Proposition de projet : Origram

09/10/2013

Grégoire Beaudoire
Baptiste Jonglez
Fabrice Mouhartem, coordinateur
Pierre Pradic
Armaël Guéneau
Jérémy Ledent
Antoine Pouille
Damien Rouhling

Table des matières

1 Introduction et objectifs

L'art de l'origami est très ancien, et pourtant ce domaine a peu fait usage des outils modernes offerts par les avancées de l'informatique.

Il a toujours été difficile de « transmettre » l'information en origami. En effet, même s'il y a bien sûr la méthode directe (d'enseignant à élève), transmettre une méthode de pliage à grande échelle reste un exercice délicat. Comment communiquer le cheminement de plis nécessaire pour arriver au résultat final ?

L'une des méthodes les plus faciles à lire et à comprendre pour un plieur est la découpe en *diagramme* : le pliage est découpé en étapes, à l'image d'une notice de montage de meuble, et chaque étape est agrémentée d'un dessin en trois dimensions. Cela permet au plieur de visualiser le cheminement à suivre. Il existe d'autres méthodes, mais beaucoup moins faciles à lire, et donc moins efficaces pour partager les connaissances d'origami.

A l'heure actuelle, quand un créateur veut partager son travail — sa création — et l'apprendre ainsi à tous les potentiels plieurs dans le monde, il ne dispose pas de moyens automatisés pour l'aider. Ainsi, il peut, soit :

- ne pas partager la construction, ce qui est clairement le plus rapide mais aussi le plus regrettable d'un point de vue de la connaissance globale ;
- partager en utilisant des moyens de communication peu lisibles — par exemple via un *canevas de plis*, c'est-à-dire en dépliant complètement l'origami et exposant les plis, sans indiquer aucunement la méthode ou l'ordre suivi pour les faire ;
- réaliser un diagramme *manuellement* : avec du papier et un crayon, ou bien un logiciel de dessin vectoriel comme *Inkscape*. Cela prend un temps non négligeable : à titre d'exemple, pour un modèle relativement simple d'une trentaine d'étapes, une semaine de travail est nécessaire.

Il y a ainsi un important volume d'origamis dont la construction est inconnue sauf pour le créateur du pliage, et un nombre plus important encore dont la portée est très limitée car ils nécessitent, pour arriver à reproduire le pli, des connaissances techniques très poussées, afin d'être capable de lire et décoder la construction, connaissances dont ne disposent pas la plupart des origamistes. Cela met également un frein à l'expansion de cet art, étant donné d'une part le manque de données informatisées, et d'autre part le manque de pliages couverts par les diagrammes.

Notre objectif est de répondre à ce manque dans le domaine de l'origami et de parvenir à fournir aux créateurs et aux plieurs un outil pour leur permettre de communiquer et d'échanger des connaissances plus facilement. En termes plus précis, nous voulons créer un logiciel — Origram — permettant d'une part l'aide à la construction de diagrammes, et d'autre part l'aide à la création. L'objectif est de rendre plus facile et plus rapide pour les créateurs la construction de diagrammes, facilitant ainsi la communication par diagrammes. Les plieurs de tous niveaux pourraient également l'utiliser pour « essayer » des pliages, ce qui faciliterait tant la création originale de nouveaux plis que l'accès à l'origami pour les débutants, qui ne seraient plus tributaires d'un accès à un club d'origami, mais pourront apprendre sur internet.

2 Description de l'existant

Deux catégories peuvent être distinguées dans l'état de l'art actuel de l'informatique de l'origami : les logiciels aidant à la conception de diagrammes, et les logiciels orientés vers la création de l'origami. Nous allons en observer les caractéristiques dans cette partie afin de pouvoir en dégager ce qui pourrait être réutilisé dans notre projet.

2.1 Dans la conception de diagramme

Il y a actuellement différentes manières d'aborder ce problème : *Doodle*[3] est un langage de description générant un diagramme ; *Origami simulator and diagrammer*[6] suit une approche identique à la nôtre mais peu ergonomique et dont résulte un diagramme peu lisible. Enfin, on a avec [13] une tentative d'application web — avec l'idée de toucher un large public — inactive aujourd'hui.

2.1.1 Doodle

Doodle est un langage destiné à décrire un diagramme, et permet de produire des diagrammes esthétiques et lisibles.

Ce langage se fonde sur des règles appliquées séquentiellement pour décrire l'avancement du diagramme.

Cependant ces règles sont majoritairement des primitives de dessin, permettant d'indiquer sur quelle ligne marquer un pli (comme les règles sont incrémentales, ces lignes restent dessinées, ce qui permet au concepteur du diagramme de ne pas avoir à les redessiner à chaque étape). Toutefois, dès que l'on décide de faire un pli, on est obligé d'indiquer manuellement comment retracer le nouvel état de l'origami, ce

qui ne rend pas forcément la tâche très aisée et accessible. Pour avoir un ordre d'idée, un diagramme de 9 étapes se fait en 200 instructions [1]. La tâche reste donc fastidieuse.

2.1.2 Origami simulator and diagrammer

Ce programme a été conçu dans le cadre d'un mémoire[7] pour savoir si un diagramme pouvait être généré informatiquement ou non. Il a donc été initialement créé pour fournir des résultats, plutôt que dans le but de s'adresser aux concepteurs d'origami.

On retrouve des fonctionnalités similaires à notre proposition de projet :

- pliage à la souris, avec points de référence pour aider à aligner les bords ;
- fenêtre 3D dans laquelle il est possible de faire tourner le pliage ;
- animation montrant les étapes de pliage ;
- production d'un diagramme en POSTSCRIPT.

Cependant, de nombreux défauts rendent le programme inutilisable en pratique. L'interface est mal pensée et peu intuitive, en plus d'être graphiquement très pauvre. Les diagrammes obtenus sont souvent incomplets, et ne sont pas suffisamment clair pour être utilisables par un origamiste. Enfin, le programme utilise des bibliothèques spécifiques à une plate-forme (`win32` et `DirectX`), les sources ne sont pas disponibles, et il n'est plus maintenu depuis 2005.

Nous pouvons cependant utiliser le manuscrit du mémoire afin d'étudier les techniques utilisées, ce qui peut nous aider pour l'implémentation.

2.1.3 Foldinator

Il s'agit d'un projet prometteur en flash développé en 2009 : un simulateur-diagrammeur, mais dont la version actuelle ne fonctionne plus. Il est possible qu'une mise à jour du plugin flash ait nui à la rétro-compatibilité , le programme ne réagissant pas aux entrées.

En revanche, d'après les démonstrations données sur le site du projet, il semblerait que les diagrammes obtenus, ainsi que l'interface utilisateur soient clairs et esthétiques. Le programme étant disponible sous la forme d'une application web, il serait également facile d'accès.

Nous pouvons néanmoins signaler que dans le cadre d'un diagramme compliqué ou non traditionnel (les pliages faisant intervenir par exemple des grilles, ne serait-ce que dans une moindre mesure), le programme montre ses limites, d'une part par les risques inhérents aux programmes web (saut de connexion, fermeture du navigateur pendant un pliage qui nécessite un minimum de temps), ou par le manque de précision au niveau du pliage (pour revenir aux cas des grilles parallèles, c'est le manque de repères qui fait défaut). Il est d'ailleurs difficile de trouver des traces d'utilisation de cet outil, et nous ne pouvons pas savoir comment il se comporterait face à des modèles complexes. Par ailleurs, nous ne disposons pas des sources du projet, ni de quelconques spécifications.

Contrairement au logiciel précédent dont le but était de soutenir une thèse, ce logiciel avait pour public les origamistes, ce qui montre que notre projet se fonde sur une idée qui pourrait intéresser de nombreuses personnes.

2.2 Dans l'aide à la création

Nous disposons actuellement de plusieurs outils destinés à aider le créateur, principalement développés par Robert Lang, et que nous allons exposer ici, puisqu'ils font intervenir des techniques qui pourraient être réutilisées dans le cadre de notre projet. Nous allons ici présenter *ReferenceFinder*[8], *TreeMaker*[8], *Oripa*[12] et un projet en cours de développement : *BPMaker*[4]

2.2.1 ReferenceFinder

ReferenceFinder est un programme destiné à générer une séquence de plis afin de déterminer un point de repère.

Nous y définissons un point de repère par ses coordonnées sur la feuille. Ces points sont utiles en origami dit « traditionnel », où les points peuvent difficilement être déplacés sur le canevas de plis puisque le déplacement d'un point entraine le déplacement des autres points du modèle.

1. http://doodle.sourceforge.net/resources.html : pajarita.

La méthode utilisée par *ReferenceFinder* se base sur des coordonnées flottantes, ce qui donne en fait une approximation du point demandé. L'avantage par rapport à une méthode exacte est qu'on obtient ainsi une séquence de plis relativement courte, qui permet d'éviter les plis de construction superflus.

2.2.2 TreeMaker

Il s'agit d'un programme développé par Robert Lang dont le principe est de partir d'un arbre, c'est-à-dire une représentation d'une base expliquant la taille des volets indépendant de la feuille ainsi que leur disposition, pour construire le canevas de plis de la base associée.

Le fonctionnement de ce programme utilise la méthode du remplissage de cercles, qui est une méthode géométrique afin de déterminer un remplissage maximal de la feuille par des polygones, on utilise ensuite des molécules pour les remplir.

Il serait possible que nous récupérions l'algorithme de construction de la base pour trouver un moyen de la plier : le pliage des molécules pouvant être calculé, il suffirait de générer la séquence de plis adaptés. Ce qui permettrait de prendre un raccourci non négligeable dans la construction du diagramme, et de rendre la technique de l'arbre plus utilisée en origami (la génération de canevas de plis rebutant souvent certains créateurs).

2.2.3 Oripa

Oripa est un éditeur de canevas de plis calculant pour les origamis les plus simples un pliage en deux dimension — s'il existe — .

Il est utilisé par la communauté lorsqu'il s'agit de construire des canevas de plis à distribuer puisqu'il propose une interface facile d'accès et relativement complète, permettant de vérifier qu'il n'y a pas d'erreurs en simulant le pliage final.

Nous pourrions donc nous inspirer de l'ergonomie de l'interface d'oripa dans le cadre de notre projet, et examiner comment le plieur 2D fonctionne afin de l'appliquer sur les couches de notre origami simulé.

2.2.4 Box pleating maker

Il s'agit d'un projet neuf (septembre 2013) mais avançant assez vite permettant de travailler sur des modèles en box-pleating. C'est à dire les modèles fondant leur conception sur une grille à partir de laquelle des séquences simples existent afin de faire sortir des parties indépendantes.

Son but est de construire un canevas de plis de box-pleating en se fondant sur la technique du remplissage par carrés (équivalent en box-pleating du remplissage de cercle, mais plus simple à mettre en œuvre).

Nous pouvons remarquer qu'il s'agit d'un projet web, en effet un modèle en *box-pleating* comme nous l'avons signalé en introduction est facile à concevoir, on peut donc se permettre de le faire au travers d'une connexion internet pour générer un canevas de plis à travailler ensuite.

Conclusion

Nous avons donc pu observer qu'il existe en origami de nombreuses tentatives de fournir des outils informatisés pour aider cet art. Cela montre que la communauté est intéressée par cette voie pour permettre à l'art de se diffuser. Nous pouvons de plus signaler qu'il existe des composantes intéressantes de ces programmes qui pourraient être utilisés dans le cadre de notre projet.

2.3 Mathématiques

La construction d'origami est un sujet qui peut être abordé du point de vue mathématique.

Des articles traitent notamment de l'usage de l'art de l'origami afin de résoudre des équations ([1, 5]). La démarche principale est la suivante : les variables sont considérées comme des coordonnées cartésiennes de points du plan ; l'objectif est alors de trouver une série de plis qui, par des formules de symétrie (un pli implique généralement l'usage d'une symétrie), nous ramène à nos équations. Nous savons alors quels points vérifient ces équations.

Cette démarche pose la question de la constructibilité de certains motifs ([10, 11]). Il existe des axiomes définissant les pliages autorisés. Tout origami se construit par une série d'axiomes bien choisis.

Il est alors intéressant de trouver des preuves constructives de constructibilité de figures, afin d'en déduire des séquences de plis.

3 Cahier des charges et applications

Notre objectif est de fournir un outil permettant aux créateurs de pliages de construire aisément un diagramme, afin de diffuser leur découvertes en origami, permettant à cet art de se propager.

3.1 Description des fonctionnalités

La simulation du pliage (modèle 3D de l'origami) : celle-ci permet tout d'abord au créateur de voir les étapes successives de son modèle comme s'il le pliait réellement ; ensuite, cela permet de rendre le diagramme moins sensible aux erreurs (il peut arriver que dans un logiciel de dessin vectoriel, un trait soit oublié ou mal placé). Finalement, cela peut rendre le programme plus ludique en permettant à un néophyte de l'utiliser comme un simple simulateur de pliage.

Une interface épurée mais complète : l'objectif de notre projet étant de toucher un public large, l'interface utilisateur se devra être simple d'utilisation. Celle-ci devra par ailleurs offrir les fonctionnalités suivantes : intégrer la visualisation 3D du pliage en cours de construction, définition de plis, avec « macros » préenregistrées pour les plis complexes mais usuels. Possibilité également de pointer interactivement sur le modèle pour définir de nouveaux points et effectuer les pliages.

Un diagramme lisible : le diagramme généré ne devra pas rebuter les plieurs débutants : il est important de ne pas négliger cet aspect et essayer de rendre le résultat final attractif. Nous allons pour cela nous baser sur les travaux de Lang[9].

Un des challenges sera de parvenir à offrir un rendu 2D permettant de distinguer des plis superposés grâce à la projection 3D appropriée, tout en restant lisible et agréable à l'œil.

Une sortie animée du pliage : il s'agit d'un objectif optionnel. On pourra chercher à générer une sortie animée du pliage, afin de rendre accessible le pliage aux débutants n'ayant pas encore l'habitude de lire les diagrammes. L'interface interactive pourra être réalisée en *javascript*, afin d'être facilement portable.

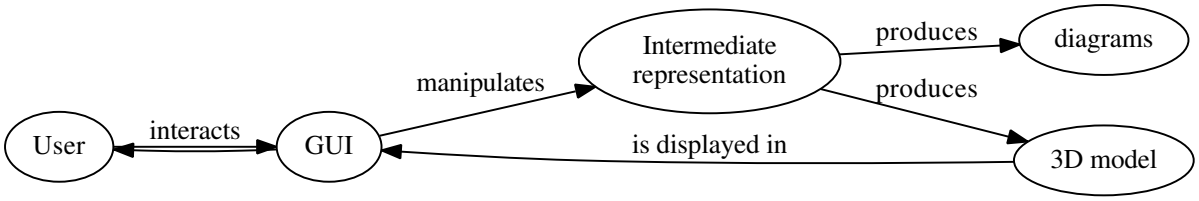


FIGURE 1 – Interaction entre les différents composants

3.2 Aspects techniques

Origram sera conçu comme logiciel installable sur ordinateur par souci d'ergonomie et de stabilité. En effet, son utilisation pourrait être prolongée lors de la description de modèles complexes, une maîtrise de l'environnement est donc souhaitable. Les plateformes visées seront donc *Windows*, *Linux* et éventuellement *MacOS*, en utilisant des outils libres et multiplateformes.

Notre interface sera développée en C++, en utilisant la bibliothèque *Qt*, ainsi que *OpenGL* pour la visualisation 3D : de nombreux efforts ont été réalisés sur la bibliothèque *Qt* afin de produire des interfaces portables et visuellement homogènes, quel que soit le système de l'utilisateur. Par ailleurs, *Qt* et *OpenGL* s'interfacent bien, notamment grâce à des bibliothèques comme *libqglviewer* qui entre autres étend la gestion des événements.

Ces bibliothèques fournissent, utilisées ensemble, un environnement de travail complet et bien documenté, qui nous permettra de gagner un temps certain. En économisant le développement d'un nouveau visualiseur 3D, nous pourrons ainsi nous concentrer sur les aspects spécifiques au projet.

Nous utiliserons un langage de représentation intermédiaire afin d'exprimer les opérations effectuées dans l'interface graphique, de les appliquer sur une structure interne représentant le pliage en 3D, puis de représenter cette structure directement dans l'interface, ou exportée en un diagramme.

Ce langage intermédiaire s'inspirera des règles de réécriture présentées par David Fisher dans sa thèse[2].

Un gros travail sera à fournir du côté des algorithmes permettant de transformer les étapes de pliages, décrites dans le langage intermédiaire, en un modèle 3D. Il s'agit en effet de pouvoir calculer la « physique du pliage », afin de déduire d'une simple instruction la forme tridimensionnelle résultante.

3.3 Licence

En ce qui concerne le choix de la licence, nous souhaitons produire un logiciel libre. Ainsi, les choix possibles sont en autres GPL, BSD, MIT.

Il est à noter que *libqglviewer* est sous licence GPL. Une solution de simplicité serait alors de réaliser le projet sous licence GPL. Cependant, il est possible d'intégrer du code sous licence GPL à du code sous licence BSD/MIT, en indiquant précisément quelle partie est concernée par la licence GPL. De plus, nous préférons une licence type BSD/MIT car elle est moins restrictive et nous souhaitons partager au mieux notre production, sans complications superflues.

Le projet sera donc réalisé sous une licence type BSD ou MIT.

3.4 Retombées attendues

Notre objectif principal est de fournir aux origamistes un moyen efficace de propager leur art, ce qui peut être aujourd'hui difficile en l'absence d'outils adaptés. Le succès de ce projet représenterait un bond dans le cadre de la diffusion de l'origami auprès des néophytes ainsi que la possibilité pour les créateurs de pouvoir exprimer facilement les techniques qu'ils ont découvertes et montrer leurs créations les plus originales.

4 Planification et organisation

Les réunions hebdomadaires du lundi matin permettent à l'ensemble de l'équipe de faire le point sur les avancées de chacun dans ses tâches respectives et de faire un bilan sur l'avancement d'un point de vue global. Des réunions seront organisées entre les membres d'un même *workpackage*, ou de *workpackage* voisins, plus irrégulièrement : elles serviront à affiner le travail plus précis et plus spécialisé, où la coordination est essentielle. Le gestionnaire de versions Git nous permettra également une contribution et communication relativement continue et facile entre membres du projet.

4.1 Répartition des tâches

Nous avons divisé le projet en plusieurs *workpackages*, eux-mêmes subdivisés en *tâches*. La répartition temporelle des différentes tâches est présentée sur le diagramme de Gantt en annexe. Le détail et la répartition humaine des tâches sont les suivants (la personne en gras est responsable du bon déroulement de la tâche) :

WP 1 : Communication Ceci comprend toutes les opérations de communication, du sondage des attentes des gens à la gestion d'un site internet [2]. Il y a d'ailleurs déjà eu des bonnes avancées dans ce domaine lors de la récente convention d'origami LUO7, avec des plieurs se disant en général très intéressés par notre projet, et certains se disant prêts à tester notre logiciel dès que celui-ci serait utilisable. Elle a de plus permis d'obtenir la participation d'intervenants extérieurs, comme *Jérôme Goût* le créateur de `Doodle`, ou encore *Aurèle Duda*, le créateur des modules inkscape fournissant les notations conventionnelles.

Initial communication : Consiste à créer le site web, contacter la communauté origamiste pour connaître les fonctionnalités attendues

Production : site web, document résumant les attentes des utilisateurs.

2. http://perso.ens-lyon.fabrice.mouhartem/origram

Fabrice Mouhartem, Damien Rouhling

Real-world tests : Organiser des séances de test du logiciel auprès de la communauté origamiste. Cibler des publics variés : origamiste débutant ou créateur, âge, ... Concevoir des tests pertinents pour avoir des retours d'expériences sur différentes parties du projet.

Production : document décrivant les retours d'expérience des testeurs.

Jérémy Ledent, Fabrice Mouhartem

WP 2 : Documentation et bibliographie Ceci comprend tout le travail de bibliographie : tout ce qui concerne l'étude et l'analyse ainsi que l'intégration des travaux déjà effectués (que ces travaux soient de natures théoriques, comme une thèse, ou pratique, comme un logiciel) est regroupé dans ce paquet.

Production : archive de papiers scientifiques, recensement des logiciels existants.

Grégoire Beaudoire, Fabrice Mouhartem

WP 3 : Interface graphique Ceci comprend toute la partie ergonomie du projet : ce à quoi l'interface finale doit ressembler, ainsi que la GUI et toutes ses fonctionnalités. Le comportement standard de fonctionnement sera de type besoin/solution. Tous les autres *packages* fournissant au fur et à mesure des demandes d'accessiblité à des fonctionnalités au sein de l'interface graphique, qui devront ensuite être regroupées et organisées dans une interface logique et ergonomique.

GUI ergonomy : Concevoir l'interface utilisateur : ce qu'elle doit permettre de faire et ce à quoi elle doit ressembler en pratique.

Production : document décrivant les fonctionnalités attendues et la façon dont la GUI répond à cette attente. Maquette d'interface.

Grégoire Beaudoire, Jérémy Ledent, Antoine Pouille

GUI implementation : Mettre en place les outils permettant de construire la GUI. Construire effectivement la GUI d'après le travail de l'équipe d'ergonomie. Concevoir un moyen de relayer les actions de l'utilisateur aux différents modules du programme.

Production : implémentation de la GUI. Moyen d'interfacer la GUI avec les autres parties du projet qui en ont besoin (API, bibliothèque, boucle événementielle, ...).

Antoine Pouille, Pierre Pradic

User feedback on GUI : Modifier et adapter l'interface graphique en suivant les retours d'expérience.

Grégoire Beaudoire, Antoine Pouille

WP 4 : Représentation intermédiaire Ceci comprend la définition de primitives permettant de représenter les étapes d'un pliage, et la production d'un format adapté à la représentation interne d'un pliage (avec éventuelle réutilisation d'un format existant). Ce format de représentation intermédiaire est indispensable aux *workpackages* « 3D » et « Production de diagramme ».

Intermediate representation : Concevoir un langage de description ou une structure de données permettant de représenter une séquence de pliage. Les actions de l'utilisateur doivent pouvoir être représentées. D'autre part, cette représentation doit permettre de générer un modèle 3D pour l'afficher à l'utilisateur (réalisé par la tâche « paper physics »), et doit permettre de générer un diagramme (réalisé par la tâche « diagram generation »).

Production : un langage de description ou une structure de données permettant de représenter une séquence de pliage.

Damien Rouhling, Jérémy Ledent, Fabrice Mouhartem, Armaël Guéneau, Pierre Pradic

User interaction : Prendre en compte les actions de l'utilisateur de la GUI, notamment sur la vue 3D (sélection de point, ligne, pointe, pliage). Traduire les actions de l'utilisateur en utilisant la représentation intermédiaire, notamment par l'implémentation de macros dans le langage intermédiaire.

Production : code permettant de manipuler la représentation intermédiaire à partir des actions de l'utilisateur.

Pierre Pradic, Damien Rouhling, Antoine Pouille

User feedback on intermediate representation : Modifier et adapter la représentation intermédiaire pour répondre aux nouveaux besoins.

Armaël Guéneau, Jérémy Ledent, Pierre Pradic

WP 5 : Modélisation du papier/3D Ce *package* va fournir le contenu de l'affichage OpenGL de la fenêtre Qt, où l'affichage du modèle et le choix des opérations s'effectue. Il va développer une traduction de l'entrée utilisateur pour le langage intermédiaire, et ensuite la conversion en un modèle 3D à afficher. Cela demande alors un travail de modélisation de physique du papier, gérant le marquage des plis et les tensions parfois nécessaires à la construction du modèle. L'interaction entre ce groupe et celui s'occupant de la représentation intermédiare sera très importante.

3D OpenGL : Mettre en place les primitives d'affichage et de manipulation 3D : lumière, rotation, textures, modèle 3D. Les besoins de la tâche « user interaction » doivent être anticipés (l'utilisateur devra pouvoir sélectionner des points, des lignes, ...).

Production : format permettant de décrire un modèle 3D quelconque, code générique permettant à l'utilisateur d'afficher et de manipuler un modèle 3D (faire tourner, zoomer).

Armaël Guéneau, Baptiste Jonglez, Antoine Pouille

Paper physics : Utiliser la représentation intermédiaire pour générer un modèle 3D du pliage. Prendre en compte des règles de physique élémentaires pour générer un modèle réaliste (exemple du bateau). Donner des critères sur la faisabilité des actions de l'utilisateur (pliage impossible à réaliser).

Production : code permettant de générer une vue 3D du pliage à partir de la représentation intermédiaire.

Baptiste Jonglez, Grégoire Beaudoire, Fabrice Mouhartem, Armaël Guéneau

WP 6 : Production de diagramme Ceci comprend la génération d'un diagramme dans différents formats, à partir de la représentation intermédiaire, des rendus 3D, et d'éventuels choix de l'utilisateur, en respectant les conventions des origamistes. Cela implique de plus le développement d'heuristiques afin d'obtenir un résultat lisible et esthétique. Ce *workpackage* dépend fortement des autres : en particulier du *workpackage* « 3D ».

Production : code permettant de générer un diagramme dans différents formats (PDF, PS, image, HTML), en respectant les conventions usuelles des diagrammes.

Fabrice Mouhartem, Jérémy Ledent, Damien Rouhling

WP 7 : Intégration Assembler les différents composants logiciels pour obtenir un produit complet. Cela peut inclure l'harmonisation des systèmes de compilation, l'écriture de code d'interface entre composants, voire la définition de nouvelles sous-tâches au sein de composants existants. Ce *workpackage* dirigera le cycle intégration/test qu'il faudra répéter deux ou trois fois au cours du projet.

Production : un prototype complet du projet

Baptiste Jonglez, Damien Rouhling

Nous avons de plus défini trois tâches annexes ne faisant pas partie des *workpackages* :

Coding conventions : Définir les standards à suivre, aussi bien pour le code (indentation, style) que pour les dépôts git ou le système de compilation du projet.

Production : document décrivant les bonnes pratiques à suivre.

Baptiste Jonglez, Armaël Guéneau

Mid-term report : Coordiner l'écriture du rapport de mi-parcours. Collecter les différentes parties écrites par chaque module.

Production : le rapport de mi-parcours

Damien Rouhling, Grégoire Beaudoire

Advanced features : Implémentation de fonctionnalités additionnelles, éventuellement proposées au cours des tests. Intégration de logiciels existants comme Treemaker.

Antoine Pouille, Baptiste Jonglez

4.2 Mise à l'épreuve du programme

Afin d'obtenir un logiciel efficace, il est nécessaire de procéder à des tests pour détecter d'éventuels problèmes techniques, mais aussi ergonomiques.

Tout au long du développement, le *workpackage* « Intégration » s'occupera de réunir les différentes parties du programme, et de déclencher des tests généraux, puis par *workpackage*. Cela conduira à des premières versions qui estimeront les besoins des utilisateurs finaux.

Pour aller plus loin, le logiciel sera proposé à d'authentiques origamistes, n'ayant éventuellement que peu de compétences en informatique, permettant de réellement se rendre compte des besoins typiques d'un utilisateur et des manques qu'il faudrait combler. Des démarches ont déjà été faites en ce sens et des contacts ont d'ores et déjà accepté de jouer ce rôle indispensable.

Conclusion

Ce projet, dans le cadre d'un premier semestre de M1 à l'ENS de Lyon, se propose d'améliorer de façon conséquente le travail de conception et de partage de modèles d'origamis. En effet, les solutions actuelles sont soit longues et pénibles, soit mal conçues au niveau de leur usabilité ou de leur rendu. Il permettrait enfin à cette communauté de numériser facilement leurs travaux et donc de les partager bien plus largement.

Ce type de logiciel a été tenté a plusieurs reprises, mais n'a jamais vraiment réussi à aboutir, par manque d'ambition ou erreurs de conception. Nous avons confiance en notre capacité à fournir cet outil, en le concevant de manière intelligente et se souciant des besoins de nos futurs utilisateurs.

Références

[1] R. C. A. Robert J. Lang. One-, two-, and multi-fold origami axioms. 2006.

[2] D. Fisher. *Origami On Computer*. PhD thesis.

[3] J. Goût and V. Osele. Doodle. http://doodle.sourceforge.net/about.html.

[4] M. Hassine. Box pleating maker. http://h3md.free.fr/travaux/BPMaker/.

[5] T. C. Hull. Solving cubics with creases : The work of beloch and lill. *The American Mathematical Monthly*, 118 :307–315, 2011.

[6] T. K. Lam. Origami simulator and diagrammer. http://www.angelfire.com/or3/tklorigami0/.

[7] T. K. Lam. An investigation of the usability of software for producing origami instructions, 2005.

[8] R. J. Lang. Computational origami. http://www.langorigami.com/science/computational/computational.php.

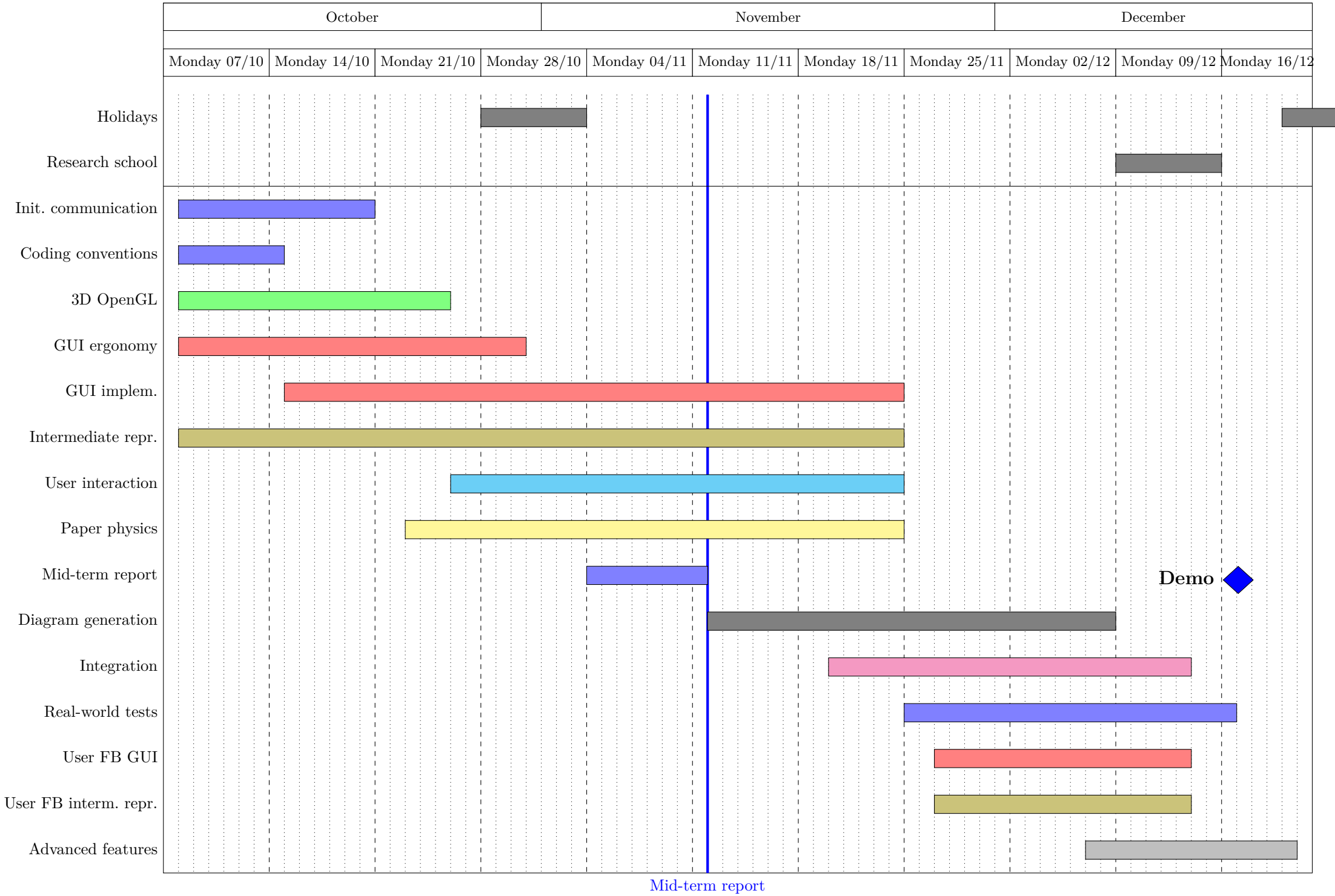
[9] R. J. Lang. Origami diagramming conventions. 1989-1991.

[10] R. J. Lang. Angle quintisection, 2004.

[11] R. J. Lang. Origami and geometric constructions. 2010.

[12] J. Mitani. Oripa. http://mitani.cs.tsukuba.ac.jp/oripa/.

[13] J. Zing Man. Foldinator. http://zingman.com/origami/foldinator.php.


October
November
December
Monday 07/10
Monday 14/10
Monday 21/10
Monday 28/10
Monday 04/11
Monday 11/11
Monday 18/11
Monday 25/11
Monday 02/12
Monday 09/12
Monday 16/12
Holidays
Research school
Init. communication
Coding conventions
3D OpenGL
GUI ergonomy
GUI implem.
Intermediate repr.
User interaction
Paper physics
Mid-term report
Demo
Diagram generation
Integration
Real-world tests
User FB GUI
User FB interm. repr.
Advanced features
Mid-term report