



RAPPORT DE PROJET T.E.R

PROJET INFORMATIQUE HLIN405

PUNYDUCK



Etudiants :

- Valentin FONTAINE
- Paul BUNEL
- Esteban BARON
- Valentin PERON
- Julien LEBARON

Année : 2019-2020

Encadrante : Anne-Elisabeth BAERT

Sommaire

Introduction	1
1 Technologies utilisées	3
1.1 Langages	3
1.2 Outils	3
2 Développement logiciel, Conception, Modélisation, Implémentation	5
2.1 Présentation du développement	5
2.2 Modélisation	8
2.3 Fonctionnalités de l'interface	10
2.4 Statistique	11
3 Algorithmes et Structures de Données	12
3.1 Présentation des principales structures de données	12
3.2 Présentation des principaux algorithmes	12
3.3 Complexité théorique	12
4 Gestion du Projet	13
4.1 Organisation et planification	13
5 Bilan et Perspectives	15
6 Bibliographie et Annexes	16

Introduction

Dans le cadre du TER de notre deuxième année à la faculté des sciences de Montpellier nous avons proposé un projet s'intitulant PunyDuck. Le but de ce projet est la réalisation d'une plate-forme de distribution des projets des étudiants du département informatique.

Le groupe de développement est composé de cinq personnes, Valentin FONTAINE, Paul BUNEL, Valentin PERON, Julien LEBARON et Esteban BARON. Nous sommes encadré par Mme Anne-Elisabeth BAERT.

Motivation

Le TER est un module qui apporte beaucoup aux étudiants en gestion de projet ainsi qu'en programmation. Seulement une fois terminés les projets ne sont pas valorisés et tombent dans l'oubli. Notre solution est de proposer une application permettant à chaque étudiants de déposer leurs projets pour les rendre visibles et téléchargeables par tous.

Approches

Notre objectif final étant de produire une application fonctionnelle pouvant être utilisée par tous, nous avons besoin : d'une interface graphique décente, totalement opérationnelle et facilement modulable, ainsi que d'un système de communication efficace entre l'application et une base de donnée. Nous avons donc décidés de diviser le projet en trois parties : tout d'abord la conception d'un framework permettant la réalisation d'interfaces graphiques, puis la réalisation de l'interface graphique elle-même à partir de ce framework, et enfin la partie réseau permettant à l'application de communiquer avec un serveur distant. Ces trois parties communiquent ensembles part le biais d'une architecture logiciel MVC (Modèle-Vue-Contrôleur).

Notre application est téléchargeable par tous depuis le site web <https://fvostudio.com/punyduck/>.

Cahier des charges

Le but de Punyduck est de proposer aux étudiants en informatique une plateforme de distribution de leurs différents projets. Cette plateforme devait être une application téléchargeable sur un site internet, et connectée à un serveur pour permettre l'échange de projets entre différents clients.

Pour l'organisation du projet, nous l'avons tout d'abord divisé en plusieurs étapes :

- Mise en place d'un serveur qui servira d'intermédiaire entre les utilisateurs et la base de données.
- Création d'un framework pour faciliter la réalisation de l'application.
- Conception de l'application graphique à l'aide du framework.
- Connexion entre l'application graphique et le serveur.
- Création d'une base de données pour stocker les comptes des utilisateurs et les projets.
- Mise en service d'un site internet permettant le téléchargement de l'application.

Ensuite, une fois les différentes parties du projet dégagées, pour bien définir chacune d'entre elles nous avons décidé du cahier des charges suivant :

Le serveur : Le serveur devra fonctionner de manière asynchrone (nous définiront ce principe dans la prochaine partie). Il pourra héberger de manière sécurisée les données des utilisateurs (nom, mots de passes, ...) et devra être capable de gérer la plupart des erreurs de réseau, comme les coupures de connexion lors d'un téléchargement.

Le framework : Le framework définira un ensemble de classes et fonctions qui serviront de base à la structure d'une nouvelle application. Son utilisation doit être facile avec assez de fonctionnalités déjà disponibles pour pouvoir réduire les appels de fonctions bas niveau. De plus, il devra également permettre la plasticité des interfaces.

L'application : Notre application présentera une interface graphique permettant de naviguer facilement entre différents onglets, de communiquer avec le serveur, et de personnaliser un minimum l'apparence : mode clair / sombre, position de certains éléments de la fenêtre, couleur de l'arrière plan.

La base de données : Elle stockera les données relatives aux comptes des utilisateurs ainsi que les projets. Les données confidentielles (mots de passe) seront cryptées.

Le site : Il sera constitué d'une unique page permettant de télécharger l'application. Il possèdera un design « responsive » et dynamique.

Partie 1

Technologies utilisées

1.1 Langages

Pour la programmation de notre application, nous avons choisis assez naturellement d'utiliser le langage C/C++. En effet, c'est le langage que nous avons le plus étudié à l'université, avec lequel 3 des 5 membres du groupe avaient codé leur projet de Licence 1 CMI, et qui nous offrait un support puissant et efficace pour notre interface graphique.

De plus le C++ nous a permis de coder en orienté objet, ce qui était une nécessité pour la création de notre application. En effet, notre programme utilise énormément ce type de programmation : nous développeront ce point plus tard dans la partie 3 sur le développement logiciel.

Pour l'interface graphique, nous avons utilisé la bibliothèque OpenGL : une interface regroupant environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plates-formes, OpenGL est une des bibliothèques graphiques les plus utilisées par la majorité des applications scientifiques, industrielles ou artistiques 3D.

L'utilisation de cette bibliothèque (associée à GLFW pour la gestion des fenêtres) était donc un choix assez basique puisqu'elle est très répandue, mais nous avons dû apprendre à l'utiliser : celle-ci est beaucoup plus complexe que les librairies que nous avons utilisées auparavant, comme SFML ou PyGame, de par son bas niveau.

Pour gérer la partie réseau de notre application, nous n'avions pas besoin d'utiliser un langage puissant comme le C/C++, nous pouvions donc nous rabattre sur un langage moins performant mais plus facile d'accès. De ce fait, nous nous sommes dirigés vers le langage python, qui est un langage très simple d'utilisation, pratique pour débiter dans le domaine complexe de la programmation que représente la programmation en réseau. De surcroît il est plutôt efficace pour gérer les entrées sorties, notamment pour la lecture et l'écriture dans des fichiers, qui est une notion centrale dans notre projet.

Cependant, le langage Python de base ne permet pas de faire de la programmation en réseau, nous avons donc du choisir une des nombreuses bibliothèques disponibles permettant de faire de la programmation réseau en Python. Après réflexion, nous avons opté pour le module « asyncio » pour deux raisons : premièrement, ce module offrait une interface de programmation en réseau de haut niveau donc plus simple d'utilisation, ce qui nous arrangeait particulièrement étant donné que nous sommes parfaitement novices dans ce domaine ; Deuxièmement, le gros point fort de ce module est le fait qu'il implémente une nouvelle manière de programmer : la programmation asynchrone. La programmation asynchrone pour les entrées/sorties est une forme de programmation parallèle permettant d'exécuter d'autres parties d'un programme lorsque celui-ci est en attente d'une transmission de donnée, afin de grandement diminuer le temps d'exécution du programme.

1.2 Outils

Pour réaliser notre projet nous avons utilisé des outils différents et spécifiques à chaque tâche.

Nous avons utilisé le système de gestion PostgreSQL¹ pour gérer la base de donnée de notre serveur, ce système nous permettant d'utiliser des requête SQL en Python et ainsi de mettre en application les connaissances acquises cette année avec le module HLIN304.

Pour la partie modélisation de l'application nous avons opté pour le Langage de Modélisation Unifié (UML) vu en cours, dans le module HLIN406.

Toutes les communications du groupe ce sont faites sur le logiciel Discord², un logiciel facilitant grandement les communications en groupes avec par exemple le partage d'écran, les groupes vocaux et textuels, le fait de pouvoir partager des morceaux de code directement dans le canal de discussion textuel etc..

Pour ce qui est du partage et des sauvegardes du code, nous avons utilisé Git (un logiciel de gestion de versions décentralisé) via un serveur GitHub³ qui nous a permis de garder nos anciennes versions, de ne rien perdre en cours de route, et de pouvoir partager l'avancée du projet avec les autres étudiants du groupes et notre encadrante.

Pour ce qui est de l'éditeur de code utilisé, nous nous sommes tous penché sur Visual Studio Code⁴ pour sa fiabilité et sa mise en page agréable. De plus, nous avons pu grâce à ce logiciel coder à plusieurs en même temps avec sa fonctionnalité de partage en temps réel.

Enfin, nous avons utilisé le langage L^AT_EX via la plateforme Overleaf⁵ pour rédiger ce rapport.

1. PSQl : <https://www.postgresql.org/>

2. Discord : <https://discordapp.com/>

3. GitHub <https://github.com/>, notre projet : <https://github.com/valfvo/punnyduck>

4. VSCode : <https://code.visualstudio.com/>

5. Overleaf : <https://www.overleaf.com/>

Partie 2

Développement logiciel, Conception, Modélisation, Implémentation

2.1 Présentation du développement

Le développement du projet est séparé en trois parties :

- le framework de l’interface graphique ;
- l’interface graphique utilisant le framework ;
- la communication réseau entre le client et le serveur.

Le développement de l’application s’est déroulé en plusieurs étapes qui vont être développées points par points.

2.1.1 Apprentissage de l’utilisation d’OpenGL

La première étape fut d’apprendre l’OpenGL, qui fut difficile à comprendre et à utiliser de par son bas niveau comparé aux bibliothèques qu’on a l’habitude de voir en L1 et L2.

La première partie de l’apprentissage d’OpenGL a été de pouvoir dessiner un triangle d’une certaine couleur. Puis grâce à ce triangle de pouvoir dessiner un rectangle. Ainsi de suite nous avons pu dessiner toutes sortes de formes tel que les ronds et les cercles par exemple. La deuxième partie a été de pouvoir coller des textures sur ces surfaces précédemment créées. Cette partie n’a pas posé réellement de problème. Pour la troisième partie, nous devons pouvoir mettre une surface dans une autre surface, comme dans la figure 2.1. Ceci était primordial pour la réalisation de l’application graphique car cette simple tâche a permis la superposition des éléments graphiques de l’application.

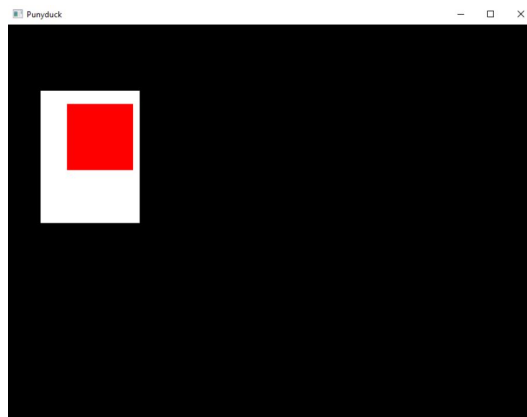


FIGURE 2.1 – Capture d’écran de notre premier essai d’une superposition de surfaces

La dernière partie était de créer le texte, c'est-à-dire de pouvoir afficher du texte à l'écran.

La seconde étape du développement de l'application a été de modéliser en UML les différentes parties de l'application et ses mécanismes.

Pour commencer, il fallait trouver un système permettant de lier des éléments entre eux afin de pouvoir les superposer, en créer des nouveaux à partir d'anciens, de supprimer un élément et ses sous éléments qui le compose etc.. C'est pour cela que nous avons créé les "Quarks", un système où un élément a un père et n fils (où n appartient à l'intervalle $]0, +\infty[$) et a accès à ses frères. Grâce à ce mécanisme, tous les problèmes cités précédemment ont été résolus. Ce système a donc été la base de toute l'application.

Ensuite nous avons créé les différentes classes qui allaient nous servir pour l'implémentation. Comme par exemple :

- "LQTexture" qui génère les textures
- "LQSurface" pour les surfaces
- "LQViewable"
- "LQColor" pour la couleur
- "LQNumber"
- "LQText" pour le texte
- "LQButton" pour les boutons

La troisième étape fut la phase d'implémentation. D'une part nous nous sommes occupés des classes en rapport avec d'OpenGL tels que "LQShader", "LQTexture" et "LQSurface". Par la suite nous avons codé les différentes classes en suivant l'UML précédemment réalisé.

2.1.2 Création de l'interface graphique

L'interface graphique a été créée en plusieurs temps. Dans un premier temps il a fallu trouver une représentation graphique validée par tous les membres du groupe. Pour cela on a fait la liste des différentes pages qui seraient dans l'application, tel que les pages :

- "Connexion" qui correspond à la page de connexion
- "Projet" qui est la page des projets, là où on peut télécharger les projets disponibles
- "Dépôt" pour pouvoir déposer ses projets sur l'application

Puis on les a dessinés sur papier une par une pour que chaque membre du groupe ait une vision très précise de l'aspect visuel de l'application. Enfin, pour l'implémentation des pages on a utilisé le framework créé dans ce but. Il a fallu créer les différents éléments (qu'on a appelé les composants) de la page pour au final les assembler et former la page souhaitée (qu'on a appelé une vue).

Grâce au framework, on a pu lier toutes les pages (vues) entre elles permettant ainsi la navigation entre les différentes pages de l'application. Par exemple pour la page "Projet", il a fallu créer cette suite d'éléments qui la compose :

- Une liste de boutons permettant de choisir comment trier les projets ("mieux notés", "plus récents"...) ici pas encore implémenté donc c'est simplement une image
- Une barre de recherche pour accéder à un projet en écrivant son nom
- Des boutons qui permettent de trier les projets sous forme de liste ou sous forme de mosaïque avec les images des projets, ici pas encore implémenté donc ce sont simplement des images
- Les projets composés d'un nom, d'un "tag" (jeu, algorithme, application...) et d'une image

Une fois créés, on les a assemblés pour former la page. Pour cela on a créé la classe "LQTreeCreator" qui nous a facilité l'écriture des composants et permis une meilleure structure pour la création des pages. C'est une structure d'arbre, chaque composant est un "Quark" qui a une référence sur son père, son précédent frère (prevSibling) et sur son prochain (nextSibling). De plus, on peut parler de la position des composants avec des méthodes spécifiquement créées telles que top(), right(), bottom(), left(). Ainsi nous avons pu placer et créer des composants en fonction de leur père ou des

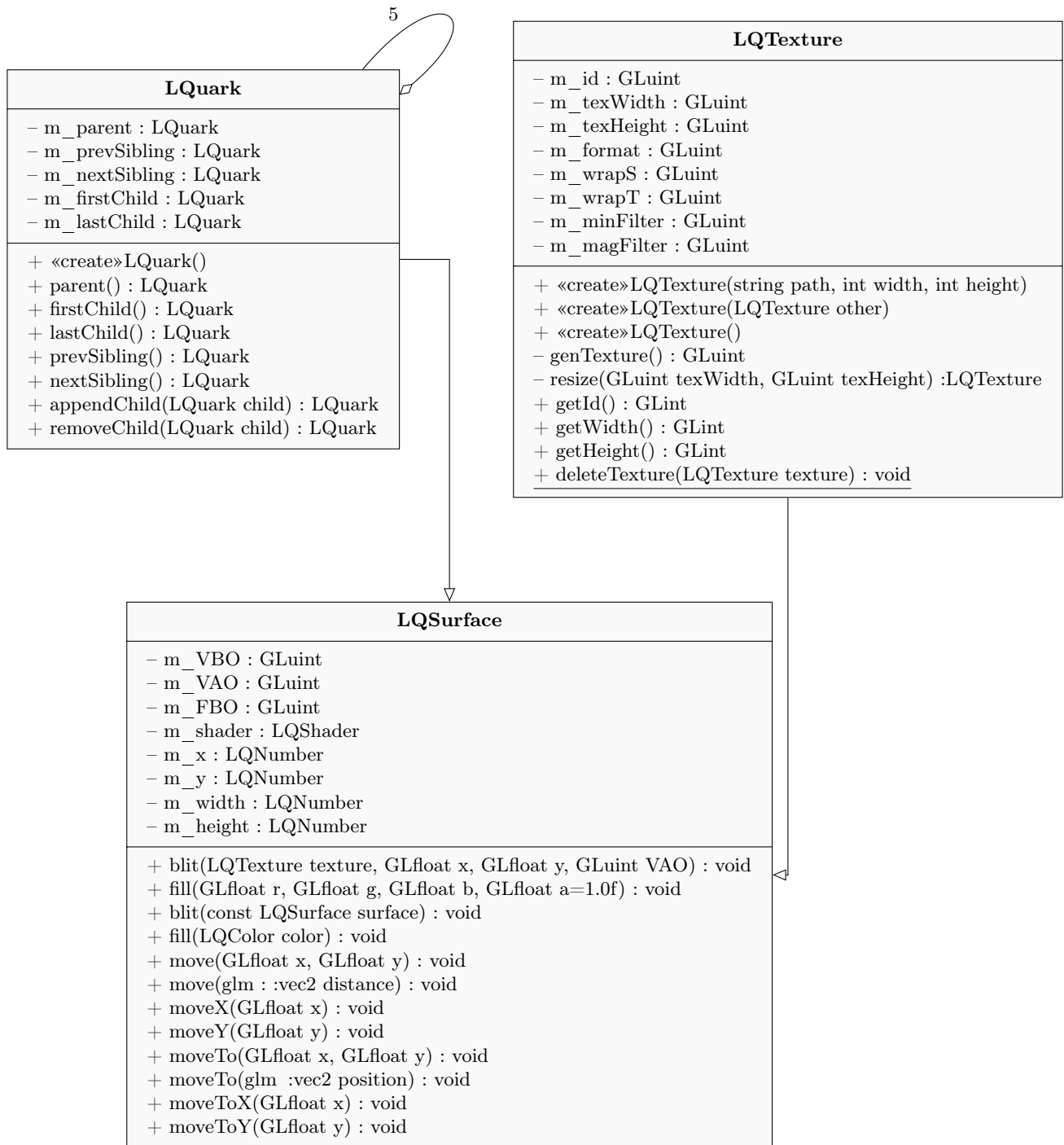
composants précédent ou suivant.

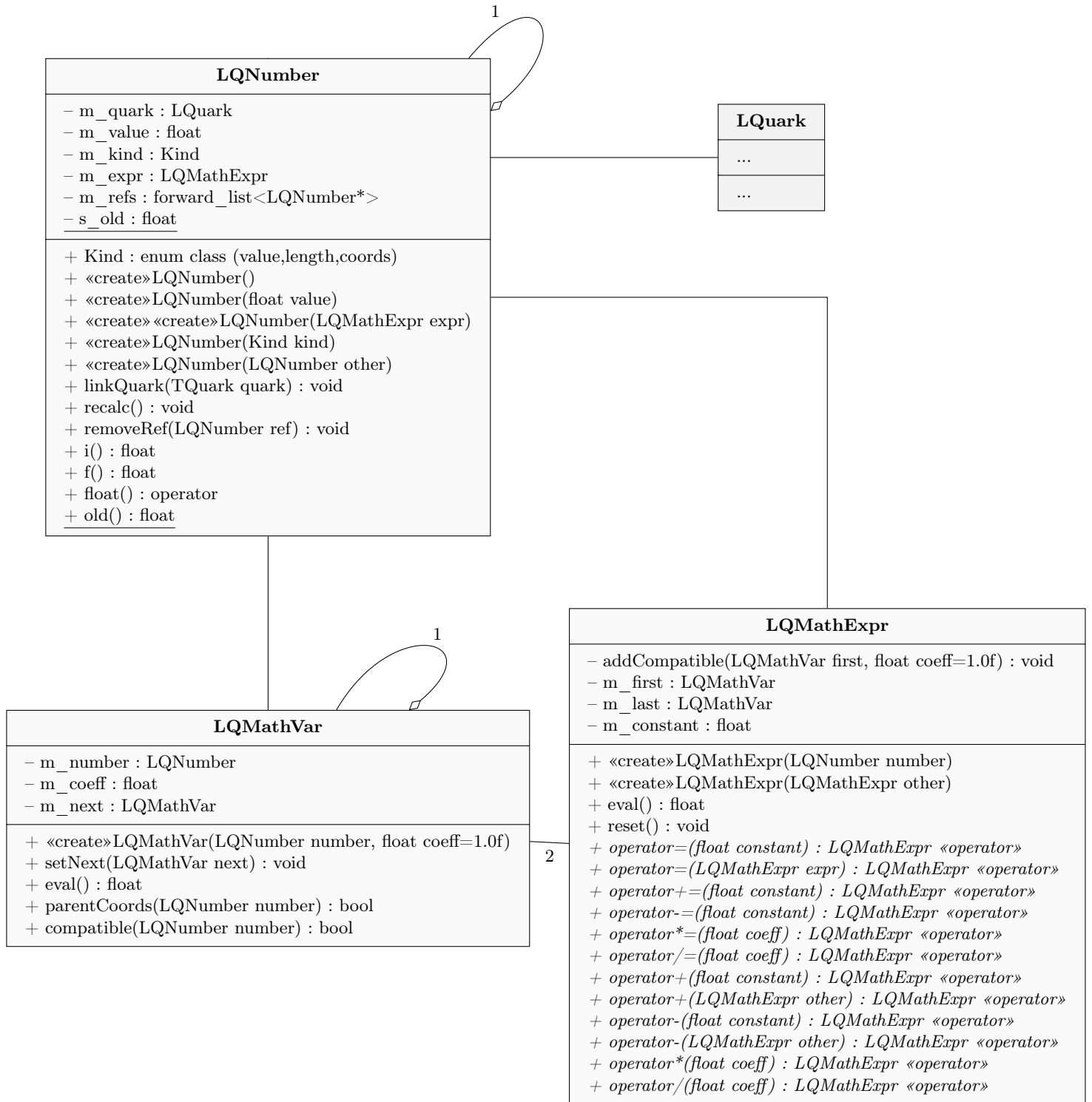
Lors de la création des pages avec les composants, on a du avoir accès au père et aux frères et aux fils. Pour cela les méthodes `sub()` et `super()` ont été créés. La méthode `sub()` permet d'accéder au fils de ce composant, amenant donc la possibilité de créer de nouveaux éléments à partir d'ancien déjà créés. Une fois le composant souhaité créé, nous voulons revenir sur composant père qui se trouve le plus souvent être la vue en elle même que l'on créer. Pour cela, on fait appelle à la méthode `super()`.

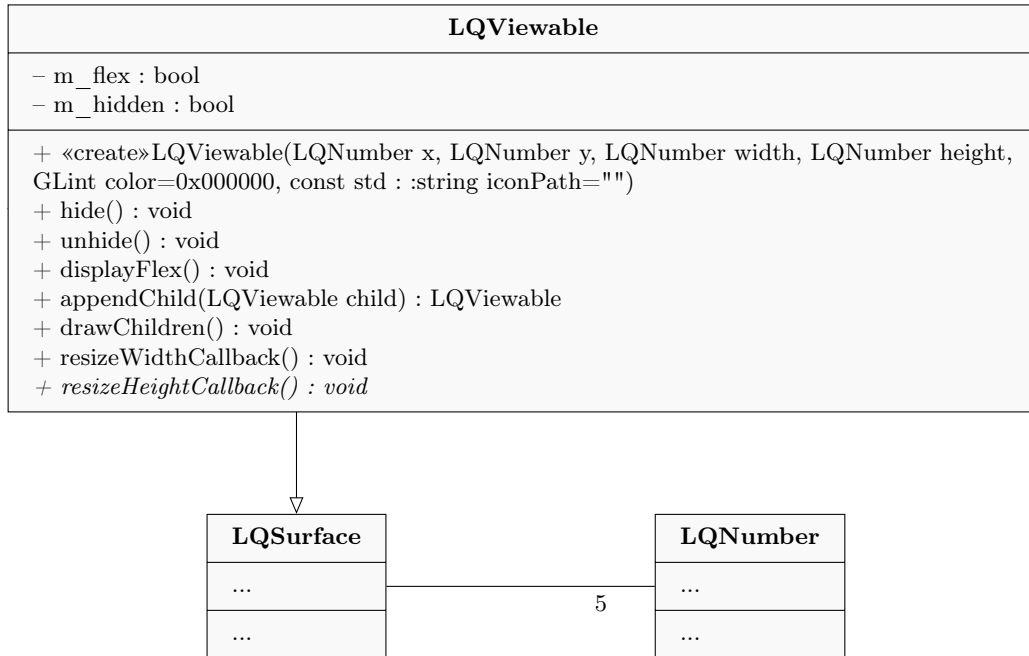
Cette structure doit nous faire penser à une structure d'arbre où l'on peut fixer des branches (composant) sur la racine ou sur la branche où l'on se situe.c'est-à-dire créer des branches à partir d'autres branches existantes et fixer cette nouvelle branche à un objet (soit la racine, soit la branche sur laquelle on se situe).

2.1.3 Communication réseaux entre le client et le serveur

2.2 Modélisation







2.3 Fonctionnalités de l'interface

L'interface graphique est composé de plusieurs fenêtres, chacune spécifique. D'autant plus que l'ensemble peut être personnalisable (placement de certains bloc présent sur l'interface). Pour la partie commune à chaque fenêtre de l'interface on a la barre des onglets contenant, dans cet ordre :

- "L'accueil", comprenant les lancements rapide d'application dernièrement utilisées ainsi que les nouveautés lié à l'application ou de certains projets.
- Les "Projets", où l'ensemble des projets mis en ligne seront affichés.
- Le "Profil" est l'endroit où l'on peut modifier ses données personnelles, sa page de profil et les paramètres de l'application.
- Le "Dépôt" vas servir à faire une demande de dépôt de projet, afin qu'il soit vérifier avant mise en ligne (pour éviter les dépôt de virus ou autre).
- Icônes basique tel que épinglé, réduire et fermer. Mais contient aussi le mode nuit (fond clair qui devient foncé).

La fonctionnalité principale étant d'aller sur la page projet, voir un qui nous intéresse ou faire une demande de dépôt pour notre projet. Maintenant cas par cas voyons les différentes interactions.

Pour l'accueil il est possible de cliquer sur les articles mis en avant enfin de pouvoir les lire en détails, mais aussi les lancements rapides des derniers projet lancer en cliquant sur l'icône en question. Pour les projets, les fonctionnalités qui changes sont les touches de tri et d'affichage ainsi que la barre rechercher. Pour la collection, c'est pareil sauf que l'on peut accéder à un menu d'action a coté de chaque projet (ressemblant à 3 petit points) :

- Désinstaller
- Emplacement
- Page projet
- Désabonner / Ne plus suivre

Ainsi que les boutons d'action, lancer qui vas démarrer l'application, et installer qui vas télécharger et installer l'application. En ce qui concerne la page de présentation du projet, on peut interagir avec la notations (données une notes ou envoyé un commentaire) mais également un lien qui va envoyé vers un forum existant en lien avec le projet. On a notamment accès au bouton suivre pour l'avoir dans notre collection et le bouton j'aime pour aimé un projet. Les interactions avec l'interface concernant le profil sont, la modification de fond et de l'image de profil ainsi que des autres paramètres (nom, prénom, mot de passe,...), l'accès au paramètre du client (langue,emplacement,...) et le bouton de déconnexion.

2.4 Statistique

Partie 3

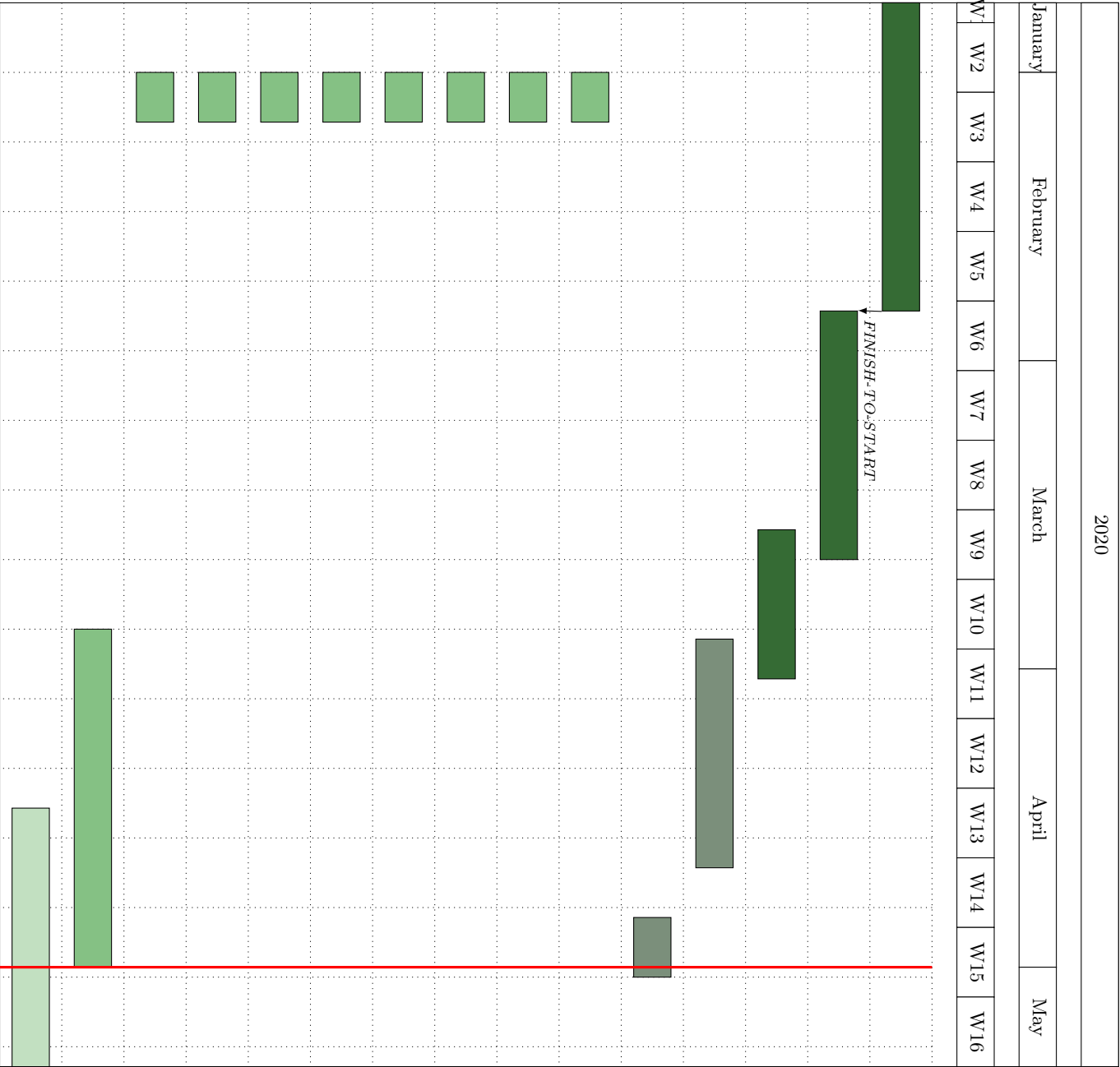
Algorithmes et Structures de Données

- 3.1 Présentation des principales structures de données
- 3.2 Présentation des principaux algorithmes
- 3.3 Complexité théorique

Partie 4

Gestion du Projet

4.1 Organisation et planification



Partie 5

Bilan et Perspectives

Le bilan de cette fin de projet est très positive, en comparant à notre cahier des charges, la quasi totalité des demandes et des objectif a été atteints. Malgré le manque de mains d'oeuvres pour la partie informatique, 3 au lieu de 5, le projet à pu voir le jour avec succès. Bilan partie réseau Bilan partie front et framework Bilan des aides (cours, site, grâce à quoi, etc...)

Conclusion

Conclusion global, point négatif et positif plus terminer par les perspectives du client.

Partie 6

Bibliographie et Annexes