Rapport final : TriComp

Vendredi 19 Décembre 2014

William Aufort

Julien Bensmail
coordinateur

Agathe Herrou
chef de projet

Romain Labolle

Frédéric Lang

Maxime Lesourd

Laureline Pinault

Léo Stéfanesco

Résumé

Ce document présente le rapport final de notre projet TriComp. Ce projet consiste en le développement d'un logiciel, TriComp, qui permettrait à l'utilisateur d'obtenir les instructions à effectuer pour tricoter l'ouvrage de son souhait, à partir de la description globale de celui-ci.

Nous y détaillons tout le travail que nous avons effectué, les considérations théoriques qui y ont mené, ainsi que les améliorations que nous avons envisagées.

Table des matières

Introduction

Le tricot est un art que l'on imagine souvent réservé aux personnes âgées, mais qui pourtant s'adresse à un public bien plus diversifié.

Aujourd'hui, le tricot est pris d'assaut par des jeunes aussi, parfois dans un mouvement de regain pour le manuel, dû à une ambiance morose de crise financière...les temps sont durs, tricotons.

Mais encore : les londoniens de Knit The City s'en emparent pour décorer la ville à leur manière, des campagnes de dons caritatifs basées sur du tricot se mettent en place (tricoter ensemble un bonnet, puis le revendre pour soutenir une association), de nombreux blogs apparaissent, délivrant conseils et idées de tricot à reprendre et améliorer...on est loin de la grand-mère assise devant sa cheminée !

Partant de là, pourquoi ne pas mettre de l'informatique dans le tricot et compiler nos bonnets ? Puisqu'il existe déjà des machines à tricoter, pourquoi ne pas aller encore plus loin ?

Les magazines de tricot présentent souvent des modèles à réaliser soi-même, combinant différents points. Il est cependant vite fastidieux d'écrire la suite des instructions à suivre pour réaliser une pièce. Notre but ici était donc de réaliser un logiciel permettant d'une part de décrire des pièces de tricot de manière *user-friendly*, et, d'autre part, à partir de cette description, de générer une suite d'instructions à destination du tricoteur permettant de réaliser la pièce en question.

Cette opération de traduction d'un langage de « haut niveau » (l'ensemble des pièces du tricot) vers un langage de « bas niveau » (les instructions) rappelait fortement le principe d'un compilateur, c'est pourquoi nous avons d'abord considéré notre projet comme un « compilateur de tricot ». À l'instar de la compilation d'un langage de programmation informatique, cette compilation de tricot était susceptible de mener à des problèmes théoriques intéressants, à mesure que l'on enrichirait le langage.

Notre projet est donc motivé par interêt pour le tricot d'une part, et notre envie résoudre des problèmes complexes d'autre part.

Nous commencerons donc par présenter le travail réalisé pour mener à bien ce projet, puis nous exposerons des améliorations possibles, et enfin nous évoquerons quelques problèmes théoriques que nous avons pu rencontrer.

Notions de tricot

Le vocabulaire du tricot peut sembler un peu obscur aux non-initiés. Voici donc quelques définitions qui seront utiles pour la compréhension de la suite du rapport :

- une *maille* est l'unité de base d'un tricot. Il s'agit de la boucle qui constitue l'étoffe en étant reliée à ses voisines, horizontalement car partageant la même portion de fil, verticalement car ces boucles sont imbriquées les unes dans les autres. Elle peut être à l'endroit ou à l'envers, selon la manière dont la boucle est réalisée et imbriquée dans la boucle précédente. La figure 1 montre comment sont composées des mailles et les rangs.

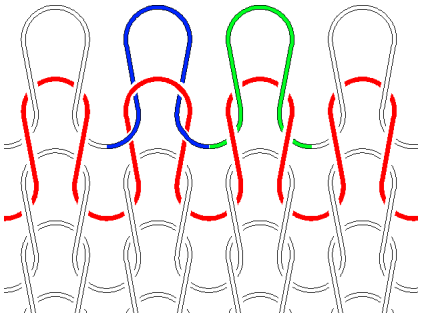

FIGURE 1 – Un schéma sur une partie de tricot, où l'on peut observer des rangs (en rouge), ainsi que des mailles endroit (en vert) et envers (en bleu)

- un *point* est une manière de combiner les différentes opérations réalisables sur les mailles (les tricoter à l'endroit, à l'envers, en tricoter plusieurs ensemble, croiser plusieurs mailles...). Ceci montre que l'on peut obtenir un nombre de motifs possibles extrêmement important.
- l'*ouvrage* désigne la pièce de tricot tout entière, par exemple l'écharpe ou le pull.
- une *diminution* est la suppression d'une maille dans le cours de l'ouvrage ; une technique courante consiste à tricoter ensemble deux mailles. Symétriquement, une *augmentation* est une maille ajoutée dans le cours de l'ouvrage ; une technique courante consiste à enrouler le fil sur l'aiguille pour former une nouvelle maille (on parle alors de *jeté*).

Le tricot est réalisé rang par rang, à l'aide d'aiguilles, qui servent d'une part à maintenir les boucles libres (pour éviter qu'elles ne se défassent et ne libèrent les boucles du rang précédent qu'elles retiennent), d'autre part à former de nouvelles boucles qui deviendront les mailles du rang suivant.

1 Présentation du logiciel

Le nom TriComp de notre logiciel vient de la contraction des mots « tricot » et « compilateur », à juste titre puisqu'il s'agit d'un compilateur de tricot. En effet, le but du logiciel est de transformer une description globale d'un tricot en instructions à suivre par le tricoteur pour le réaliser. Ces instructions sont écrites en langage naturel. Le logiciel se compose de trois parties :

- **Un langage descriptif**, TriLang, qui permet à l'utilisateur de décrire globalement son ouvrage (la forme, la taille, les motifs, ...)
- **Un compilateur** qui effectue la transformation de la description en une liste d'instructions
- **Une interface graphique** qui permet à la fois de visualiser le tricot décrit grâce à Trilang, de faire appel au compilateur, et d'afficher la suite des instructions.

Le logiciel fonctionne de la manière suivante :

- L'utilisateur décrit l'ouvrage qu'il veut réaliser grâce au langage descriptif. Il fait cela dans un nouveau fichier texte, auquel il donne l'extension `.tricot`, et qu'il range dans le dossier de son choix.
- L'utilisateur lance ensuite l'interface graphique. Une fois dans l'interface graphique il peut ouvrir son fichier `.tricot`. Il obtient une visualisation sous la forme du patron de l'ouvrage qu'il a décrit précédemment.
- L'utilsateur peut, toujours dans l'interface graphique, utiliser quelques outils d'éditions qui ont été implémentés (modification du motif d'un ou plusieurs trapèze(s)). Il peut sauvegarder les modifications qu'il effectue.
- Une fois satisfait du modèle qu'il a défini, l'utilisateur peut, toujours dans l'interface graphique, faire appel au compilateur. Celui-ci va tout d'abord effectuer quelques vérifications pour s'assurer que l'ouvrage est tricotable (il va par exemple vérifier qu'il n'y a pas de cycle dans les pièces définies). Il va ensuite générer la liste des instructions à suivre afin de tricoter l'ouvrage. Ces intructions sont affichées dans l'interface graphique, et l'utisateur n'a plus qu'à les suivre pour obtenir son tricot.

La figure 2 montre schématiquement le fonctionnement décrit ci-dessus.

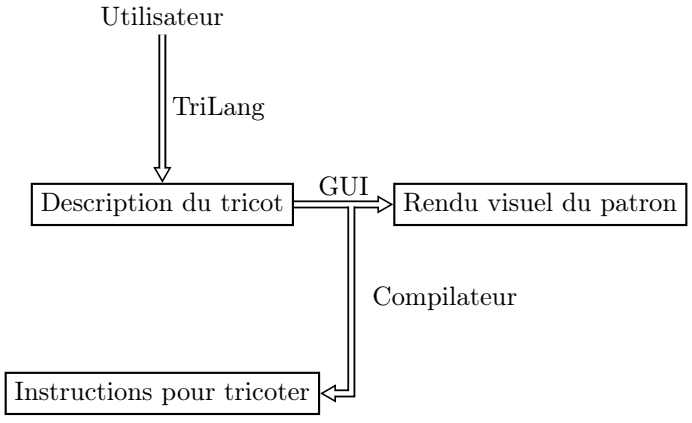

FIGURE 2 – Fonctionnement du logiciel

2 Travail réalisé

Le travail que nous prévoyions de réaliser impliquait d'une part de définir différents langages de « programmation » du tricot (un langage de haut niveau pour décrire de manière rigoureuse les ouvrages, et un langage de bas niveau correspondant aux mailles à tricoter), d'autre part d'écrire un compilateur réalisant la traduction d'un langage vers l'autre, et enfin de réaliser une interface graphique permettant de les manipuler intuitivement et de visualiser les pièces en cours de définition. Nous détaillons dans cette partie le travail qui a été réalisé dans chacun de ces objectifs. Nous rappelons que tout le travail (logiciel, documents et site web) sont disponibles et consultables sur le dépôt Git du projet TriComp (https://github.com/TriComp/).

2.1 Définition des langages

Nous avons été amenés au cours de ce projet à définir deux langages, l'un purement statique, puisque servant à décrire les ouvrages de tricot, l'autre dyamique, car décrivant le processus de réalisation du tricot. Nous les détaillons ici.

2.1.1 Langage descriptif

Une première version du langage descriptif avait été détaillée dans le rapport de mi-parcours. Quelques modifications ont été effectuées depuis. Nous exposerons donc ici le langage final, les modifications effectuées ainsi que leur intêret, le tout illustré par un exemple.

L'idée générale est de concevoir un langage capable décrire un tricot de la façon la plus globale possible. Dans les (rares) logiciels de tricot existants, le tricot est décomposé en sa plus petite entité : la maille (cf figure 3). L'avantage de cette décomposition atomique est que l'on peut réaliser des choix de points (voire de couleurs) avec une très bonne précision. Son gros inconvénient est qu'il est souvent pénible de travailler avec cette représentation trop précise, car les tricots ont en général une taille assez importante. Notre démarche, quant à elle, propose une vision du tricot plus globale, et donc beaucoup plus simple à manipuler.

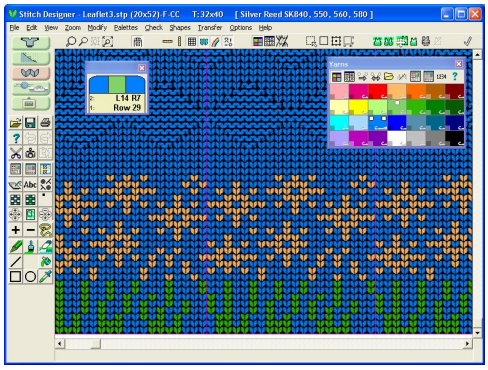

FIGURE 3 – Le logiciel Stitch Designer propose une vue très précise du tricot, mais celui-ci a donc une taille beaucoup trop importante (comme en témoignent les barres de défilement à droite et en bas de la fenêtre)

L'élément de base dans ce langage est le trapèze. Un trapèze définit une zone où l'on trouve un même motif. Un trapèze est défini par sa hauteur, le décalage de sa base supérieure, et la longueur de ses bases (voir figure 4). Ce choix se justifie par le fait que le trapèze est l'unité atomique la plus générale possible dans un ouvrage. En effet, comme le tricot est réalisé rang après rang, la pièce tricotée d'un tenant a deux côté parallèles. En revanche, grâce aux augmentations et aux diminutions les deux

autres cotés ne sont pas forcément parallèles. Par l'assemblage de trapèzes, nous arrivons à obtenir n'importe quelle pièce tricotable.

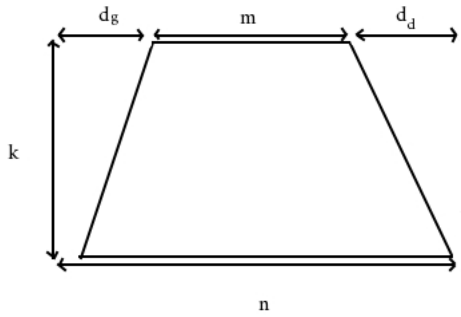

FIGURE 4 – Les paramètres d'un trapèze que l'on utilise dans la description

Ainsi, un tricot est subdivisé en trapèzes qui sont reliés les uns les autres d'une certaine manière : un trapèze connaît l'entité qui le suit directement (dans le parcours du tricot) via un pointeur. Il se peut que plusieurs trapèzes suivent un même trapèze : dans ce cas, on ajoute des aiguilles afin de pouvoir continuer à tricoter les deux branches indépendamment. Cette situation se modélise dans le langage via le mot-clé `split`. Inversement, il se peut que plusieurs trapèzes aient le même successeur, c'est à ce moment là que plusieurs morceaux peuvent être réunis afin de libérer une ou plusieurs aiguilles : c'est un `link`.

Notre format de fichier nous autorise à définir différentes pièces avec différents noms. Ces pièces permettent de décomposer une pièce plus importante, notamment en cas de présence de `link`, qui prennent en entrée le nom de la pièce suivante.

La figure 5 illustre et résume les possibilités du langage, via une description d'un poncho.

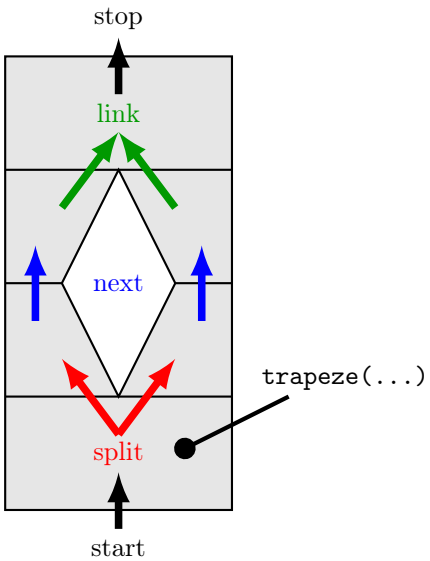

FIGURE 5 – Les possibilités du langage illustrées sur un tricot (en l'occurence un poncho). Une attention toute particulière est portée aux jonctions entre les différents trapèzes.

Comme nous le disions précédemment, quelques modifications ont été apportées au niveau de ce

langage depuis le rapport de mi-parcours. Deux choses importantes ont été modifiées :

- La largeur inférieure des trapèzes est cette fois-ci déduite de la largeur supérieure du trapèze précédent, et ne figure donc plus dans les paramètres du trapèze. En revanche, `start` et `split` prennent en argument la largeur inférieure du ou des trapèze(s) qui les suivent. Cette modification permet d'éviter des redondances sans se compliquer la tâche.
- Les `split` peuvent générer plus de deux pièces différentes. Par conséquent, les `link` ont également été modifiés : ils se lient avec leur successeur via le nom du successeur, suivi de la position sur ce successeur (alors qu'avant, un simple positionnement gauche/droite suffisait). Nous avons effectué cette modification car certains tricots (par exemple une salopette) ne pouvait être écrit de façon simple avec le précédent langage. Ceci est du au fait que précédemment, le positionnement des pièces après un `split` était déterministe (une pièce à gauche, une pièce à droite) et donc restrictif. Cette partie a apporté beaucoup plus de modifications dans le code que la modification précédente.

Au final nous obtenons un langage permettant de représenter une quantité importante de tricots d'un point de vue assez original par rapport à ce qui est fait dans les autres logiciels de tricot. Les tricots qui suivent notre langage portent l'extension `.tricot`.

À titre d'exemple, voici le code décrivant une petite écharpe faite uniquement avec du point mousse :

```
Name : toy_scarf
Description : "The easiest knit you can make"
piece my_piece := start 20
|| trapezoid (height : 60, shift : 0, upper_width : 20, pattern : point_mousse)
|| stop
```

Un exemple plus complexe se trouve en annexe. De plus, la documentation complète du langage est disponible sur le site de TriComp (http://tricomp.github.io).

2.1.2 Langage de bas niveau

Le *langage de bas niveau* correspond aux instructions que doit suivre l'utilisateur pour pouvoir réaliser son tricot. Ces instructions sont produites par le compilateur, dont on détaillera le fonctionnement dans la prochaine partie. Ici nous présentons le format (ou langage) utilisé pour présenter ces instructions, proches de ce que l'on peut trouver dans un manuel de tricot.

Ce langage doit refléter exactement les mêmes informations que celles que l'on pourrait trouver dans un manuel de tricot : les points à tricoter. Dans les manuels, la périodicité des motifs est exploitée afin de ne pas donner une instruction pour chaque maille. Cette périodicité se traduit à la fois sur un rang (horizontalement, voir la figure 6) et sur un ensemble de rangs (verticalement, voir la figure 7).

```
Ligne 1 : 140 fois 1 maille endroit, 1 maille envers
```

FIGURE 6 – Ici, on repète une succession de points plusieurs fois sur un même rang.

```
Répetez 245 fois le motif suivant :
Ligne 1 : 140 fois 1 maille endroit, 1 maille envers
Ligne 2 : 140 fois 1 maille envers, 1 maille endroit
```

FIGURE 7 – Ici, on repète une succession de rangs plusieurs fois (verticalement).

Les deux figures précédentes illustrent des instructions que nous pouvons obtenir avec notre logiciel. Dans un manuel de tricot, elles seraient volontairement moins verbeuses, à cause du nombre important de motifs différents à décrire que contiennent ces livres.

Mais tout tricot ne peut être construit uniquement à partir d'instructions de ce type. En effet, nous avons vu qu'avec les `split` et les `link` on peut avoir à ajouter (ou enlever) une ou plusieurs aiguilles, et cette information doit figurer parmi les instructions renvoyées.

Lors d'un `split`, le tricoteur laisse de côté une partie de son ouvrage sur une aiguille et prend une autre aiguille afin de tricoter une des branches du split. Les intructions alors données à l'utilsateur sont les suivantes :

- On donne à l'utilisateur les instructions pour tricoter la premières branche après le split.
- Une fois cette branche terminée on demande à l'utilsateur, selon les cas, de fermer les mailles correspondant à cette branche ou de laisser son ouvrage sur l'aiguille.
- On donne les instructions pour la branche suivante.
- Et ainsi de suite.

Lors d'un `link`, le tricoteur doit assembler, c'est à dire tricoter à la suite avec le même fil, des partie d'ouvrages ayant précédemment été laissée de côté sur des aiguilles. L'instruction correspondante correspond tout simplement à dire à l'utilsateur d'« assembler les dépendances ». En effet, l'utilisateur ayant sous les yeux le patron du tricot qu'il réalise, il ne peut y avoir d'ambiguïté.

Enfin, la dernière information importante concerne le début et la fin du tricot. En effet, pour commencer un tricot, le tricoteur doit *monter des mailles* (c'est-à-dire créer une première rangée de mailles sur son aiguille) tandis qu'à la fin il doit les fermer.

Les premières mailles sont particulières, car elles ne reposent pas sur des mailles précédemment tricotées. Nous indiquons donc au début du tricot le nombre de mailles à monter. Nous indiquons également les mailles à fermer quand besoin est. Ces étapes ne figurent pas dans les manuels de tricot, car ceux-ci traitent souvent de motifs, et non de pièces.

Ainsi, les instructions que renvoie notre logiciel sont à la fois précises comme celles d'un manuel, mais également claires et faciles à suivre pour un être humain ayant des connaissances basiques en tricot.

2.2 Compilateur

Le rôle du compilateur est, à partir du fichier `.tricot`, d'une part de s'assurer qu'il décrit un tricot possible (par exemple qu'il ne contient pas de cycle, que tous les `link` ont un sens, etc ...), et d'autre part de générer des instructions dans le langage de bas niveau.

Il est implémenté en OCaml, et il utilise la bibliothèque standard alternative Core. L'analyse syntaxique utilise le générateur d'analyseur syntaxique Menhir, et est robuste : il émet des messages d'erreurs précis lorsque le fichier d'entrée n'est pas valide.

Pour ce faire, il créé un graphe de dépendance entre les différentes pièces du tricot : un arc dénote qu'il faut tricoter une pièce avant l'autre. On peut alors utiliser des algorithmes de graphes de la littérature, par exemple pour la détection de cycle.

2.2.1 Vérifications

Avant de générer les instructions, le compilateur s'assure que le tricot passé en entrée est tricotable. Il vérifie les points suivants :

- Que chaque pièce est tricotable. Pour cela il vérifie qu'il n'y a pas de recouvrement lors des `split` et des `links`. En effet, lorsqu'on effectue un `split` ou un `link`, on indique le positionnement absolu des accroches. La syntaxe du langage n'interdit donc pas, a priori, de définir deux trapèzes partant des mêmes mailles depuis le trapèze précédent (dans le cas d'un `split`) ou un trapèze

s'accrochant en une maille aux mailles de deux trapèzes différents (dans le cas d'un `link`). Or ces deux cas ne sont pas possibles en réalité. Pour détecter ces configurations impossible, le compilateur calcule tout simplement les mailles auxquelles s'accrochent les trapèzes grâce aux positions et largeurs passées en paramètres, et vérifie qu'il n'y a pas de recouvrement.
- Qu'il n'y a pas de cycle dans les pièces à tricoter, et donc que l'ensemble de l'ouvrage est tricotable. Il s'assure ainsi que le tricot commence quelque part et finit quelque part. Pour cela il vérifie qu'il n'y a pas de cycle dans le graphe de dépendance.

Exemple d'instruction rendant une pièce non tricotable

Supposons que l'instruction suivante se trouve dans la description d'un pièce :

```
|| split
  10 20 { trapezoid ( height : 50, shift : 0, upper_width : 20, pattern : jersey ) }
  20 30 { trapezoid ( height : 50, shift : 0, upper_width : 30, pattern : jersey ) }
```

Cette intruction signifie que le trapèze courant pointe vers un trapèze qui s'accroche à la maille 10 sur une largeur de 20 mailles et vers un trapèze qui s'accroche à la maille 20 sur une largeur de 30 mailles. Ceci n'est pas réalisable en tricot, car sur les mailles 30 à 40 il faudrait pouvoir tricoter à la fois le premier et le deuxième trapèze.

Le compilateur va détecter ce recouvrement par le calcul et renvoyer une erreur.

2.2.2 Parcours du tricot

Pour la génération des instructions bas niveau, on parcourt le graphe dans un ordre topologique, qui de plus est adapté au tricot.

Le parcours du tricot utilise l'algorithme 1, qui est un algorithme de dataflow à work list, très courant en compilation. Ici le workset est implémenté à l'aide d'une pile, ce qui nous permet de parcourir complètement une branche de tricot avant de s'occuper d'une autre. Ainsi on ne demande pas au tricoteur de faire un va-et-vient entre plusieurs aiguilles si ce n'est pas nécessaire.

Algorithm 1 Algorithme de parcours du tricot

```
1: WorkSet ← racines du graphe.
2: Done ← (λn → ∅) // Un dictionnaire
3: while WorkList ≠ ∅ do
4:     x ← Pop (WorkList)
5:     Imprimer les instructions pour x
6:     for y ∈ Successors(x) do
7:         Done(y) ← Done(y) ∪ { x }
8:         if Done(y) = Predecessors(y) then // On a parcouru toutes les dépendances de x
9:             WorkSet ← WorkSet ∪ { y }
```

2.3 Interface graphique

Nous exposons dans cette partie le travail qui a été fourni concernant l'élaboration de l'interface graphique, notamment les caractéristiques de l'interface, les fonctionnalités ainsi que des détails d'implémentation.

2.3.1 Caractéristiques et fonctionnalités

Avant toute chose, il nous fallait cibler les caractéstiques que devait avoir l'interface. Destinée à être utilisée par des tricoteurs, celle-ci devait être facile d'utilisation. L'utilisateur dispose des outils indispensables à toute interface (ouverture, sauvegarde de fichiers, zoom...) mais également d'outils propres au tricot et aux objectifs de TriComp : un bouton pour générer les instructions, ainsi que quelques outils d'édition de tricot (choix des points sur les trapèzes).

Le choix du motif à mettre sur chaque trapèze peut être fait parmi une dizaine de points disponibles via des boutons. Une amélioration future pourrait être de permettre à l'utilisateur de définir ses propres points.

L'interface est divisée en trois parties, comme on peut le voir sur la figure 8 : une partie contenant les outils d'édition de tricot, une fenêtre où est affiché le tricot, et une zone où se trouve la liste des instructions générées par le compilateur.

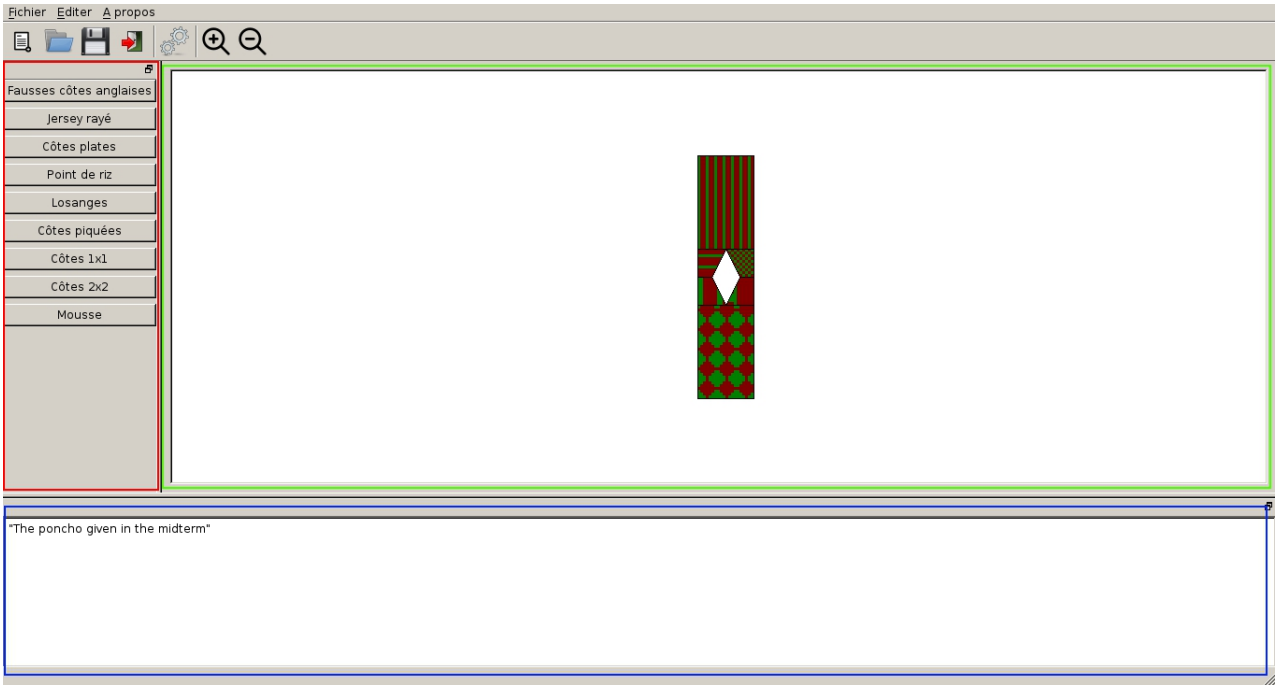

FIGURE 8 – L'interface graphique de TriComp. En rouge, la zone de choix des points, en vert, la zone de visualisation du tricot en cours, en bleu, la zone où s'affichent les instructions.

2.3.2 Détails d'implémentation

L'interface a été créée à l'aide de la bibliothèque Qt. Une grande partie du projet a été intégrée au sein de la partie Qt, pour des questions de simplicité au niveau de l'intégration. Seule la partie compilation (écrite en OCaml) est dissociée de la partie Qt, et est appelée dans l'interface via des appels systèmes.

Détailler l'ensemble des structures offertes par Qt utilisées dans le projet serait ici long et inutile, le lecteur pourra se référer au code disponible sur la page Github [1]. On mentionnera juste que pour l'affichage, on ne travaille pas directement sur le tricot en temps qu'objet, mais sur une représentation du tricot formée d'`items` (qui forme, en quelque sorte, une couche supérieure). Même si les deux représentations sont isomorphes (grossièrement un `item` pour un trapèze du tricot), cela permet d'éditer autant qu'on le veut sans modifier l'objet de base (sauf lors d'une sauvegarde).

2.4 Communication

Nous détaillons dans cette partie la partie communication du projet, qui englobe la communication interne et externe (via le site web).

2.4.1 Site Web

Un site web a été déployé à l'adresse http://tricomp.github.io, il est hébergé sur Github (utilisé comme serveur Git du projet). Le site est basé sur Jekyll, un CMS en Ruby spécialisé dans les blogs et qui a la particularité de ne pas utiliser de base de données, avec l'avantage que Github gère Jekyll automatiquement. Le site est notamment une vitrine pour le projet : il en contient une présentation rapide, avec des liens vers le code source sur Github, ainsi que différents articles autour du projet. La page dispose également d'une page pour aiguiller rapidement l'internaute anglophone.

2.4.2 Communication au sein du projet

La communication au sein du projet s'est principalement effectuée par le biais de notre liste de diffusion (tricomp [at] ens-lyon.fr). L'échange des idées au début du projet s'est fait par l'intermédiaire d'un pad hébergé par l'association AliENS (http://pad.aliens-lyon.fr/p/tricomp). Nous nous réunissions également toutes les semaines avec notre encadrant afin de faire le point sur l'avancement du projet, de discuter de choix (tels que la syntaxe du langage descriptif, la modélisation à adopter pour les points, les fonctionnalités du logiciel final...), mais également au cours de séances non officielles où certains groupes se réunissaient afin de résoudre certaines problématiques un peu plus coriaces (notamment par rapport au langage, au compilateur ou à la gestion de l'éditeur de l'interface).

3 Améliorations possibles

Le manque de temps et de moyens humains nous a empêché de mettre en place la totalité des fonctionnalités que nous avions prévues. Cependant, ce projet porte en lui un potentiel certain ; nous détaillons ici quelques améliorations auxquelles nous avons songées, sachant qu'elles verront peut-être le jour (certains membres de l'équipe étant intéressés par la poursuite du projet).

3.1 Langage descriptif

En ayant plus de temps à notre disposition, nous pourrions enrichir le langage de manière à permettre une personnalisation plus importante des motifs. Ceci pourrait se faire sous deux angles différents et complémentaires :

- Une idée intéressante serait de laisser l'utilisateur définir ses propres points, en plus de ceux que nous avons prédéfinis. En effet, les possibilités de points sont infinies, et l'utilisateur pourrait ainsi exprimer toute sa créativité.
- Une autre idée serait de permettre à l'utilisateur de définir des zones dans son ouvrage où tel motif serait appliqué, au lieu de ne permettre le changement de motif que lors d'un changement de trapèze. En effet, notre langage tel quel ne permet pas par exemple de définir un pull dont la moitié gauche serait en point mousse et la moitié droite en jersey : la séparation entre les motifs

1. Rappel : https://github.com/TriComp/

se fait forcément en bandes horizontales. Bien qu'en soit pas très compliqué à mettre en oeuvre avec un repérage absolu, la définition des zones entraînerait des vérifications supplémentaires à effectuer auxquelles il faut bien penser.

De même, les tresses n'ont pas été intégrées au langage. Elles pourraient être intégrées dans un futur relativement proche en utilisant un paradigme similaire à celui utilisé pour décrire les points :

- Il faudrait définir un format décrivant une tresse. Par exemple donner le nombre de brins de la tresse, la largeur de ces brins, et une description de la manière dont ces brins se croisent sous forme d'un motif minimal à répéter. De même que pour les points, l'idéal serait à terme que l'utilisateur puisse définir ses propres tresses.
- On pourrai se servir également de la définition de zones dans le tricot afin de décrire où positionner les tresses.

En outre, il pourrait être intéressant d'inclure dans le langage une description des coutures à effectuer afin d'assembler le tricot à la fin. En effet, bien que cette étape ne relève pas du tricot en tant que telle (et c'est pour cette raison qu'elle n'a pas été traitée jusqu'à présent), elle permettrait à long terme l'échange de modèles de tricots sous forme de fichier `.tricot` entre les tricoteurs.

3.2 Compilateur

Le compilateur actuel ne gère que les rectangles. La gestion des trapèzes (l'idée centrale du langage) serait un des premiers points sur lequel nous pourrions nous pencher de nouveau. A travers cette question se pose notamment les problèmes dela gestion des augmentations et des diminutions, notamment pour des motifs complexes. Nous avons déjà réfléchi à un algorithme pour répartir ces augmentations et diminutions. Celui-ci se trouve dans la section 4.

3.3 Interface graphique et édition

Du côté de l'interface, une grande partie du travail que nous n'avons pas eu le temps d'aborder portait sur l'édition des tricots. En effet pour l'instant, le seul outil d'édition qui a été implémenté est celui qui permet de changer le motif des trapèzes. Nous voudrions étoffer ces outils d'éditions, permettant à l'utilisateur par exemple de chager la taille des pièces de son tricot. L'idée serait qu'à terme l'utilsateur puisse entièrement décrire son tricot dans l'interface graphique sans avoir à passer par le langage que nous avons défini (qui peut, bien qu'il soit facile de prise en main, rebuter certains des utilisateurs).

Par ailleurs d'autres améliorations plus mineures (afficher le nom des pièces sur le patron pour faciliter la lecture des instructions par exemple) peuvent être envisagées.

3.4 Interfaçage avec des machines à tricoter

L'intégration de machines à tricoter dans notre projet avait été écartée dès le début du projet, d'une part parce qu'aucun membre de l'équipe n'avait les compétences en électronique requises pour un tel travail, d'autre part parce que nous savions que le temps nous manquerait pour nous en occuper.

Cependant, cela semble un prolongement raisonnable de notre projet, dans le sens où cela pousserait l'automatisation du tricot jusque dans sa réalisation. Il nous faudrait pour cela tout d'abord nous associer avec des personnes ayant de l'expérience dans le domaine des machines à tricoter, puis travailler à adapter les instructions bas niveau à l'utilisation par une machine à tricoter (une différence majeure entre un tricoteur humain et une machine à tricoter étant que la machine à tricoter tricote toutes les mailles d'un rang en parallèle, tandis que le tricoteur les tricotera de manière séquentielle).

4 Résultats théoriques

4.1 Algorithmes de répartition des diminutions

Rappels Une *augmentation* est une opération consistant à ajouter une maille au rang en cours : ainsi, à la suite d'une augmentation, ce rang comportera une maille de plus que le rang précédent, la largeur du tricot aura donc augmenté.

Une *diminution* est l'opération inverse d'une augmentation : on supprime une maille au rang en cours, et on fait donc diminuer la largeur du tricot.

Réalisation des trapèzes Dans l'optique d'intégrer des trapèzes non rectangulaires au langage, nous nous sommes penchés sur la manière de répartir des diminutions et augmentations au cours des rangs, de manière à ce que les côtés apparaissent rectilignes. Nous nous sommes donc penchés du côté des algorithmes d'image, permettant de tracer une ligne apparaissant la plus droite possible en utilisant des pixels discrets.

Cet élément du langage n'a pas encore été implémenté ; cependant, en prévision de l'intégration des diminutions et augmentations au langage, nous avons fait quelques recherches, et avons décidé d'utiliser l'algorithme de Bresenham [2].

4.2 Allocation d'aiguilles

Lors de la génération des instructions, on souhaiterait déterminer le nombre d'aiguilles nécessaires à la réalisation du tricot, et si possible le minimiser. Ce problème ressemble fort à celui de l'allocation de registres en compilation, et on peut montrer qu'il lui est en fait équivalent.

En effet, faisons l'analogie suivante : on fait correspondre les aiguilles supplémentaires (par rapport aux 2 aiguilles de base) aux temporaires dans lesquels stocker les variables, et les variables aux zones où le tricot est séparé en deux, introduites par les `split` (en gardant en tête qu'on peut évidemment effectuer un `split` sur une branche déjà issue d'un `split`). Ainsi, on peut réaliser la même réduction de ce problème depuis la coloration de graphe qu'à celui de l'allocation de temporaires.

Cela nous permet de montrer que le problème de l'allocation d'aiguilles est NP-complet ; il est donc fort probable que nous devions utiliser des heuristiques pour déterminer le nombre d'aiguilles à prévoir pour réaliser un tricot. En pratique, ce nombre est rarement significativement élevé, il est donc probable que ces heuristiques soient suffisants.

Conclusion

Au terme des trois mois qui nous étaient donnés pour développer le logiciel, nous avons créé un langage robuste, ainsi qu'un logiciel fonctionnel sur ses différentes parties (compilateur, interface). Le logiciel reste incomplet par rapport aux fonctionnalités que nous voulions développer en début de projet (voir la proposal pour plus de détails). Cependant, la partie implémentée répond aux objectifs primaires que nous avions ciblés : la définition d'un langage pour descrir globalement des tricots, l'édition (au moins partielle) de ces tricots et la génération d'instructions associées via une interface graphique.

Remerciements

Merci à Guillaume de nous avoir soufflé l'idée d'un compilateur de tricot, et à Oriane pour ses précieux conseils de communication !

2. http://fr.wikipedia.org/wiki/Algorithme_de_trac%C3%A9_de_segment_de_Bresenham

A Exemples de codes sources

A.1 Exemple d'un ouvrage décrit avec le langage descriptif TriLang

Nous donnons dans cette section un exemple d'ouvrage décrit dans le langage descriptif que nous avons détaillé précédemment, dans un fichier avec l'extension `.tricot`. Il s'agit d'un poncho.

```
Name : ponzo
Description : "Un magnifique exemple de pull sans manche, autrement appelé poncho"

piece my_piece := start 840 || trapezoid ( height : 200, shift : 0, upper_width : 840,
      pattern : cotes_plates)
 || trapezoid ( height : 840, shift : 0, upper_width : 840, pattern : cotes_1x1)
 || split 0 280 {   trapezoid ( height : 490, shift : 0, upper_width : 280,
         pattern : cotes_1x1)
 || link top 0 } 560 280 {   trapezoid ( height : 490, shift : 0, upper_width : 280,
         pattern : cotes_plates)
 || link top 560 }

piece top := start 840 || trapezoid ( height : 840, shift : 0, upper_width : 840,
      pattern : cotes_1x1)
 || trapezoid ( height : 210, shift : 0, upper_width : 840, pattern : cotes_plates)
 || stop
```

Voici le patron correspondant à ce code (le patron a été obtenu avec l'interface graphique) :

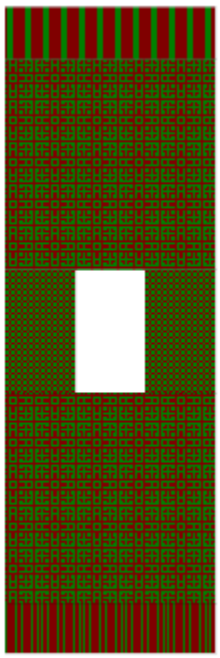

FIGURE 9 – Le poncho tel que l'interface graphique le visualise

A.2 Instructions en sortie du compilateur

Lorsqu'on compile le code donné précédemment, on obtient les instructions suivantes :

```
Instructions pour "ponzo":

Début de la pièce "my\_piece":
Montez 840 mailles.

Répetez 100 fois le motif suivant :
Ligne 1 : 105 fois 5 mailles envers, 3 mailles endroit
Ligne 2 : 105 fois 3 mailles endroit, 5 mailles envers


Répetez 420 fois le motif suivant :
Ligne 1 : 420 fois 1 maille endroit, 1 maille envers
Ligne 2 : 420 fois 1 maille endroit, 1 maille envers


Répetez 245 fois le motif suivant :
Ligne 1 : 35 fois 5 mailles envers, 3 mailles endroit
Ligne 2 : 35 fois 3 mailles endroit, 5 mailles envers

Laissez l'aiguille de côté pour le moment.
Fermez 280 mailles

Répetez 245 fois le motif suivant :
Ligne 1 : 35 fois 5 mailles envers, 3 mailles endroit
Ligne 2 : 35 fois 3 mailles endroit, 5 mailles envers


Début de la pièce "top":
Assemblez les dépendances.


Répetez 420 fois le motif suivant :
Ligne 1 : 420 fois 1 maille endroit, 1 maille envers
Ligne 2 : 420 fois 1 maille endroit, 1 maille envers


Répetez 105 fois le motif suivant :
Ligne 1 : 105 fois 5 mailles envers, 3 mailles endroit
Ligne 2 : 105 fois 3 mailles endroit, 5 mailles envers


Félicitations, vous avez fini!
```