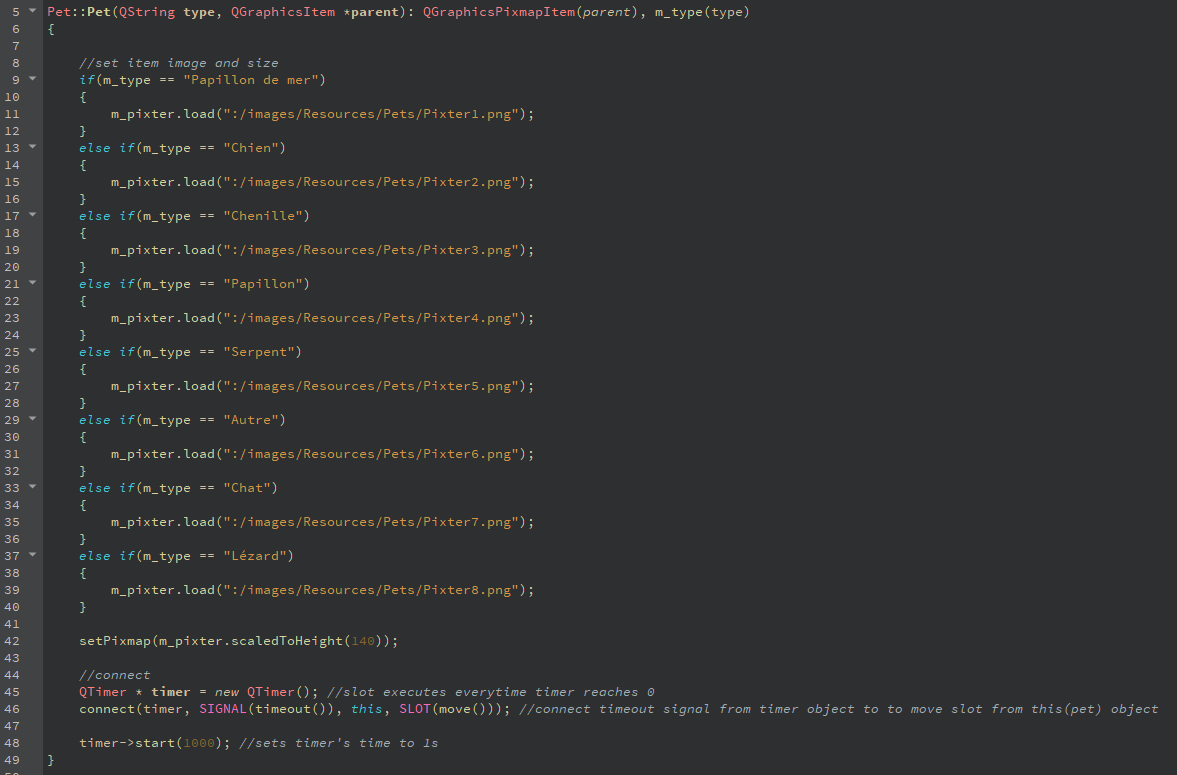
La fonction "*restartSession()*" appelle la fonction "*start()*" afin de démarrer une nouvelle partie.



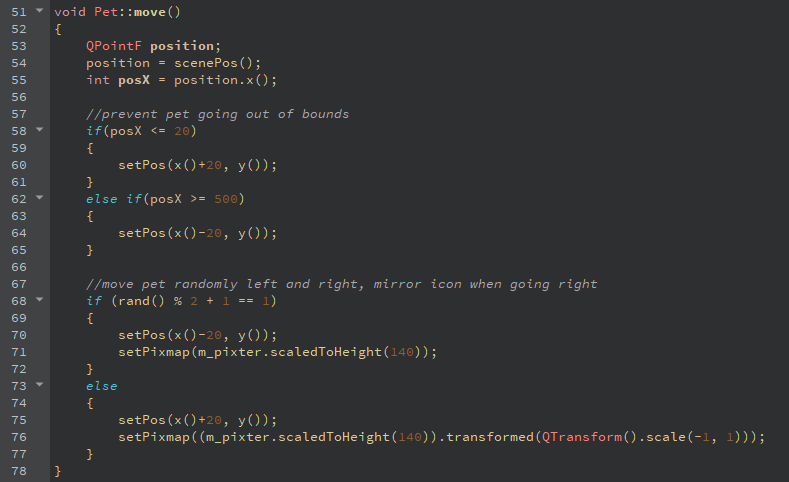
La classe dispose en outre de getters et de setters pour les variables "*m\_username*", "*m\_pet\_type*" et "*m\_pet\_name*".

**Le pixter** (Pet.cpp/h)

Le constructeur de la classe "*Pet*" initialise l'image du pixter en fonction de l'espèce choisie, stockée dans la variable "*m\_type*". Il crée également un objet "*QTimer*" qui appelle la fonction "*move()*" une fois par seconde, permettant le déplacement régulier à l'écran de l'image du pixter.



La fonction "*move()*" fait bouger le pixter horizontalement et de façon aléatoire dans un cadre fixe, et adapte l'orientation de l'image en fonction de la direction du mouvement.



La classe dispose également de getters et de setters pour les variables "*m\_name*" et "*m\_type*".

**L'interface** (Meter.cpp/h)

Le constructeur de la classe "*Meter*" crée les boutons "*m\_feed*" et "*m\_heal*" qui appellent respectivement la fonction "*eat()*" et "*healthIncrease()*". Il crée également l'affichage du score, plusieurs objets "Health" pour la barre de santé, des objets "*Hunger*" pour la barre de faim et un objet "Satisafaction" pour l'indicateur de satisfaction. En outre, il initalise quatre objets "*QTimer*" qui appellent régulièrement les fonctions "*increaseScore()*", "*healthDecrease()*", "*starve()*" et "*happinessChange()*", permettant une évolution dynamique de la session.



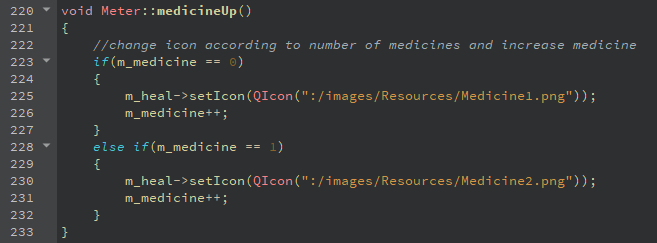
La fonction "*healthDecrease()*" appelle la fonction "*healthDown()*" de la classe "*Health*" quand le nombre de fraises est égal à 0. Elle joue également un son d'alerte lorsque le nombre de cœur atteint 1.



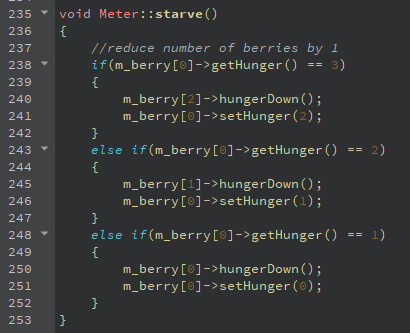
La fonction "*healthIncrease()*" appelle la fonction "*healthUp()*" de la classe "*Health*" lorsque le nombre de médicaments restants est supérieur à 0 et réduit de 1 le nombre de médicaments. Elle joue également un son signalant le bon déroulement du soin ou une alerte en fonction du nombre de médicaments disponibles.



La fonction "*medicineUp()*" augmente de 1 le nombre de médicaments stocké dans "*m\_medicine*" et change l'icône du bouton en fonction stock actuellement disponible.



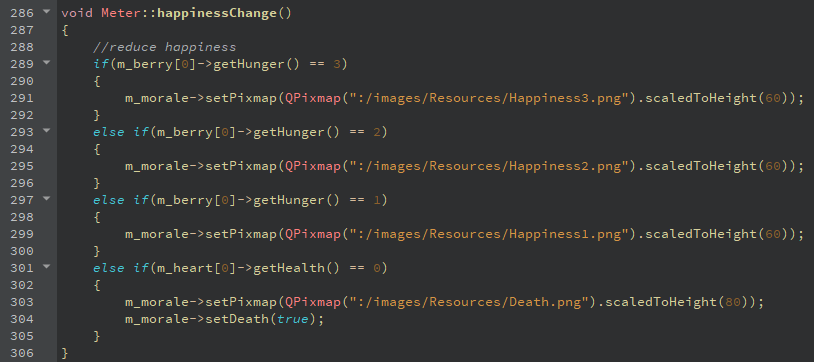
La fonction "*starve()*" appelle la fonction "*hungerDown()*" de la classe "*Hunger*".



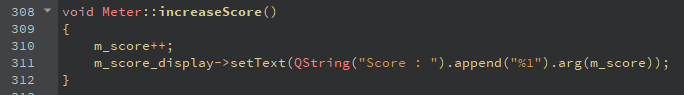
La fonction "*eat()*" joue un son indiquant que le pixter mange, et appelle la fonction "*hungerUp()*" de la classe "*Hunger*".



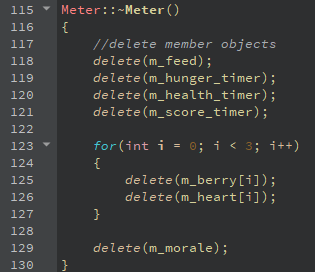
La fonction "*happinessChange()*" change l'icône de l'objet "*Satisfaction*" en fonction du nombre de fraises restantes et met la valeur de "*m\_death*" de la classe "*Satisfaction*" à *true* lorsque le nombre de cœurs atteint 0.



La fonction "*increaseScore()*" augmente de 1 le score stocké dans "*m\_score*".

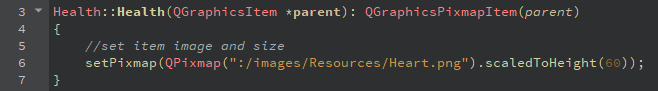


La classe dispose en outre d'un getter et d'un setter pour la variable "*m\_score*" ainsi que d'un destructeur.

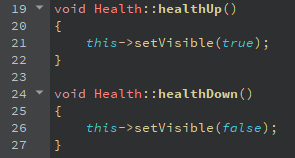


**La barre de santé** (Health.cpp/h)

Le constructeur de la classe "*Health*" initialise l'image de l'objet.



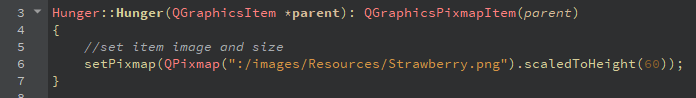
Les fonctions "*healthUp()*" et "*healthDown()*" rendent respectivement l'objet visible ou invisible.



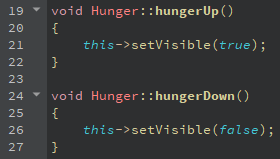
La classe dispose également d'un getter et d'un setter pour la variable "*m\_health\_nb*" qui stocke le nombre de cœurs.

**La barre de faim** (Hunger.cpp/h)

Le constructeur de la classe "*Hunger*" intitialise l'image de l'objet.



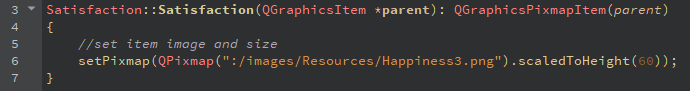
Les fonctions "*hungerUp()*" et "*hungerDown()*" rendent respectivement l'objet visible ou invisible.



La classe dispose également d'un getter et d'un setter pour la variable "*m\_berry\_nb*" qui stocke le nombre de fraises.

**L'indicateur de satisfaction** (Satisfaction.cpp/h)

Le constructeur de la classe "*Satisfaction*" initialise l'image de l'objet.



La classe dispose d'un getter et d'un setter pour la variable "*m\_death*". Si celle-ci est changée, le setter émet le signal "*valueChanged()*".