Animation par croquis dans le logiciel RUMBA - Rapport final du projet COLLODI2

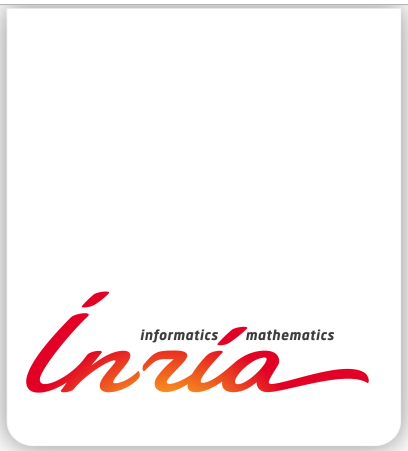
Julien Daval, Maguelonne Beaud de Brive, Valérian Daunis, Rémi Ronfard

Animation par croquis dans le logiciel RUMBA - Rapport final du projet COLLODI2

Julien Daval, Maguelonne Beaud de Brive,
Valérian Daunis, Rémi Ronfard

TECHNICAL REPORT
N° 0505
September 2019
Project-Team IMAGINE

ISSN 0249-0803 ISRN INRIA/RT--0505--FR+ENG



Animation par croquis dans le logiciel RUMBA - Rapport final du projet COLLODI2

Julien Daval, Maguelonne Beaud de Brive,
Valérian Daunis, Rémi Ronfard

Équipe-Projet IMAGINE

Rapport technique n° 0505 — September 2019 — 51 pages

Résumé : Ce rapport décrit les travaux de l'équipe IMAGINE d'INRIA pendant le projet FUI COLLODI2 en partenariat avec TEAMTO et Mercenaries Engineering. En particulier, nous décrivons le portage qui a été fait de nos outils de posing et d'animation par croquis dans le logiciel professionel RUMBA; les nouveaux développements réalisés au cours du projet; et les évaluations qui en ont été faites par un animateur professionnel.
Ce travail a été financé par la Région Rhone Alpes Auvergne.

Mots-clés : Informatique graphique, animation 3D, modélisation par croquis.

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Sketch-based animation in the RUMBA software - COLLODI2 Project Final report

Abstract: This report describes the work of the IMAGINE team at Inria during the FUI COLLODI2 project in partnership with TEAMTO and Mercenaries Engineering. THis includes the rewriting of our previous work in sketch-based posing and animation for the RUMBA professional animation software; new features developped specially for COLLODI2; and an experimental evaluation of the quality, reliability and usability of the tools by a professional animator.

This work was funded by Region Rhone Alpes Auvergne.

Key-words: Computer graphics, 3D animation, sketch-based modeling.

Table des matières

1 Définitions

Commençons par définir un certain nombre de termes, qu'on utilisera souvent dans la suite de ce document.

On appelle **rig** d'un modèle 3D l'ensemble des éléments qui permettent de le poser et de l'animer (c'est une sorte de « marionnette » numérique que l'on peut manipuler). Pour les modèles les plus basiques, le rig n'est constitué que du squelette (un ensemble d'os - *bones* - hiérarchiquement connectés). Mais les rigs utilisés en production sont en général beaucoup plus complexes : la manipulation du modèle se fait alors au moyen de **contrôleurs**, sorte d'abstraction haut niveau du squelette. La manipulation des contrôleurs permet de respecter des contraintes définies sur le modèle (contraintes de cinématique inverse, de limitation d'angle, de parenté ...). Afin de ne pas « casser » le rig, nos outils **ne manipuleront que les contrôleurs**. Pour les modèles basiques (en général dépourvus de contrôleurs), on pourra sans soucis assimiler les contrôleurs aux bones.

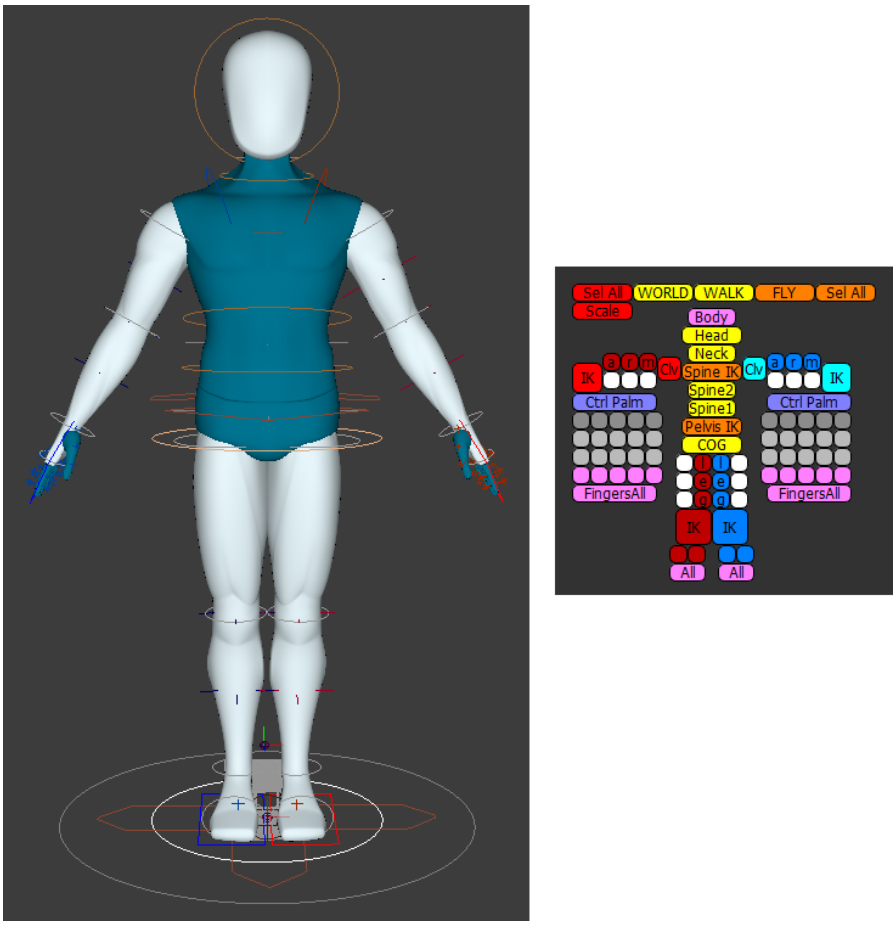


FIGURE 1 – Exemple d'un rig et du *picker* (permettant de sélectionner les contrôleurs) associé. Les différents cercles sont les contrôleurs

Une **bodyline** est une chaîne de contrôleurs, qui pourra par la suite être posée avec une ligne d'action. Par définition, une bodyline est **maximale** : elle reliera toujours deux extrémités du modèle (par exemple, pour un personnage, un pied et une main). Cependant, nous autorisons également la manipulation de bodylines non-maximales (un seul bras, un buste ...).

Une **ligne d'action** est une courbe 2D utilisée pour définir la nouvelle forme d'une bodyline. La ligne d'action a en général une forme très simple (on parle de courbe en « C » ou en « S »), qui sert normalement à représenter la forme générale du modèle.

2 Pose par ligne d'action

Cette partie expose et détaille toutes les fonctionnalités implémentées dans Rumba concernant la pose de modèles 3D par le dessin de lignes d'action. L'objectif est de pouvoir, à un instant donné, sélectionner une partie du modèle et la modifier uniquement à l'aide de sketches 2D. Cet outil fonctionne de la manière suivante :

1. L'animateur sélectionne d'abord la bodyline qu'il souhaite poser ;
2. Il dessine ensuite la ligne d'action. Les contrôleurs de la bodyline sont alors bougés pour prendre la forme de la ligne d'action.

Tout ce qui suit (algorithme de pose, twist de bodyline ...) est principalement inspiré des travaux effectués par Martin Guay pendant sa thèse [3] [4] [5].

2.1 Hiérarchie de contrôleurs

Afin de pouvoir sélectionner des bodylines, il est nécessaire de connaître l'organisation du rig, pour déterminer les contrôleurs faisant partie d'une bodyline. Par exemple, la bodyline allant d'un pied à une main contiendra les contrôleurs de la jambe, du bras mais aussi ceux du buste. La *nodelist* de Rumba (qui décrit la scène 3D) peut nous aider mais elle ne décrit malheureusement pas tous les liens entre les contrôleurs. Nous avons en effet souvent rencontré le cas où deux contrôleurs n'étaient pas hiérarchiquement liés mais où le mouvement de l'un entraînait l'autre. Or la connaissance de cette structure hiérarchique est primordiale pour que les processus de sélection et de pose de bodyline soient corrects.

Pour résoudre ce problème, nous avons fait le choix de demander à l'utilisateur (qui dans ce cas serait le *rigger* et non pas l'animateur) de définir manuellement la hiérarchie de contrôleurs, dans un fichier séparé au format *JSON* (qui se prête bien à la représentation de hiérarchies). Cela demande certes plus de travail mais :

— Le fichier n'a besoin d'être crée qu'une seule fois, il sera ensuite automatiquement chargé en même temps que le modèle ;
— Pour les modèles les plus simples, ce fichier peut être généré automatiquement, les contrôleurs étant assimilés aux bones ;
— Cela permet de définir une hiérarchie consituée d'une sous partie des contrôleurs. On peut alors enlever certains contrôleurs (qui ne seront donc pas utilisés dans les processus de sélection et de pose) ou en ajouter selon les besoins.

L'annexe B illustre un exemple de fichier de hiérarchie de contrôleurs ainsi que les éléments qui le constituent.

2.2 Sélection de bodyline

Le processus de sélection de la bodyline est effectué de la manière suivante :

— On détermine les deux contrôleurs les plus proches (en espace écran) des deux extrémités du sketch (une conversion de la position des contrôleurs de l'espace 3D vers l'espace écran est donc nécessaire). Ces contrôleurs définissent alors les extrémités de la bodyline ;
— On utilise ensuite la hiérarchie de contrôleurs pour trouver le chemin entre ces deux contrôleurs (en utilisant un algorithme classique de recherche de chemin dans un graphe).

La recherche des contrôleurs les plus proches peut se faire soit uniquement sur les contrôleurs qui n'ont pas d'enfant dans la hiérarchie (si on ne veut que des bodylines maximales) soit sur tous les contrôleurs (si on autorise tout type de bodyline).

Notons que **seules les extrémités de la courbe** sont utilisées pour déterminer les contrôleurs les plus proches. Il n'est donc pas nécessaire de respecter scrupuleusement la forme de la bodyline que l'on souhaite sélectionner. Par exemple, si on veut sélectionner la bodyline allant d'un pied à l'autre, on peut tracer un trait allant directement du pied à l'autre, sans forcément respecter la forme des deux jambes.

Le sens de sélection de la bodyline est très important : la bodyline allant d'un contrôleur A à un contrôleur B sera différente de celle allant de B à A, même si ces bodylines ont les mêmes contrôleurs. Cet ordre de sélection a une influence sur la manière avec laquelle on dessinera la ligne d'action par la suite. Pour éviter toute ambiguïté, on affiche la bodyline avec une flèche indiquant le sens de sélection.

Une fois la bodyline sélectionnée, ses contrôleurs sont tous ajoutés à la sélection de Rumba, ce qui permet d'insérer très facilement des clés.

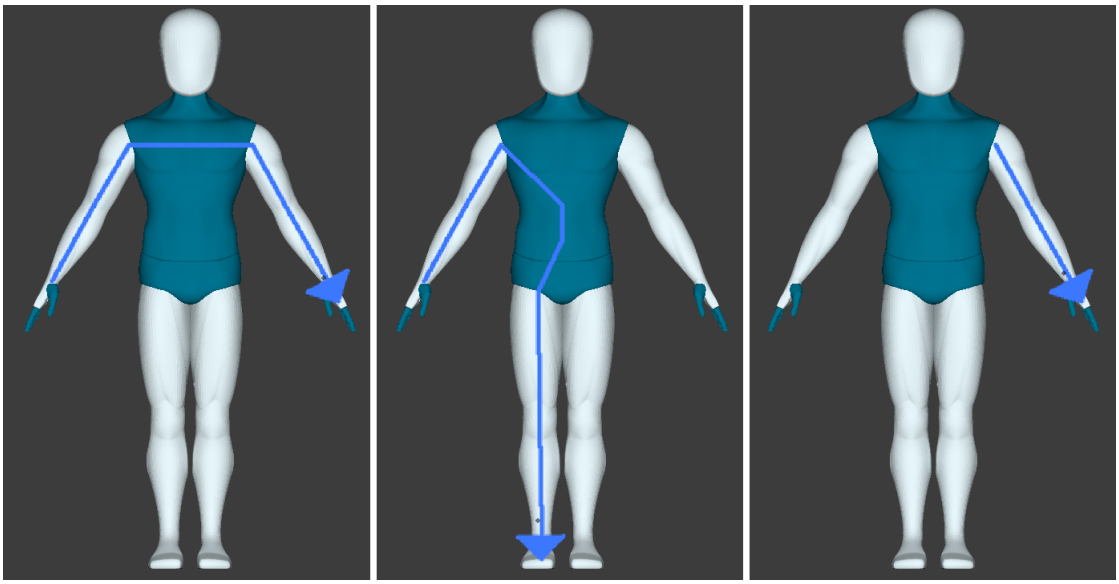


FIGURE 2 – Sélection de bodylines

2.3 Pose de bodyline

L'animateur définit ensuite la nouvelle forme de la bodyline sélectionnée en dessinant une ligne d'action. L'algorithme de pose de la bodyline s'effectue alors en plusieurs étapes.

2.3.1 Pré traitement

Une ligne d'action est par définition en « C » ou en « S », ce qui se modélise mathématiquement par une **courbe de Bézier cubique**. Mais ce n'est pas toujours le cas de la courbe dessinée par l'animateur. S'il le souhaite, il peut demander à ce que la courbe dessinée soit remplacée par la courbe « ligne d'action » qui l'approxime au mieux. Cette courbe est déterminée via une méthode des moindres carrés : on cherche la courbe de Bézier cubique $B = \sum_{k=0}^{3} B_k P_k$ telle que

$$E = \frac{1}{2} \sum_{i=0}^{N-1} \|B(t_i) - Q_i\|^2 + \alpha \|B'(t_i) - V_i\|^2$$

soit minimal. B' correspond à la dérivée de la courbe de Bézier, les Q_i (V_i) sont les points (tangentes) du sketch initial que l'on cherche à approximer et α est un paramètre permettant de régler l'importance de la correspondance *en forme* par rapport à la correspondance *en position*.

L'animateur peut aussi demander à ce que la racine du modèle soit déplacée pour que la bodyline coïncide exactement avec la ligne d'action. Dans ce cas, on la déplace pour faire correspondre les milieux de la bodyline et de la ligne d'action (si la bodyline est maximale) ou les débuts. Ne pas la déplacer peut aussi être utile, par exemple si on veut poser le modèle sur place.

Il faut ensuite construire un « mapping » entre la ligne d'action et la bodyline, pour savoir à quel point de la ligne d'action faire correspondre chaque contrôleur. Ce « mapping » est calculé comme suit :

— Les positions des contrôleurs de la bodyline sont converties dans le même espace que celui de la ligne d'action (espace écran) ;
— Les deux courbes sont ensuite paramétrées par abscisse curviligne normalisée (une valeur comprise entre 0 et 1 représentant une fraction de la courbe - par exemple le point d'abscisse 0.5 correspond au milieu de la courbe) ;
— On associe finalement à chaque contrôleur le point de la ligne d'action qui a la même abscisse curviligne.

2.3.2 Décomposition en sous-bodylines

Une fois ce pré traitement effectué, la bodyline doit être décomposée en sous-bodylines. En effet, tous les contrôleurs ne sont pas toujours hiérarchiquement liés *dans le même sens* (par exemple, un contrôleur n'est pas forcément un enfant du contrôleur le précédent dans la bodyline). Pour éviter les erreurs de pose, il faut décomposer la bodyline de sorte que pour chaque sous-bodyline, tous les contrôleurs soient dans le même sens. Par exemple, une bodyline allant d'un pied à une main sera décomposée en deux sous-bodylines : la jambe et le buste-bras.

2.3.3 Pose des sous-bodylines

On peut donc poser chaque sous-bodyline. On se confronte alors à un problème : comment manipuler des contrôleurs 3D avec une courbe 2D (sans information de profondeur) ? Nous avons fait le choix de restreindre les mouvements des contrôleurs (translation et rotation) à un **plan d'action** P :

— Dont la normale N est la direction de la caméra ;
— Et passant par la racine du modèle.

La pose d'une sous-bodyline s'effectue de la manière suivante :

1. Soit C_i un contrôleur de la sous-bodyline et C_{i-1} son parent (on ne touche en fait pas au premier contrôleur). On appelle L_i, L_{i-1} les points de la ligne d'action qui leur sont associés ;
2. On projette C_i et C_{i-1} dans P. On peut donc calculer un premier vecteur V_C allant de la projection de C_{i-1} vers la projection de C_i ;
3. On fait la même chose pour L_i et L_{i-1}, ce qui donne un vecteur V_L ;
4. On calcule l'angle θ entre V_C et V_L, l'objectif étant d'aligner ces deux vecteurs ;
5. On applique à C_{i-1} une rotation d'angle θ et d'axe N pour que C_i soit correctement placé ;
6. On recommence l'étape 1 avec le contrôleur suivant.

L'algorithme ci-dessus suppose que tous les contrôleurs sont rigides. Si on dessine une ligne d'action d'une taille sensiblement différente à celle de la bodyline, les deux courbes se ressembleront en termes de « forme » mais pas en position, ce qui n'est pas forcément illogique. L'animateur peut cependant activer le **squash & stretch**, s'il souhaite donner à la bodyline un effet *cartoon*. L'ajout est assez simple : après avoir appliqué la rotation à C_{i-1} (étape 5), on applique à C_i une translation le long de l'axe $\overrightarrow{C_{i-1}C_i}$ de sorte que la nouvelle position du contrôleur corresponde exactement avec celle de la ligne d'action.

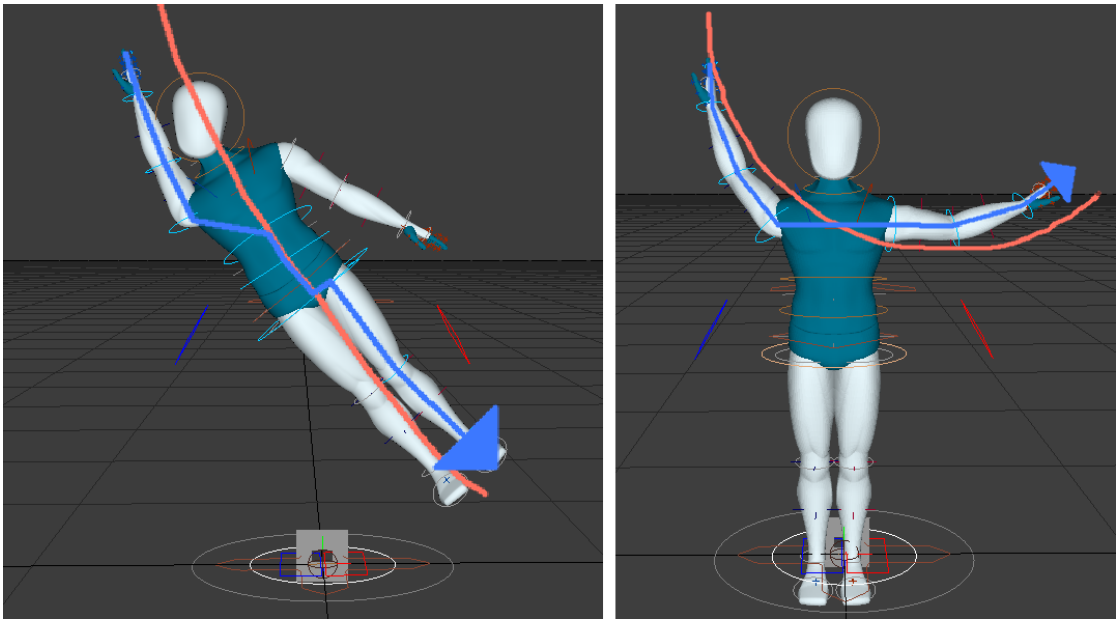


FIGURE 3 – Imprécision de la pose de bodylines (réalisé en 2017)

Les poses illustrées sur la figure 3, obtenues par les lignes d'action en rouge, sont moyennement satisfaisantes :

— La correspondance est assez bonne en termes de *forme* ;

— Mais on sent qu'on pourrait faire bien mieux en termes de *position* : sur la figure de gauche, l'épaule n'est pas positionnée sur la ligne d'action et on observe un décalage avec la jambe. On retrouve le même type de problèmes sur la figure de droite.

Nous avons grandement amélioré la précision de la pose en modifiant le processus de décomposition de la bodyline en sous-bodylines et en modifiant également l'étape de translation de la racine du modèle.

Au départ, la décomposition de la bodyline en sous-bodylines se faisait au moyen de **séparateurs**, contrôleurs fictifs que l'utilisateur pouvait ajouter dans le fichier de hiérarchie pour indiquer où la bodyline devait être « découpée ». Par exemple, la bodyline de la figure 3 (gauche) était en réalité découpée en trois sous-bodylines : le bras, le buste et la jambe. Comme chaque sous-bodyline est posée indépendamment et qu'on ne bouge jamais le premier contrôleur d'une sous-bodyline, l'épaule n'était jamais bougée. Or les contrôleurs du bras et du buste sont dans le même sens (l'épaule est un descendant du dernier contrôleur du buste). Il n'y a donc aucune raison qui justifie la séparation de ces deux sous-bodylines. Nous avons donc modifié la méthode de décomposition d'une bodyline pour éviter ce genre de séparations.

Mais cela ne règle pas le problème d'écart que l'on a entre la ligne d'action et la jambe par exemple. Nous avons réussi à résoudre ce problème, mais la solution que nous avons adoptée ne fonctionne que dans le cas où la bodyline se décompose en deux sous-bodylines. Ce cas de figure se rencontre cependant très souvent en pratique (par exemple, sur un personnage, les bodylines se décomposent en deux sous-bodylines au maximum). L'idée est de bouger le parent commun (qui peut être la racine mais aussi un autre contrôleur – le contrôleur du buste dans le cas des deux bras) des deux extrémités des deux sous-bodylines afin de faire correspondre le segment formé par les deux contrôleurs et le segment formé par les points de la ligne d'action associés aux deux extrémités.

— Après avoir projeté les deux positions des extrémités ainsi que celles des points de la ligne d'action qui leur sont associées dans le plan d'action, on applique une rotation au parent commun afin d'aligner les deux segments ;
— On applique finalement une translation pour faire correspondre les milieux des deux segments. Notons que pour éviter un squash non voulu, la translation s'appliquera toujours à la racine du modèle.

Les résultats de la pose avec ces modifications sont illustrés sur la figure 4.

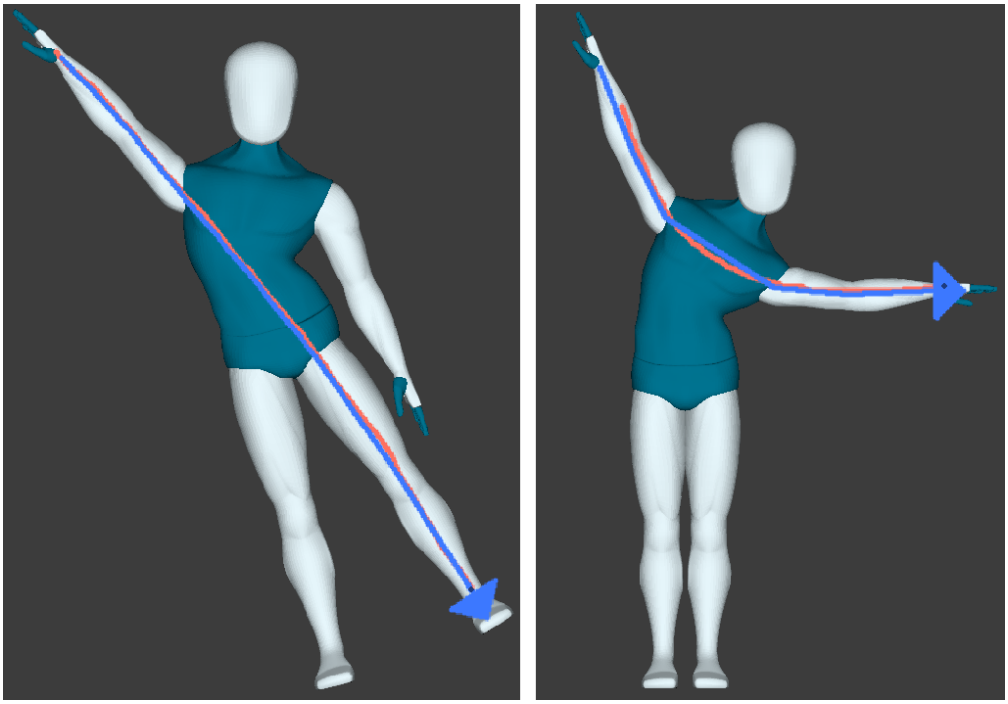


FIGURE 4 – Amélioration de la précision des poses (réalisé en 2019)

2.4 Gestion des contraintes

Les contraintes sont primordiales en animation car elles permettent d'éviter des situations peu réalistes en pratique (par exemple, un personnage a souvent ses pieds au sol). Sans ces contraintes, il faudrait constamment corriger toutes les imperfections causées par les manipulations des contrôleurs. Le côté « libre » de l'outil de pose et de ses options augmente encore plus l'intérêt de conserver ces contraintes. Nous avons adapté l'outil de pose pour prendre en compte deux types de contraintes :

— Les contraintes de *parenté* (par exemple un personnage qui tient un objet dans sa main) ;
— Les contraintes d'*environnement* (un personnage gardant sa main sur une table ou les pieds au sol), qui sont en général représentées par des contraintes de cinématique inverse (IK).

2.4.1 Contraintes de parenté

Il se trouve que notre outil de pose gère déjà indirectement ce type de contraintes. En effet, la contrainte de parenté oblige un *enfant* à suivre les mouvements (orientation et translation) d'un *parent*. Si on bouge avec une ligne d'action un contrôleur qui contraint un objet, cet objet suivra automatiquement, sans nécessiter d'autre action de notre part.

2.4.2 Contraintes d'environnement

La gestion des contraintes d'environnement est plus compliquée, compte-tenu de la manière avec laquelle Rumba gère les contraintes IK. Par souci de clarté, nous supposerons par la suite que le modèle 3D est **un personnage** (certaines des fonctionnalités que nous avons développé

sont spécifiques aux personnages, même si le côté « théorique » est valable pour n'importe quel type de modèle).

Les rigs de production proposent en général deux types de contrôleurs pour manipuler les extrémités d'un personnage (bras, jambes) :

— Les contrôleurs FK (*forward kinematics*), qui permettent de manipuler un à un les contrôleurs du membre (épaule, coude et poignet ou hanche, genou et cheville) ;
— Le contrôleur IK, situé en général au niveau de la main ou du pied, et qui contrôle tout le membre, via des algorithmes dédiés.

Ces deux modes de manipulation sont en général **incompatibles** : si on est en mode IK, alors seul le contrôleur IK peut bouger le membre (les contrôleurs FK n'ont aucun effet). Si on est en mode FK, il n'est pas possible de manipuler le contrôleur IK.

Pour poser une bodyline, il est nécessaire de manipuler les contrôleur FK, car la ligne d'action contraint en position tous ses contrôleurs. Mais le fait d'être en FK risque de casser les contraintes IK, car la manipulation des contrôleurs IK n'a aucun effet.

Nous avons adapté notre outil de pose afin de garantir au maximum le respect de ces contraintes. Lorsqu'on pose une bodyline, pour chaque membre du personnage, deux situations se présentent :

— Soit le membre ne fait pas partie de la bodyline : cela ne signifie pas qu'il ne peut pas bouger (par exemple, la bodyline allant d'un pied à une main contient le contrôleur du bassin, dont le mouvement affecte l'autre jambe). Dans ce cas, le membre reste en IK : Rumba s'assure alors que la contrainte reste respectée ;
— Soit le membre fait partie de la bodyline. Dans ce cas, on calcule dans un premier temps la pose sans tenir compte des contraintes, puis on remet le membre en IK afin de restaurer la contrainte.

Le fait de restaurer la contrainte IK peut cependant casser la forme imposée par la ligne d'action. Dans certaines situations, c'est normal car on ne peut pas à la fois avoir la forme désirée et respecter la contrainte IK. Dans les autres cas, on utilise un contrôleur particulier, souvent spécifique aux personnages : le **pole vector**. Le pole vector sert à la base à s'assurer que les jambes et les bras se plient correctement lorsqu'on les bouge avec les contrôleurs IK. Une fois que la jambe est pliée, on peut bouger le pole vector pour orienter le genou selon nos souhaits.

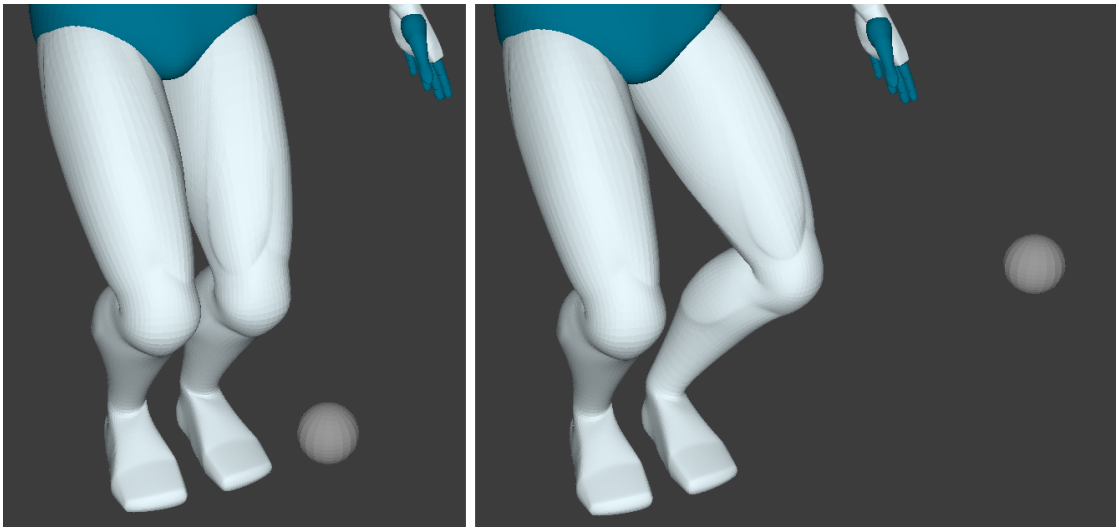


FIGURE 5 – Pole vector (représenté par la sphère) et orientation du genou

Quand on passe un membre en IK, Rumba oriente le genou/coude pour l'aligner avec le pole vector, ce qui peut casser la forme donnée par la ligne d'action. Pour éviter cela, après avoir posé le membre, on bouge le pole vector afin de faire en sorte que le genou/coude ne soit pas bougé quand on passe le membre en FK. Par exemple, pour une jambe, le pole vector est positionné dans le plan formé par la hanche, le genou et la cheville.

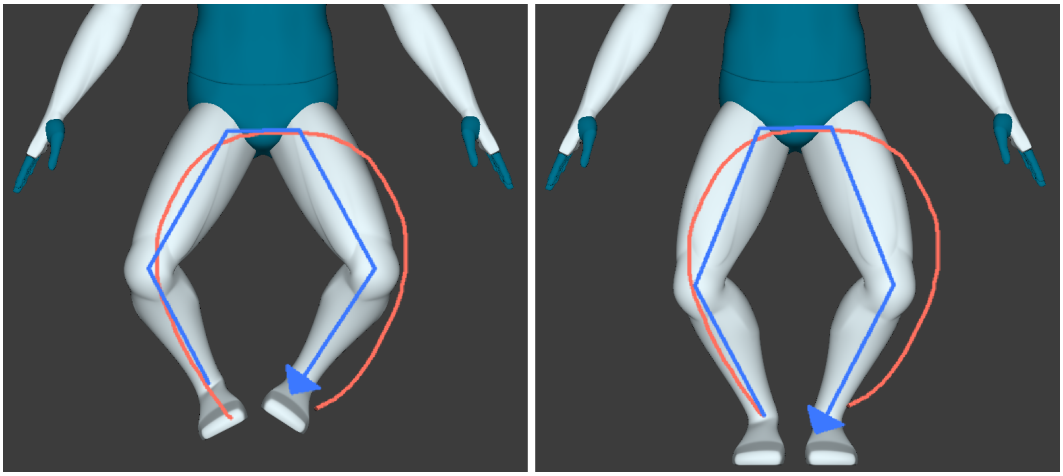


FIGURE 6 – Pose sans puis avec gestion des contraintes au sol pour les jambes

2.5 Twist de bodyline

Lorsqu'on pose une bodyline, toutes les rotations et translations sont effectuées dans le plan d'action. Mais on aimerait également inclure des effets de torsion :

— Lorsqu'un personnage saute, il arrive qu'il tourne légèrement son buste ;
— Un mouvement fluide et expressif du bras comprend souvent des torsions du coude ;
— Le mouvement de deux personnages qui dansent est constitué de rotations de parties du corps

Il est possible d'ajouter manuellement ces rotations en manipulant les contrôleurs avec les outils de rotation traditionnels. Mais cela peut nécessiter le réglage de beaucoup de contrôleurs, surtout si on veut une torsion réaliste.

Pour permettre une torsion réaliste tout en limitant le nombre de paramètres à régler, nous avons implémenté le système de **twist par canettes** (*two-cans technique*), concept issu de l'animation 2D et étudié par Guay *et al.* [5]. Son fonctionnement est le suivant :

— Si une bodyline est sélectionnée, l'animateur peut décider d'activer le twist. Dans ce cas, deux canettes apparaissent aux extrémités de la bodyline ;
— Lorsqu'il bouge une des canettes, un twist est automatiquement calculé pour chaque contrôleur de la bodyline. L'angle de twist est calculé par interpolation entre les angles des deux canettes. L'axe de twist est défini par convention comme étant l'axe $+Y$ du repère local du contrôleur.

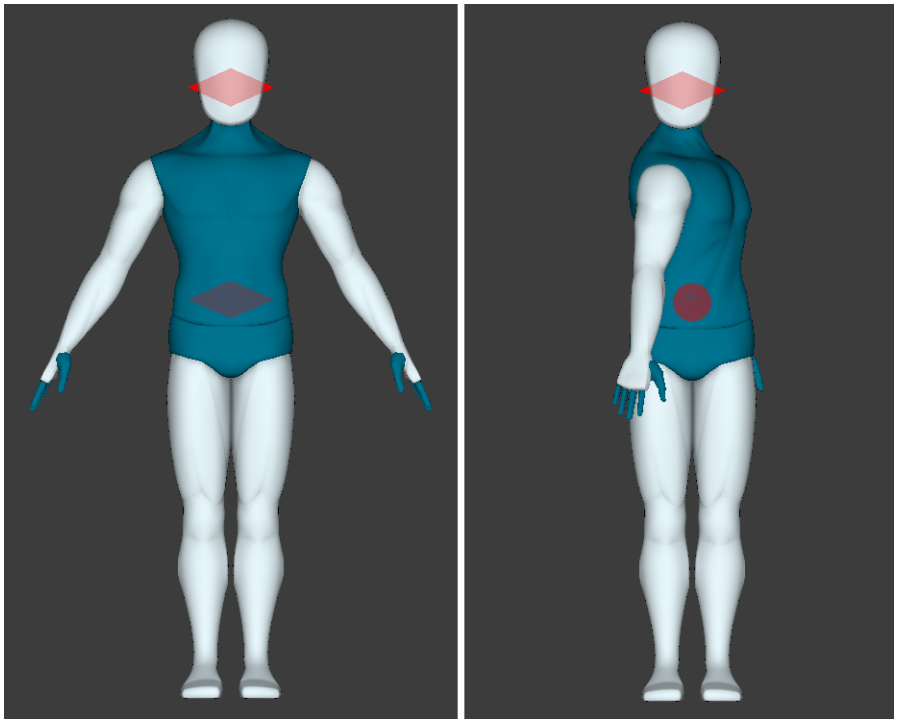


FIGURE 7 – Twist de bodyline

Notons cependant que l'implémentation actuelle du twist fonctionne pour les bodylines courtes (bras, buste ...) mais est plus difficile à appréhender pour les plus longues bodylines. De plus, le choix de l'axe de twist pose des problèmes pour certains contrôleurs (par exemple, le twist au niveau du pied ne correspond pas toujours à ce à quoi on s'attendrait).

2.6 Calcul d'une ligne d'action à partir d'une bodyline

Le principe de l'algorithme de pose est de bouger les contrôleurs d'une bodyline à partir d'une ligne d'action dessinée. Mais on peut aussi s'intéresser au problème inverse : **peut-on déterminer la ligne d'action à partir d'une pose et d'une bodyline données ?** Cet outil pourrait être utile dans un cadre d'analyse de pose (pour connaître les lignes d'action qu'il faudrait dessiner pour obtenir une pose spécifique). Dans le cas idéal, la ligne d'action calculée devrait laisser le modèle inchangé si on l'appliquait.

Pour répondre à cette question, nous avons essayé de déterminer la ligne d'action la plus proche de la bodyline donnée en utilisant la même méthode que celle illustrée en 2.3.1.

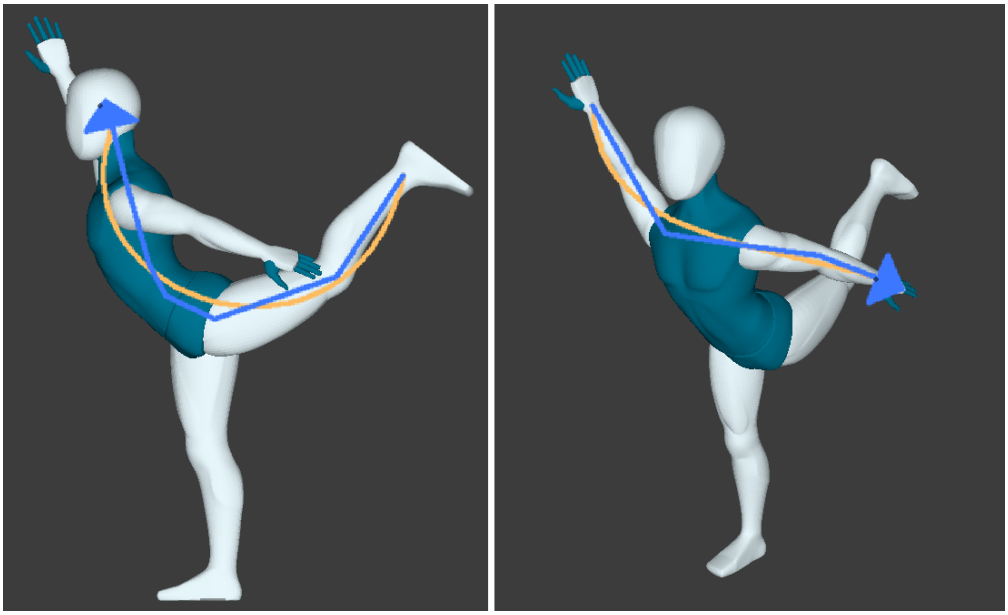


FIGURE 8 – Calcul de lignes d'action (en jaune) à partir de bodylines

Cette méthode a cependant un certain nombre de limites :

- La ligne d'action calculée sera toujours en « C » ou en « S », ce qui ne correspond pas toujours à la forme de la bodyline. On peut utiliser un modèle plus complexe de ligne d'action (par exemple une courbe de Bézier avec un degré plus élevé ou utiliser un ensemble de courbes de Bézier cubique) mais cela rend le calcul plus compliqué ;
- La pose d'une bodyline résulte souvent du dessin de plusieurs lignes d'action, selon des points de vue différents, or notre méthode calcule la ligne d'action selon un seul point de vue.

3 Pose de plusieurs modèles par lignes d'action

Cette partie expose et détaille le travail effectué concernant **la pose multi-modèles, sans et avec contacts**. Par souci de simplicité, nous parlerons par la suite de personnages, même si les outils développés sont valables sur n'importe quel type de modèles.

3.1 Scènes multi-personnages

Il est nécessaire d'adapter un peu l'outil de pose pour le faire fonctionner sur des scènes multi-personnages, car il faut savoir quel personnage est concerné par les processus de sélection de bodyline et de pose. Les modifications sont cependant simples, car seul le processus de sélection de bodyline change. L'idée est la suivante :

— On « simule » pour chaque personnage une sélection de bodyline ;
— On associe à chaque bodyline sélectionnée un **score**, qui dépend principalement de la distance entre les deux extrémités du sketch et de la bodyline. Une bodyline proche du sketch se traduira donc par un score faible ;
— On choisit ensuite le personnage dont le score est le plus faible, qui est alors défini comme le **personnage courant** ;
— Tous les processus (pose, twist ...) seront alors opérés sur ce personnage courant, tant qu'on ne change pas de bodyline.

3.2 Pose multi-personnages sans contacts

Les modifications apportées précédemment restent assez limitées, car on ne peut poser qu'un personnage à la fois. Or il serait très intéressant de poser simultanément plusieurs personnages. Nous avons donc développé :

— Une méthode permettant de sélectionner une bodyline pour chaque personnage à l'aide d'un seul sketch ;
— Une méthode permettant de poser toutes les bodylines sélectionnées en utilisant la même ligne d'action (**pose synchronisée**) ;
— Une méthode permettant de poser toutes les bodylines sélectionnées avec une seule ligne d'action, mais en associant à chaque bodyline une partie de cette ligne d'action (« **pose partagée** »).

3.2.1 Sélection multi-bodylines

Nous avons adapté le processus de sélection de bodylines pour permettre la sélection simultanée de bodylines en n'utilisant qu'un seul sketch. L'idée est de découper le sketch de l'animateur et d'assigner chaque morceau au personnage le plus proche. On utilise ensuite ces morceaux pour sélectionner la bodyline de chaque personnage.

Le découpage du sketch se fait de manière uniforme (tous les morceaux ont la même longueur). C'est une méthode très simple, qui donne de bons résultats. On assigne ensuite à chaque personnage le morceau dont le milieu est le plus proche de la racine du personnage.

Cependant, certaines combinaisons de bodylines sont parfois difficiles à sélectionner avec un seul sketch (par exemple si on veut sélectionner les deux jambes pour chaque personnage). Nous avons donc ajouté un mode de sélection qui permet de sélectionner plusieurs bodylines simultanément avec plusieurs sketches : l'animateur maintient une touche et dessine autant de sketches que nécessaire. Cette étape remplace l'étape de découpage, le reste (assignation et calcul des bodylines) lui ne change pas.

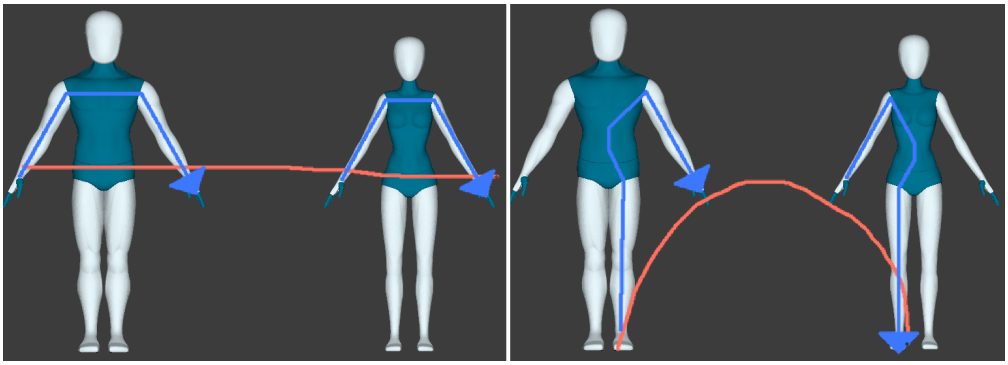


FIGURE 9 – Sélection simultanée de bodylines

3.2.2 Pose synchronisée

Dans ce mode, on utilise la ligne d'action dans son intégralité pour poser chaque bodyline. C'est un mode très utile pour poser des personnages de manière synchronisée (danse par exemple). Notons que ce mode n'est pas vraiment compatible avec le processus de translation de la racine : si l'option est active, tous les personnages viendront se coller à la ligne d'action, ce qui n'est pas forcément ce que l'on souhaite si on utilise ce mode.

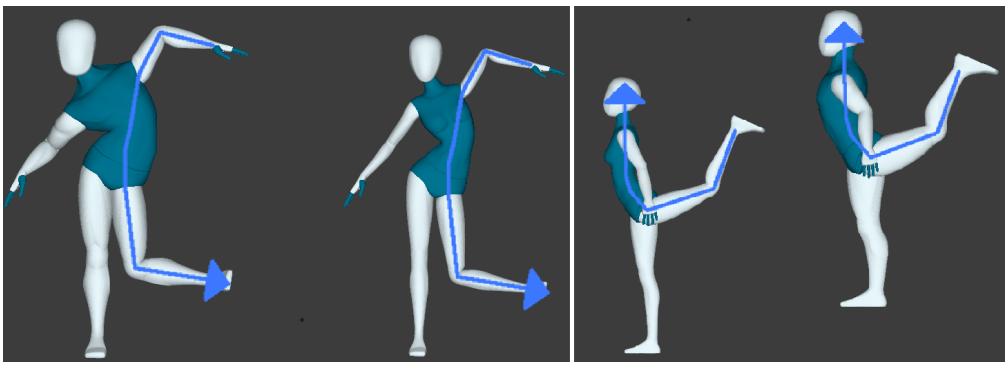


FIGURE 10 – Pose synchronisée

3.2.3 Pose « partagée »

Ce mode fonctionne de la même manière que la sélection de bodylines : on découpe la ligne d'action, on assigne chaque morceau au personnage le plus proche et on pose chaque personnage avec le morceau associé.

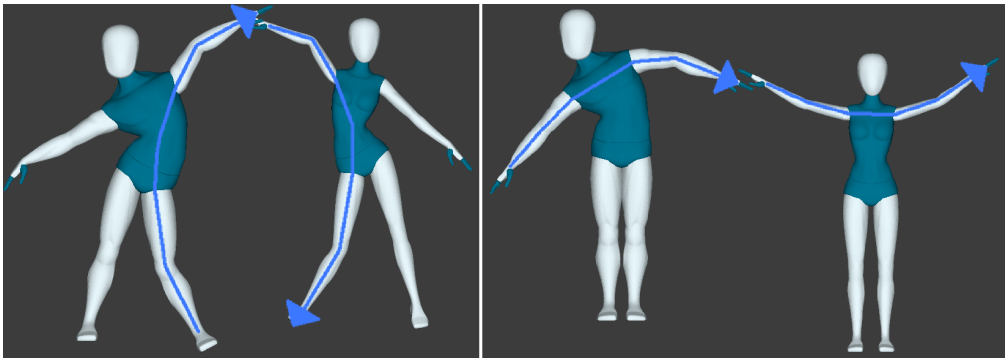


FIGURE 11 – Pose « partagée »

3.3 Pose multi-personnages avec contacts

Note : cette partie est exploratoire et n'a donné aucun développement définitif dans Rumba.

On peut désormais poser simultanément plusieurs personnages mais il n'y a aucune gestion des contacts. Par exemple, si deux personnages se tiennent la main, il est très difficile de poser les bras de ces personnages tout en gardant cette contrainte. Nous avons tenter d'explorer une solution innovante mais compliquée à mettre en place.

Cette solution a été en partie explorée par une stagiaire, Sarah Kushner, en 2017 (pour deux personnages seulement) et consiste à **fusionner les rigs des deux personnages**, pour ne former qu'un seul « méga-personnage ». Puisque les personnages sont fusionnés, on peut sélectionner et poser des bodylines comprenant des parties des deux personnages. Cependant, il n'est pas possible de fusionner les rigs directement dans Rumba, car c'est une opération très compliquée à réaliser (nécessité de reconfigurer entièrement la structure de la scène 3D) qui conduit à casser les rigs. Pour simplifier les tests, nous avons utilisé un rig de personnage basique (avec juste le squelette) et avons effectué les fusions sous Blender, avant d'importer le « méga-personnage » dans Rumba.

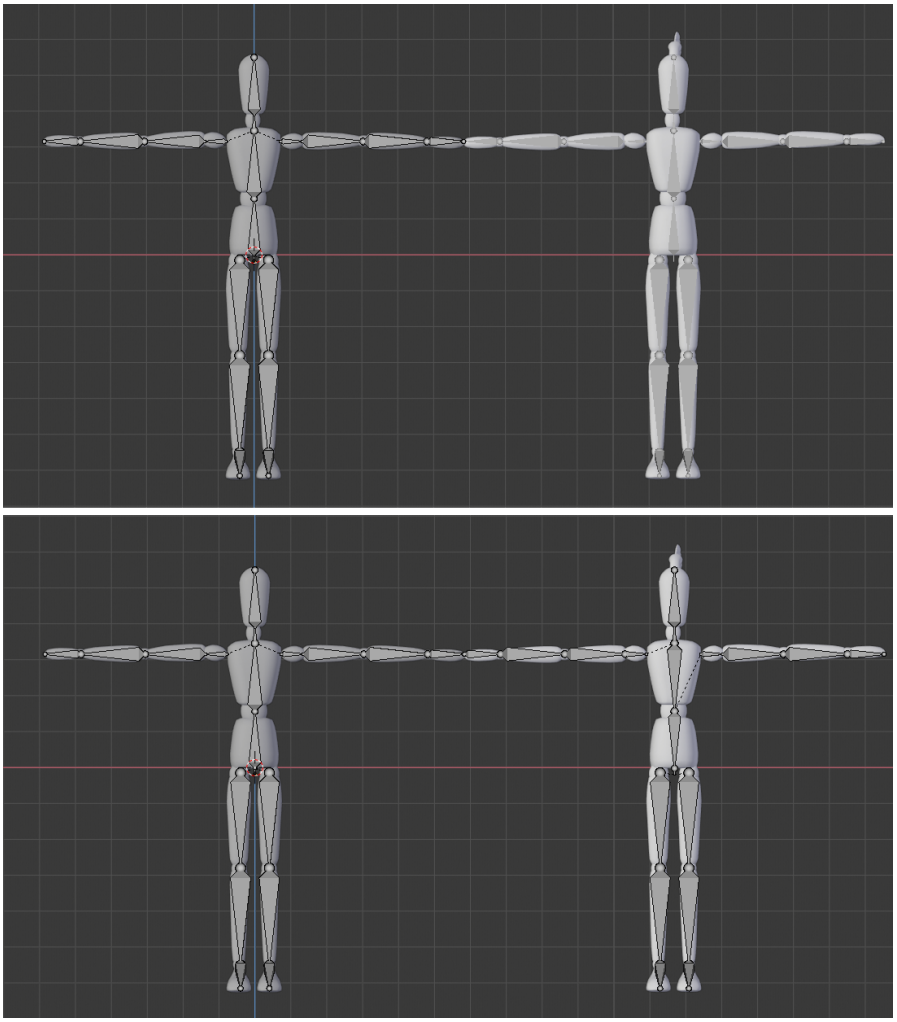


FIGURE 12 – Fusion des rigs de deux personnages se tenant la main

La figure 12 illustre un exemple de fusion de rigs pour deux personnages qui se tiennent la main. La fusion est effectuée en établissant un lien direct entre les deux mains (dans cet exemple, la main gauche du personnage de droite devient l'enfant de la main droite du personnage de gauche). Pour que le squelette résultant ait toujours une structure de squelette, il faut inverser le sens hiérarchique de certains bones du personnage de droite (ce qui explique pourquoi, sur la figure du bas, les bones de la main et du buste sont dans le sens « inverse »).

Avec cette nouvelle structure, on peut comprendre que le mouvement d'un des personnages (celui de gauche dans l'exemple) a une influence sur l'autre, compte-tenu du lien hiérarchique au niveau des mains. Un des personnages sera donc le *parent* et l'autre l'*enfant*, ce qui est cohérent dans certains cas, comme par exemple une danse où souvent un des personnages la mène.

La figure 13 illustre un exemple de pose de deux personnages qui se tiennent la main. Le fait de fusionner les deux mains en contact permet de les maintenir solidaires.

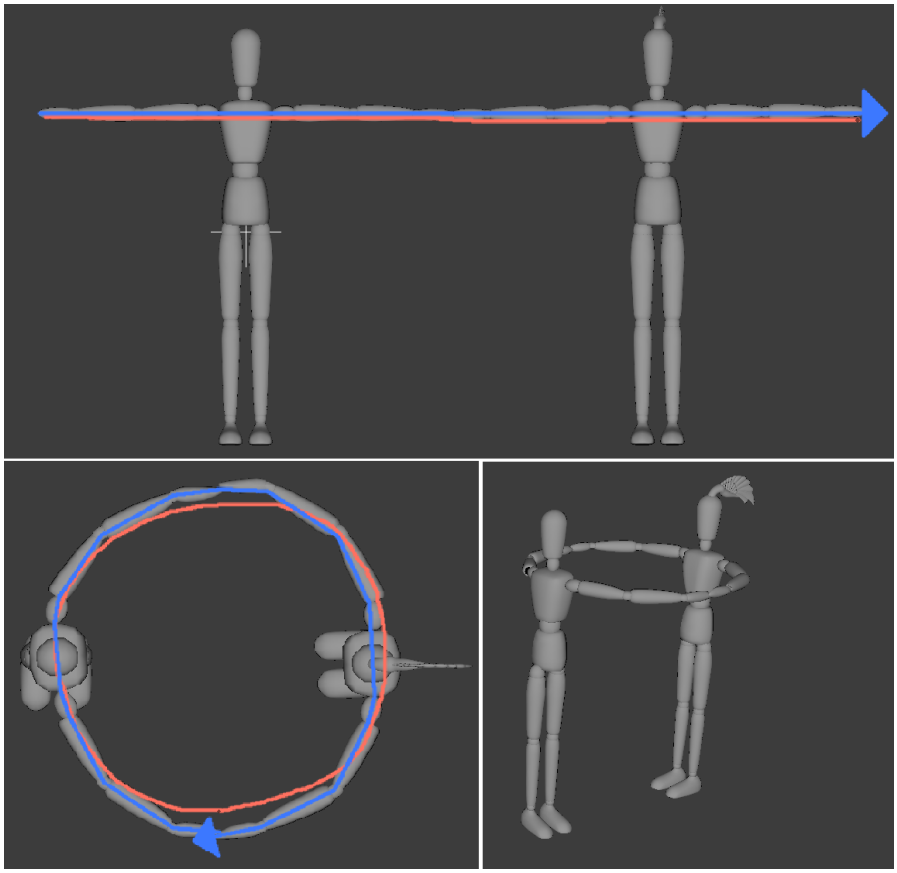


FIGURE 13 – Pose des deux personnages fusionnés

Comme évoqué précédemment, cette méthode, bien qu'intéressante, est difficile à mettre en pratique car la fusion revient à casser les rigs. De plus, importer les personnages fusionnés ne nous permet plus de les poser indépendamment. Cette solution oblige également à importer un rig différent pour chaque contact envisageable, ce qui n'est pas possible en pratique.

Une autre méthode envisageable revient à **utiliser des contraintes IK** pour assurer le contact entre deux contrôleurs. Cette méthode serait beaucoup plus facile à mettre en pratique car ces contraintes IK pourraient être activées ou désactivées dynamiquement selon les contacts que l'on souhaite conserver. Malheureusement, nous n'avons pas eu le temps d'implémenter cette méthode.

4 Animation de lignes d'action

Cette partie détaille le travail effectué dans Rumba concernant **l'animation de modèles par animation de lignes d'action**. Notons que ce travail est exploratoire et n'a pas donné d'implémentation définitive dans le logiciel.

Pour animer un modèle 3D, on utilise principalement la méthode *par keyframes* : on pose le modèle à des instants clés et on utilise les algorithmes d'interpolation 3D pour déterminer ses poses aux instants intermédiaires. Puisqu'on peut désormais poser le modèle avec des lignes d'ac-

tion, on peut se poser la question suivante : est-il plus intéressant de **conserver** les lignes d'action utilisées pour poser les instants clés et de déterminer les poses intermédiaires en **interpolant les lignes d'action** ?

4.1 Description

L'idée de l'interpolation de lignes d'action (qui est une interpolation 2D) est la suivante :

— On dessine, aux instants clés, des lignes d'action clés ;
— On interpole ensuite ces lignes clés pour former des **surfaces d'action** ;
— On utilise ces surfaces d'action pour déterminer les lignes d'action à appliquer à chaque instant.

L'interpolation 2D des lignes d'action en lieu et place de l'interpolation 3D permettrait de s'assurer que l'animation paraisse fluide pour un utilisateur. Elle a également l'avantage de séparer la spécification de l'animation du modèle sur lequel elle s'applique. Mais cette méthode d'interpolation pose également quelques problèmes :

— Il faut que l'interpolation des lignes d'action soit fluide pour que l'animation le soit ;
— Les lignes d'action clés n'ont pas forcément été dessinées selon le même point de vue. Il faudrait donc également prendre en compte le mouvement de la caméra entre les instants clés. Par souci de simplicité, on supposera que **le point de vue ne change pas**.

Nous avons exploré deux méthodes d'interpolation de courbes 2D : l'interpolation par **décomposition en angles-longueurs** et l'interpolation par **surface de Bézier bi-cubique**. D'autres méthodes d'interpolation (*patchs de Coons*) sont également envisageables.

4.2 Décomposition en angles-longueurs

Cette méthode est celle explorée par Guay *et al.* [5]. L'idée est de décomposer chaque courbe clé en segments. Chaque segment a une longueur L_i et fait un angle θ_i avec un axe de référence (en général, l'axe X). Une courbe est alors complètement déterminée par :

— La position x_0 d'un point « racine » de la courbe ;
— Les segments (θ_i, L_i)

Au lieu d'interpoler les positions 2D des courbes, on interpole les racines ainsi que les angles et les longueurs. On reconstruit ensuite les échantillons de la courbe interpolée en utilisant :

$$x_j = x_0 + \sum_{n=0}^{j} L_n R(\theta_n) X$$

où $R(\theta_n)X$ est l'image de l'axe de référence X par la rotation 2D d'angle θ_n. L'interpolation des différents éléments est effectuée au moyen de courbes d'Hermite, qui garantissent une interpolation C^1 (c'est-à-dire lisse).

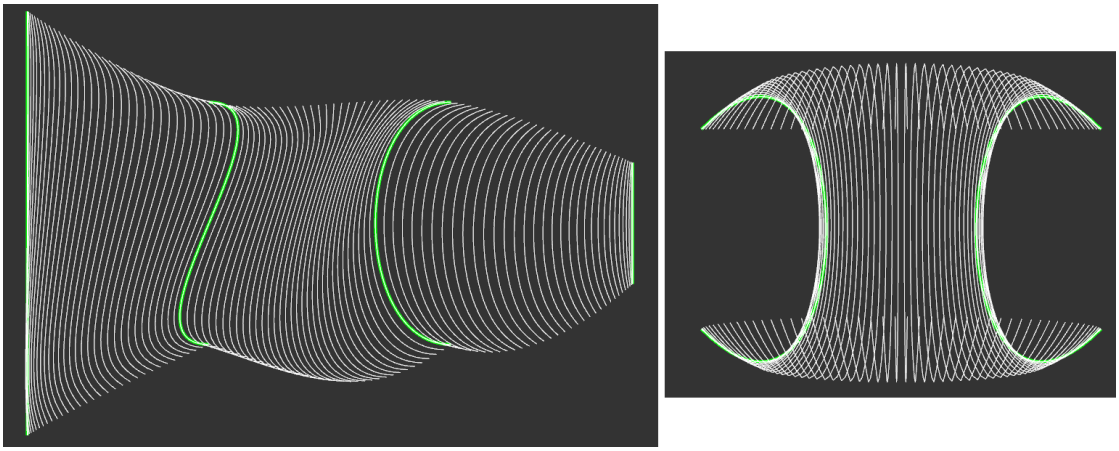


FIGURE 14 – Interpolation par décomposition en angles-longueurs

La figure 14 montre deux exemples d'interpolation de courbes 2D (les courbes clés sont en vert et les courbes interpolées en blanc). On observe dans les deux cas une interpolation lisse, qui tient bien compte des longueurs des courbes clés. La figure de droite illustre cependant un cas où l'interpolation sera cohérente mais aboutira à une « mauvaise » animation. On remarque en effet que les courbes semblent revenir sur elles-mêmes au niveau des extrémités, ce qui est mathématiquement normal car l'interpolation se fait selon le plus court chemin. Mais si on teste ce genre de lignes d'action sur un bras par exemple, on risque d'obtenir un poignet qui se tord dans le mauvais sens.

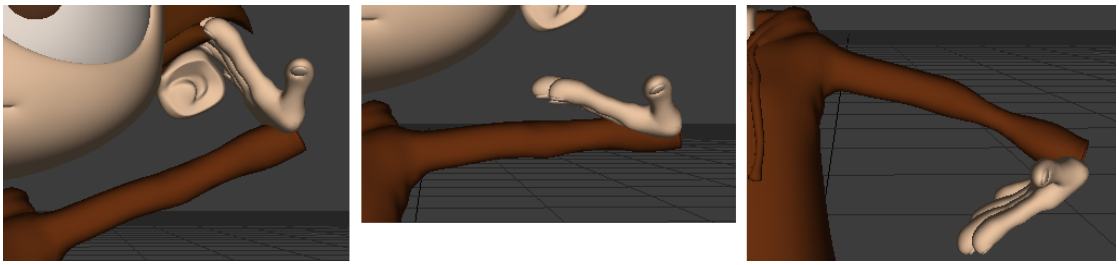


FIGURE 15 – Mauvaise interpolation du poignet

Ce problème arrive bien moins souvent avec l'interpolation 3D de Rumba car l'interpolation est effectuée sur les transformations locales des contrôleurs. Comme le poignet bouge peu par rapport au coude, l'interpolation est effectuée dans le bon sens. Actuellement, le seul moyen d'éviter ce cas de figure est d'ajouter une ligne d'action intermédiaire pour forcer l'interpolation à se faire dans le bon sens.

4.3 Interpolation par surface de Bézier bi-cubique

Puisque les lignes d'action se modélisent par des courbes de Bézier cubiques, on peut utiliser le même type de procédé pour les interpoler : on obtient une **surface de Bézier bi-cubique**.

Son expression est la suivante :

$$S(u,t) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_{i,3}(u)B_{j,3}(t)P_{ij} = \sum_{j=0}^{3}\left(\sum_{i=0}^{3} B_{i,3}(u)P_{ij}\right)B_{j,3}(t)$$

u est le paramètre d'abscisse curviligne, t le temps et les P_{ij} sont les points de contrôle de la surface. La deuxième expression de S est intéressante car elle indique qu'un point d'une surface de Bézier peut être évalué par une courbe de Bézier cubique, dont les points de contrôle ont aussi été évalués par des courbes de Bézier cubique. Cela nous permet de réutiliser les algorithmes de calcul développés pour les courbes de Bézier.

Cela pose cependant un problème : en pratique, pour pouvoir interpoler deux lignes d'action clés, on aura besoin de deux courbes supplémentaires, qui serviront à définir les points intermédiaires pour évaluer les points de la surface de Bézier. Le calcul d'un point $S(u,t)$ se fait alors de la manière suivante :

— On évalue en u les quatre courbes de Bézier (les deux courbes clés et les deux courbes intermédiaires) pour obtenir quatre points. On fait d'abord l'évaluation en u car ces courbes décrivent l'état de la ligne d'action à des instant fixes ;

— Ces quatre points sont ensuite utilisés comme points de contrôle d'une courbe de Bézier, que l'on évalue en t pour obtenir $S(u,t)$.

Nous avons imaginé des méthodes permettant de s'affranchir de dessiner ces courbes intermédiaires (par exemple, utiliser les vecteurs orthogonaux aux tangentes des courbes de Bézier clés pour calculer les points intermédiaires), sans toutefois aller plus loin dans l'implémentation et les tests.

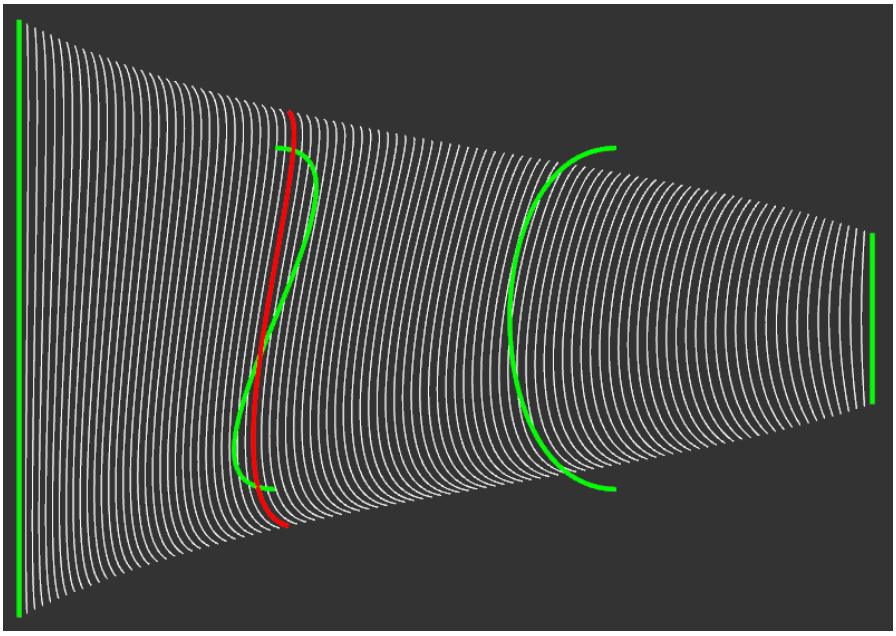


FIGURE 16 – Interpolation par surface bi-cubique de Bézier

La figure 16 illustre bien le rôle que jouent les courbes intermédiaires. Elles ne font pas partie de la surface (tout comme les points de contrôles intermédiaires d'une courbe de Bézier) mais influencent son évolution (la courbe en rouge a une forme proche de la courbe verte en « S »).

On remarque une évolution moins brusque par rapport à la méthode par angles-longueurs, ce qui peut rendre l'animation résultante plus agréable à regarder.

5 Animation par dessin de trajectoires

Cette partie expose et détaille toutes les fonctionnalités implémentées dans Rumba concernant **l'animation par dessin de trajectoires**. L'objectif est ici de pouvoir animer les contrôleurs d'un modèle 3D à l'aide de trajectoires 2D. L'édition ne se fait plus à un instant donné mais sur un **intervalle de frames**.

L'outil fonctionne de la manière suivante :

1. L'animateur sélectionne le contrôleur dont il souhaite éditer la trajectoire. Il peut également sélectionner un intervalle d'édition à l'aide de la *timeline* de Rumba ;
2. Il dessine ensuite la nouvelle trajectoire du contrôleur ;
3. Ce dessin est interprété pour déterminer, **à chaque frame de l'intervalle d'édition**, la nouvelle position du contrôleur ;
4. On utilise finalement ces nouvelles positions pour calculer les nouvelles courbes d'animation du contrôleur.

5.1 Visualisation des trajectoires

Visualiser la trajectoire d'un contrôleur est important, car elle donne une indication sur la manière avec laquelle l'animateur va pouvoir l'éditer. Nous avons donc implémenté trois modes de visualisation d'une trajectoire, inspirés par SketchiMo [2].

5.1.1 Espace global

Il s'agit du mode le plus classique, dans lequel on affiche la trajectoire du contrôleur dans l'espace 3D. Rumba propose de base ce mode de visualisation, appelé **motion trail**.

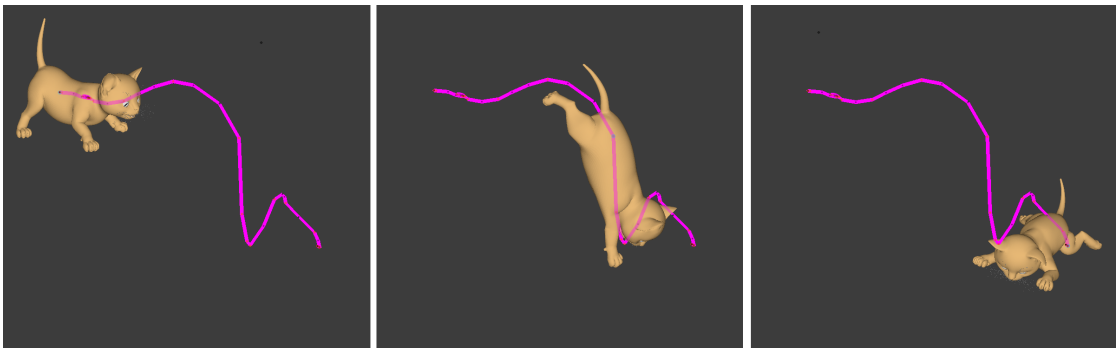


FIGURE 17 – Trajectoire en espace global

5.1.2 Espace local

Il est souvent plus intéressant de visualiser la trajectoire d'un contrôleur non pas de manière absolue (espace global) mais **relativement à un autre contrôleur**. On suppose dans ce cas que ce contrôleur « parent » est fixe, ce qui permet de s'affranchir de ses mouvements. Par exemple, la trajectoire d'une main pourra être perçue comme plus naturelle si on l'affiche par rapport à l'épaule, et non pas de manière absolue.

Nous avons modifié le motion trail de Rumba pour prendre en compte ce mode de visualisation. Par défaut, lorsqu'on sélectionne plusieurs contrôleurs, Rumba affiche les trajectoires de tous les contrôleurs en espace global. Nous avons ajouté un bouton qui, s'il est activé, affiche à la place les trajectoires des contrôleurs (sauf le dernier) relativement au dernier contrôleur sélectionné.

5.1.3 Espace dynamique

La trajectoire de certains contrôleurs peut être parfois difficile à visualiser, surtout pour des mouvements sur place. Pour résoudre ce problème, les auteurs de SketchiMo [2] ont conçu un nouveau mode d'affichage : l'espace dynamique. Dans cet espace, le temps est plus ou moins « étiré » (en fonction d'un paramètre réglé par l'animateur) afin de faciliter la visualisation de ces trajectoires.

Nous avions dans un premier temps implémenté le mode d'affichage de SketchiMo [2], mais il nécessite l'utilisation de la racine du modèle 3D. Le motion trail de Rumba propose également cette fonctionnalité, qui a l'avantage d'être indépendant du modèle affiché. Nous avons donc décidé de nous greffer dessus.

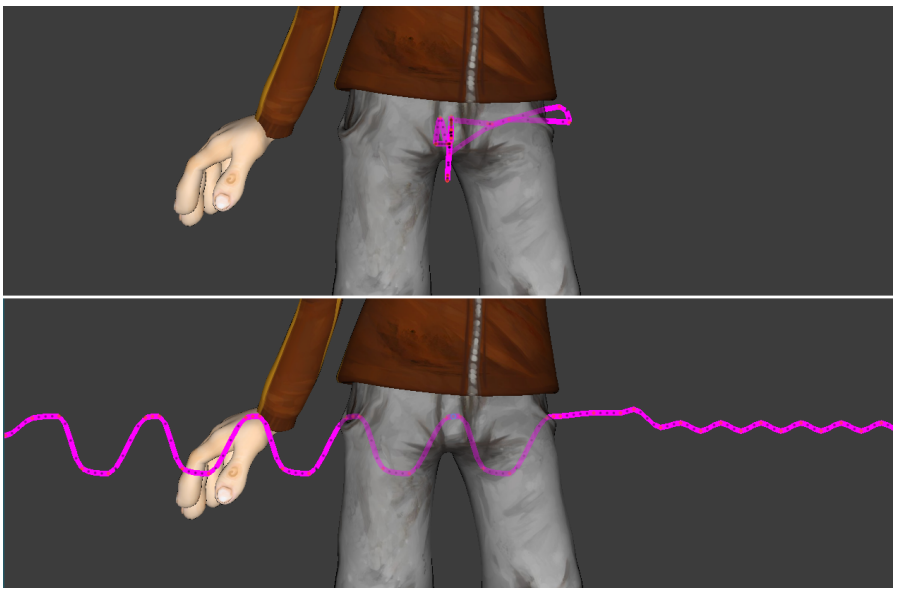


FIGURE 18 – Trajectoire du bassin en espace global puis dynamique

5.1.4 Espace caméra

Par défaut, les trajectoires sont affichées selon le point de vue de la caméra native de Rumba, celle utilisée pour naviguer dans la scène 3D. Or le rendu final de l'animation est en général fait selon le point de vue d'une autre caméra, souvent placée et animée avant même le modèle 3D.

Si on se place selon le point de vue de cette caméra et qu'on affiche une trajectoire, on la verra bouger, car le motion trail ne tient pas compte de ses mouvements. Ce n'est pas la trajectoire à laquelle un animateur ou un spectateur s'attendrait. Nous avons donc modifié le motion trail pour que cela soit le cas : maintenant, la trajectoire est toujours calculée selon **le point de vue de la caméra active dans le viewport**, et tient compte de ses éventuels mouvements.

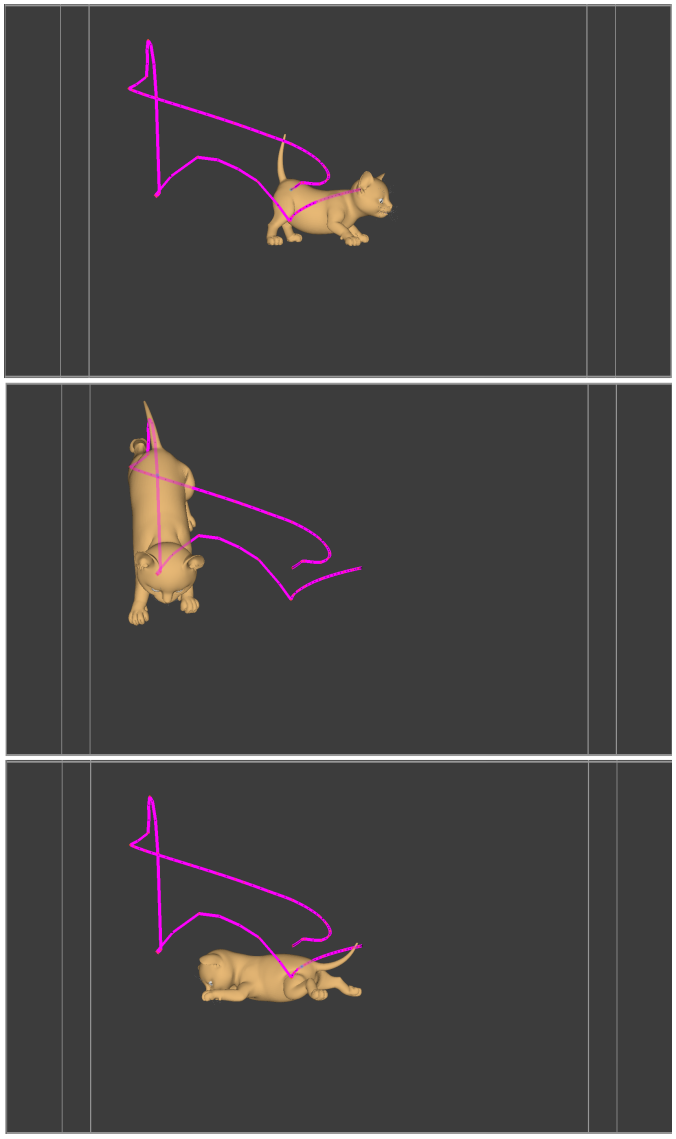


FIGURE 19 – Trajectoire de la figure 17 en espace caméra. La trajectoire reste fixe pendant l'animation

5.2 Interprétation du sketch et rythme

La première étape consiste à associer à chaque frame de l'intervalle d'édition un point du sketch, qui sera par la suite utilisé pour calculer la nouvelle position du contrôleur à cette frame. Pour cela, nous avons imaginé trois méthodes :

- On peut échantillonner la courbe de manière **uniforme**. C'est une méthode très simple, qui donne d'assez bons résultats. Mais elle produit une animation à vitesse constante, ce qui est dommage car on aimerait donner du rythme à la trajectoire en adaptant la vitesse du tracé ;
- Pour intégrer le rythme au tracé, on peut se baser sur le mode de fonctionnement du système d'acquisition du sketch, qui récupère la position de la souris à intervalles de temps réguliers. Les points seront alors proches si on dessine lentement et plus éloignés si on dessine vite. Cela nous permet de construire une fonction de « mapping » associant à un point du sketch (représenté par son abscisse curviligne) le point équivalent tenant compte du rythme.
  La figure 20 est un exemple d'une telle fonction. La courbe évolue « rapidement », puis est plus lente avant de repartir : elle représente en fait un tracé rapide puis lent puis de nouveau rapide ;
- Pour intégrer le rythme au tracé, nous avons également mis au point un **record mode** : l'animation est jouée au moment où l'animateur commence à dessiner et la position de la souris est enregistrée à chaque frame. Ce mode permet à l'animateur de régler plus précisément son timing, mais est encore expérimental : pour des scènes complexes, on observe des latences au moment d'enregistrer la trajectoire, ce qui fait que de nombreux échantillons ne sont pas enregistrés.

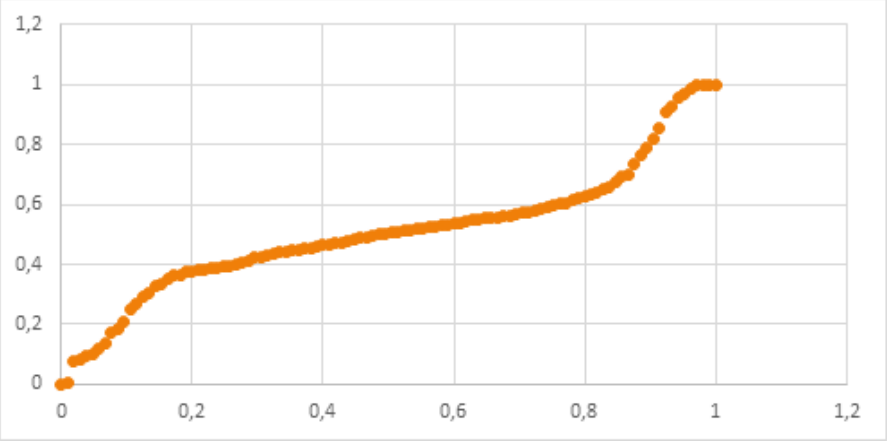


FIGURE 20 – Fonction de rythme

5.3 Calcul des nouvelles positions du contrôleur

La deuxième étape consiste à calculer, pour chaque frame de l'intervalle d'édition, la nouvelle position du contrôleur, à l'aide du point de la courbe dessinée qui lui a été associé. On se confronte alors au même problème que pour la pose : comment calculer une nouvelle position 3D avec un point 2D (sans information de profondeur) ? Nous avons fait le même choix que pour la pose : restreindre les mouvements (translation et rotation) à un plan :

— Dont la normale est la direction de la caméra ;
— Et qui passe par l'ancienne position du contrôleur (à la frame considérée).

Pour calculer la nouvelle position du contrôleur, il suffit alors de dé-projeter le point 2D associé dans ce plan.

Comme pour la visualisation, il est possible d'éditer la trajectoire en espace local ou dynamique et selon le point de vue de la caméra sélectionnée dans le viewport. La trajectoire dessinée sera alors interprétée dans l'espace correspondant. Par exemple, le calcul de la nouvelle position d'un contrôleur en espace dynamique est fait de la manière suivante :

— On calcule l'ancienne position du contrôleur, **en espace dynamique**. Appelons-le P ;
— On dé-projette le point de la courbe associé au contrôleur dans le plan dont la normale est la direction de la caméra et passant par P. On a alors la nouvelle position du contrôleur, toujours en espace dynamique ;
— On calcule finalement la nouvelle position globale du contrôleur en inversant la transformation effectuée lors de la première étape.

5.4 Calcul des nouvelles courbes d'animation

Une fois toutes les nouvelles positions calculées (que l'on appelera par la suite **contraintes**), on peut mettre à jour les courbes d'animation du contrôleur. C'est un processus assez facile ; il ne faut juste pas oublier de convertir les contraintes *globales* (dans l'espace de la scène) en contraintes *locales* (dans le repère du parent du contrôleur) car les courbes d'animation définissent le mouvement du contrôleur par rapport à son parent.

Mais ajouter une contrainte à chaque frame n'est pas la bonne solution car cela va créer une clé par frame, rendant l'édition future impossible (Rumba n'interpole pas entre deux frames consécutives donc un changement trop brusque conduira à des « sauts » dans l'animation). Nous avons alors développé des outils de simplification permettant de respecter au maximum les contraintes tout en limitant le nombre de clés à ajouter.

Nous avons imaginé deux solutions, dépendant de paramètres choisis par l'animateur :

— Le contrôleur avait-il déjà des clés ?
— Si oui, peut-on en ajouter/supprimer ou faut-il garder uniquement les clés déjà posées ?

La première solution consiste à utiliser l'algorithme de Ramer-Douglas-Peucker (RDP) [6], un algorithme classique de simplification de courbes polygonales. En l'appliquant aux courbes d'animation, on peut ainsi déterminer les clés à conserver pour que la courbe résultante ressemble plus ou moins à la courbe initiale, selon une tolérance paramétrée par l'animateur (0 % : on garde tous les points ; 100 % : on ne garde que les extrémités). Nous avons légèrement modifié l'algorithme pour qu'il puisse traiter plusieurs courbes en parallèle, afin que toutes les courbes d'animation aient les mêmes clés.

L'algorithme RDP est très efficace mais il ne conserve pas les clés déjà posées par l'animateur. On peut l'adapter pour qu'il le fasse, mais cela ne l'empêchera pas d'en ajouter, conduisant à l'accumulation de clés souvent inutiles. Nous avons donc imaginé une solution conservant uniquement les clés déjà posées, qui se base sur le mode d'interpolation entre deux clés. Dans la plupart des cas, il est spline : chaque clé possède deux tangentes gauche/droite qui permettent de contrôler l'allure de l'interpolation avec les clés précédente/suivante. Puisqu'on connaît les contraintes entre deux clés, on peut essayer de trouver les tangentes optimales qui permettront d'approcher au maximum ces contraintes. Cette méthode est très efficace, du moment que le nombre de clés est suffisant pour bien approximer la courbe dessinée (p.ex. on ne peut pas

représenter une trajectoire en forme de vague avec deux ou trois clés, car le mode d'interpolation ne permet de représenter que des formes « simples »).

Pour déterminer les tangentes optimales, on utilise une légère variante de la méthode illustrée en 2.3.1 : puisqu'on connaît les valeurs que doivent prendre les clés, seuls les deux points intermédiaires de la courbe de Bézier doivent être déterminés. Une fois la courbe de Bézier calculée, on calcule les tangentes de la manière suivante :

$$T_0 = 3(P_1 - P_0), T_1 = 3(P_3 - P_2)$$

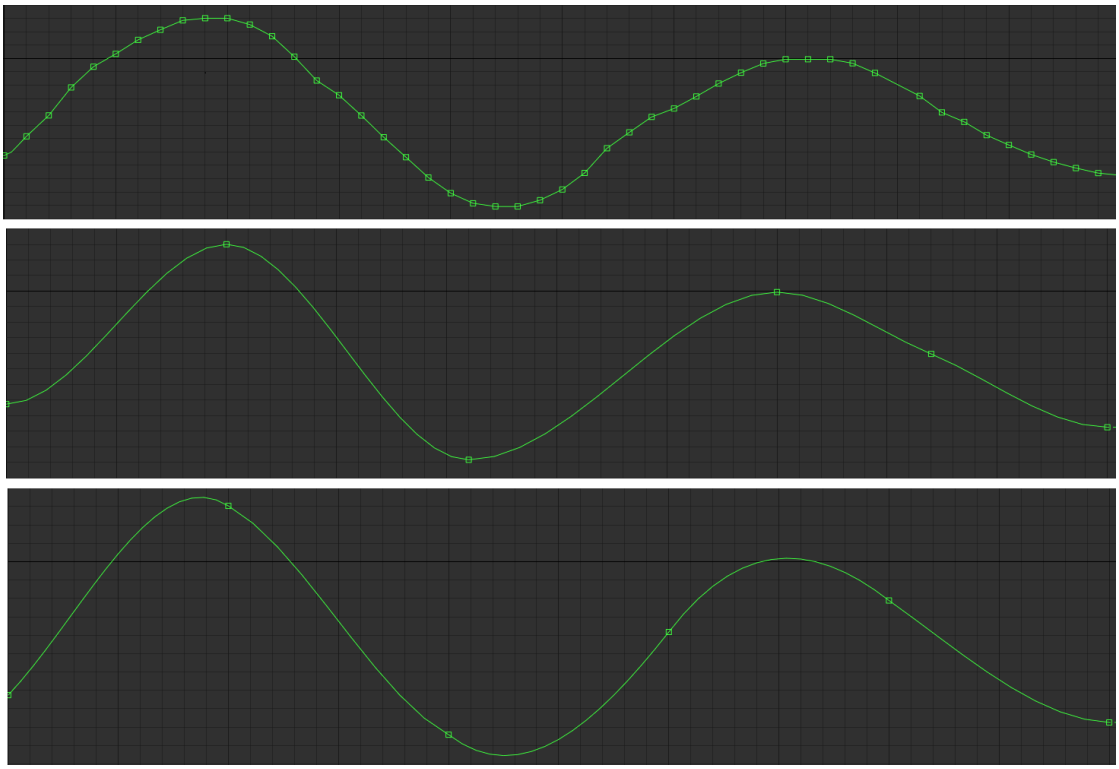


FIGURE 21 – Courbe non simplifiée, simplifiée par RDP [6] et par approximation des tangentes

5.5 Cas des chaînes IK

Dans certains cas, le contrôleur est associé à une chaîne de cinématique inverse (par exemple la main liée à la chaîne main, coude, épaule). On aimerait alors que le mouvement du contrôleur entraîne tous les contrôleurs de la chaîne.

- Si on anime un contrôleur IK, il n'y a rien à faire. Rumba gère nativement la chaîne IK associée au contrôleur IK et se charge de calculer les nouvelles positions de tous les contrôleurs de la chaîne ;
- Si on anime un contrôleur FK, on aimerait également que le mouvement du contrôleur entraîne ceux de la chaîne IK. Puisque Rumba ne propose pas cette fonctionnalité nativement (les contrôleurs FK sont faits pour être manipulés indépendamment), nous avons implémenté notre propre solveur IK.

Le solveur que nous avons choisi est FABRIK [1], qui est un algorithme de cinématique inverse populaire. L'implémentation de FABRIK a plusieurs avantages :

— Contrairement à Rumba (pour lequel les chaînes IK sont en général incluses dans le rig et ne sont pas modifiables), on peut changer comme on le souhaite les chaînes IK associées à une extrémité du personnage. Par exemple, on a ajouté une option indiquant si la chaîne IK associée à une main doit s'arrêter à l'épaule ou si elle doit s'arrêter à la racine (dans ce cas, le mouvement de la main entraînerait le bras mais aussi le buste) ;

— Contrairement au solveur IK de Rumba, FABRIK peut gérer plusieurs chaînes IK en parallèle, ce qui peut être utile quand on veut respecter plusieurs contraintes à la fois. Par exemple, si on dessine une première trajectoire sur une main puis une deuxième sur l'autre main, on aimerait que la première trajectoire reste respectée (quitte à trouver un compromis entre les deux contraintes de trajectoire).

L'outil d'animation fonctionne finalement de la manière suivante :

1. L'animateur sélectionne un contrôleur ;
2. Il dessine sa nouvelle trajectoire, qui est interprétée pour calculer ses nouvelles positions ;
3. Si le contrôleur n'est pas lié à une chaîne IK (contrôleur du bassin par exemple) ou s'il s'agit d'un contrôleur IK (dans ce cas, on laisse Rumba faire le travail), on calcule ses nouvelles courbes d'animation et on a terminé ;
4. Sinon, on applique FABRIK **à chaque frame de l'intervalle d'édition** pour calculer les nouvelles positions de tous les contrôleurs de la chaîne IK. On peut ensuite déterminer les nouvelles courbes d'animation de ces contrôleurs. Il ne faut cependant pas oublier de convertir les contraintes de position en contraintes de rotation : pour positionner correctement un contrôleur dans ce cas de figure, il faut bouger en rotation son parent (dans la chaîne IK). Autrement dit, pour mettre à jour les positions d'un contrôleur de la chaîne IK, on modifie les courbes d'animation de rotation de son parent.

6 Animation et *ghosting*

Rumba propose de manière native deux fonctionnalités intéressantes :

— Le *ghosting*, qui permet d'afficher sur un même viewport plusieurs keyframes ;

— Le *screen space offset* qui, comme décrit en 5.1.3, permet de « dilater » le temps, afin de faciliter la visualisation des ghosts.

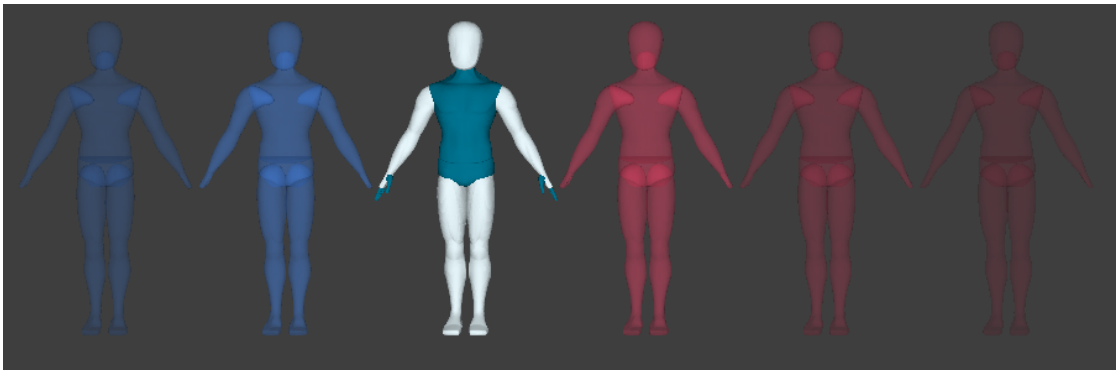


FIGURE 22 – *Ghosting*

En adaptant les outils de pose et de trajectoires, on peut alors réaliser un outil qui permet de poser et d'animer un modèle sur plusieurs keyframes sans avoir à changer de frame. Son fonctionnement est le suivant :

- L'animateur pose des clés sur son modèle (ou importe une animation). Il active ensuite le ghosting et le screen space offset si besoin ;
- Il sélectionne ensuite une bodyline sur le ghost qu'il souhaite poser ;
- Il dessine ensuite la ligne d'action, qui sera alors utilisée pour poser la bodyline du ghost sélectionné ;
- L'animateur peut aussi sélectionner un contrôleur, afficher si nécessaire son motion trail et dessiner sa nouvelle trajectoire.

Adapter l'outil de pose requiert peu de changements :

- Pour sélectionner le ghost le plus proche, on « simule » une sélection de bodyline pour chaque ghost et on utilise le score défini en 3.1 pour déterminer le ghost le plus proche du sketch d'entrée ;
- Comme son nom l'indique, le screen space offset applique aux ghosts un offset. Il ne faut donc pas oublier d'appliquer cet offset dans tous les processus (sélection, pose) afin d'avoir des résultats cohérents.

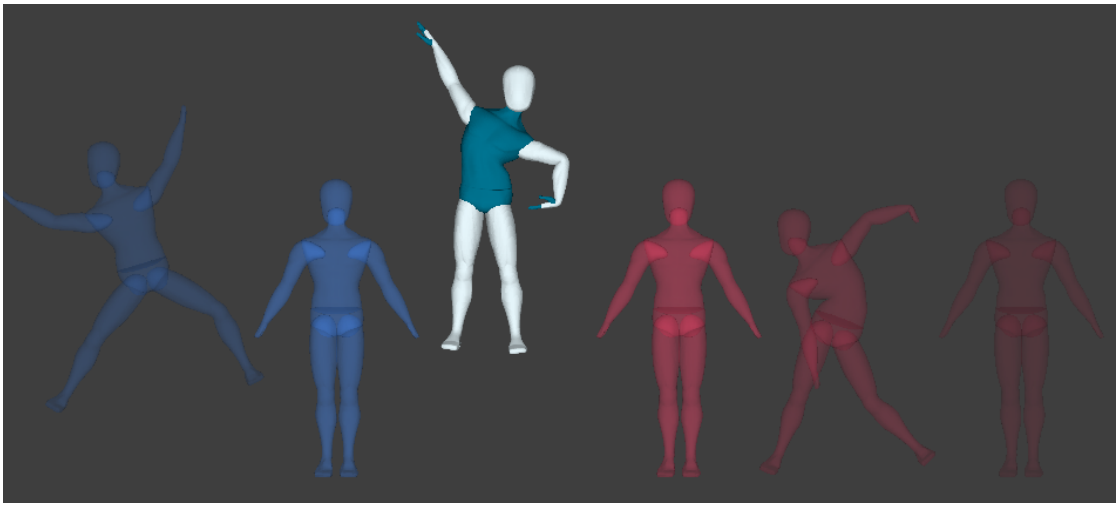


FIGURE 23 – *Ghosting* et poses

Nous avons également ajouté un bouton permettant de **propager la pose d'un ghost aux ghosts suivants**, dans le cas où on aimerait l'utiliser comme référence pour poser le ou les ghosts suivants.

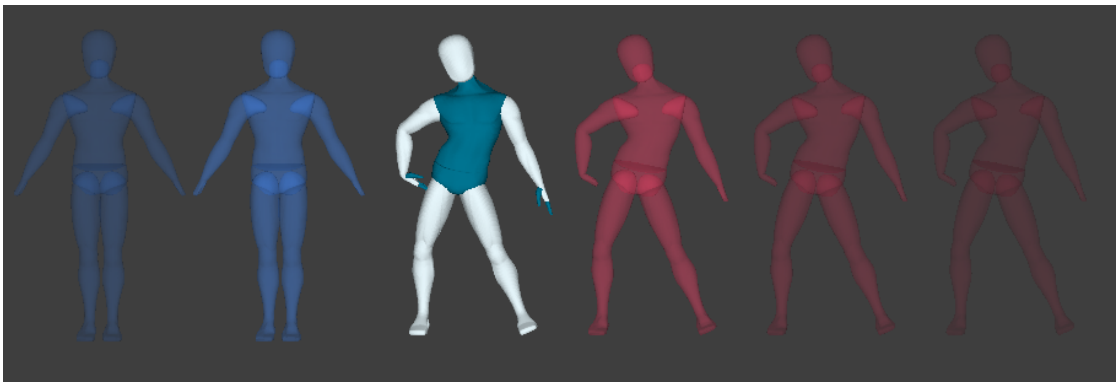


FIGURE 24 – Propagation d'une pose aux ghosts suivants

7 Retours des animateurs

7.1 Visite chez TeamTO, Septembre 2017

Nous avons présenté l'outil de pose à certains animateurs du studio de TeamTO à Valence le 25/09/2017 (un peu moins d'un an après le début du projet) et leur avons proposé de le manipuler, afin d'avoir leurs retours. L'équipe de test était constituée de 6 animateurs aux

profils variés (animateurs juniors et séniors, sup anim) provenant de l'animation 3D (mais aussi de l'animation 2D pour l'un d'entre eux). L'objectif était de savoir :

— Si l'outil était facile à utiliser, intuitif ;

— Et si les animateurs voyaient un intérêt à utiliser l'outil de pose pour leurs productions.

Nous voulions également voir comment l'outil de pose était utilisé en conditions réelles, pour déterminer les bugs à fixer et les fonctionnalités à ajouter/supprimer.

Globalement, les retours ont été **assez positifs** : les animateurs ont semblé très intéressés et enthousiastes à l'idée de tester l'outil. Alors qu'une session de test devait durer 20 minutes par animateur, les sessions ont finalement duré en moyenne plus de 40 minutes. Ils voient également l'intérêt que pourrait apporter l'utilisation de l'outil pour leurs productions, notamment celles où on peut trouver des personnages assez expressifs et avec de longues bodylines (des dragons par exemple). Take it Easy Mike a été cité mais la série Skylanders l'a aussi été (série avec beaucoup de combats, d'action ...).

Un animateur a cependant émis un peu plus de réserves, trouvant que l'outil donne parfois des résultats assez difficiles à prédire, compte tenu de son fonctionnement (rotations uniquement dans le plan d'action). Mais il pense que ça peut provenir du fait qu'il n'est pas encore très à l'aise avec l'outil.

Les animateurs ont apprécié la possibilité de pouvoir facilement switcher entre l'outil de pose et l'outil de manipulation traditionnelle des contrôleurs (pour par exemple affiner ou corriger certains contrôleurs).

Cette série de tests nous a permis d'avoir un certain nombre de remarques et d'idées de la part des animateurs, que nous avons pu ajouter :

— La première version de l'outil de pose n'indiquait pas le sens de sélection de la bodyline. C'est quelque chose qui a beaucoup gêné les animateurs, car ils devaient constamment se souvenir du sens dans lequel ils les avaient sélectionnées. L'ajout de la flèche pour indiquer le sens de sélection a été fait suite à leurs remarques ;

— Au début, les seules bodylines non-maximales que nous autorisions étaient les bodylines du type « un seul bras » ou « une seule jambe ». Mais les animateurs ont dès le départ essayé de manipuler des bodylines plus courtes (avant-bras, partie du buste ...). C'est ce qui nous a conduit dans le choix d'élargir le concept de bodyline, et d'autoriser la sélection de n'importe quel type de bodyline. Les animateurs ont également émis le souhait de manipuler des bodylines moins « classiques », permettant par exemple de poser le rig facial d'un personnage ;

— Nous avons également fait quelques ajouts suite aux retours des animateurs : sélection d'une bodyline non-maximale juste via la sélection de son extrémité, pose de sous-bodylines avec la même ligne d'action (pour poser les jambes en parallèle avec un seul trait) etc.

7.2 Retours d'un animateur sénior, Mars-Avril 2019

Valérian Daunis, animateur sénior chez TeamTO, a pu prendre en main tous les outils pendant deux mois et a pu réaliser deux séquences d'animation reproduisant les chorégraphies de deux courts extraits de comédies musicales américaines :

— Gene Kelly, *Singing in the rain*, extrait du film du même nom ;

— Cyd Charisse, Fred Astaire, *Dancing in the dark*, extrait du film *The Band Wagon*

Le suite de cette partie expose ses retours.

7.2.1 Outil de pose

C'est un outil ayant une approche 2D permettant de tracer rapidement des poses sans gestion de contraintes. Il permet de dessiner des lignes d'actions, elles sont interprétées par des rotations des contrôleurs selon la vue courante.

Dans un premier temps, il a été difficile d'intégrer cet outil dans le *workflow*, car cela nécessite une réflexion différente de celle pratiquée en production. Malgré ces difficultés, l'outil a paru plus intuitif au bout de quelques semaines de pratique.

L'outil va permettre d'avoir la forme la plus proche du tracé **mais ne respecte pas les contraintes anatomiques du rig**, c'est-à-dire que l'on peut avoir des rotations sur tous les axes des bras et des jambes (FK), ce qui conduit à :

— Une mauvaise interpolation entre deux keyframes avec des valeurs de rotations incohérentes ;
— Des mouvements du torse/COG (centre de gravité) qui surviennent quand on pose les bras/jambes ;
— Des twists sur les membres générant des artefacts de skin ;
— Une perte de productivité afin de corriger ces problèmes.

La gestion des contraintes est un point important pour un animateur qui pose quelques difficultés à l'outil, comme le fait de pouvoir poser un pied au sol. Ces limitations pour l'animateur restent cependant solubles, donc ne sont pas rédhibitoires pour l'avenir de l'outil.

Au-delà de ces limitations, l'outil est très prometteur, et une fois la gestion des contraintes terminée, celui-ci aura plusieurs avantages que ce soit en layout et/ou en animation :

— Une mise en place rapide des personnages (*rough*) ;
— Avec le *ghosting*, on peut éditer chaque keyframe et dessiner en temps réel les poses qui la précèdent ou succèdent ;
— Un système de twist de bodyline intéressant, permettant de répartir les rotations des contrôleurs autour de l'axe de la bodyline ;
— Un lock axis pour plus de permissivité de la part de l'outil (par exemple pour sculpter les doigts) ;
— Une transparence pour l'animateur car lorsqu'il quitte l'outil, celui-ci garde les états antérieurs des contrôleurs (follow, IK/FK, ...).

On peut étendre son utilisation pour dessiner/animer des parties du corps (cheveux, queue des animaux, moustaches, doigts ...).

7.2.2 Outil multi-personnages

L'outil permet de cumuler plusieurs bodylines entre les personnages, afin de dessiner une *shape* dans sa globalité ou de dessiner la *shape* de deux personnages de façon synchronisée. L'outil présente les mêmes limitations que celles du sketch, à savoir la gestion des contraintes. L'évaluation de l'outil a été effectuée sans en tenir compte.

On peut constater quelques soucis de sélection de bodyline, liés à des bugs (corrigeables avec plus de temps de développement). Une mise en pratique sur un extrait de *Dancing in the dark* a mis en lumière des problématiques similaires à celles rencontrées avec l'outil de sketch.

La plus grosse difficulté pour un animateur dans ce type d'exercice est la gestion des interactions entre les deux personnages. Il est nécessaire de pouvoir fixer et gérer ces interactions.

Le pose synchronisée permet de poser plusieurs personnages en utilisant la même bodyline, selon le rig/morphologie du personnage, le résultat varie mais reste assez semblable.

Il a été difficile d'intégrer l'outil au workflow, en effet l'absence des correctifs pour gérer les interactions entre les personnages diminue la précision dans le posing, et par conséquent demande

un temps de correction assez conséquent.

7.2.3 Outil de trajectoires

Cet outil permet de tracer la trajectoire et/ou d'éditer un contrôleur/dummy. Plusieurs options se présentent :

Trajectoire sans rythme

Ce type de tracé se focalise plus sur la forme de la trajectoire (visible avec le motion trail de Rumba), la gestion du rythme n'est pas présente, mais il permet néanmoins d'appréhender les trajectoires de manière cohérente avec le tracé. Lors de l'édition de trajectoire, l'option *Keep existing keyframes* (voir annexe A.2) permet de garder le timing existant tout en éditant la forme de la trajectoire.

Trajectoire avec rythme

Ce type de tracé est similaire au précèdent, sauf qu'il inclut la notion de rythme avec l'option *Keep drawing rhythm* (voir annexe A.2), cela permet de timer la trajectoire en fonction de la vitesse de tracé. L'inconvénient de ce mode est l'absence de temps réel, il n'est donc pas évident pour son utilisateur de voir le timing réel dans un intervalle préétabli. S'il est trop grand, notre timing va s'étaler et à l'inverse, s'il est trop petit, il va s'accélérer.

Record mode

Le « record mode » a été introduit afin de voir en temps réel où l'on se situe à la fois dans l'espace (trajectoire) et le temps (timeline qui défile). Le « record mode » commence à partir du moment où l'on trace la trajectoire et se termine dès que l'on s'arrête.

Une fois la trajectoire tracée, le curseur de la timeline revient sur la frame de départ, ce qui permet à l'animateur, s'il le souhaite, de recommencer son édition et/ou création instantanément sur la bonne frame.

Ce mode est très intéressant pour toutes les animations de *props* et placement de personnages (layout et animation). L'outil introduit une notion de tolérance (*RDP Tolerance* - voir annexe A.2), qui peut être associée à la notion de fidélité du tracé : plus la tolérance sera grande, moins il y aura de clés (donc moins de fidélité) et plus elle sera petite, plus grand sera le nombre de clés. La compatibilité avec les layers d'animation permet d'additionner les trajectoires à l'infini. Tout comme l'outil de pose, le lock axis permet de décomposer la trajectoire et de la gérer plus minutieusement.

Globalement l'édition de trajectoire permet de nettoyer totalement/partiellement les trajectoires, et d'aboutir à un résultat plus « smooth », mieux travaillé, de manière très rapide.

7.2.4 Résumé des remarques et solutions

Le tableau suivant résume les principales remarques faites par Valérian au cours de ses deux mois de tests ainsi que les solutions que nous avons pu apporter.

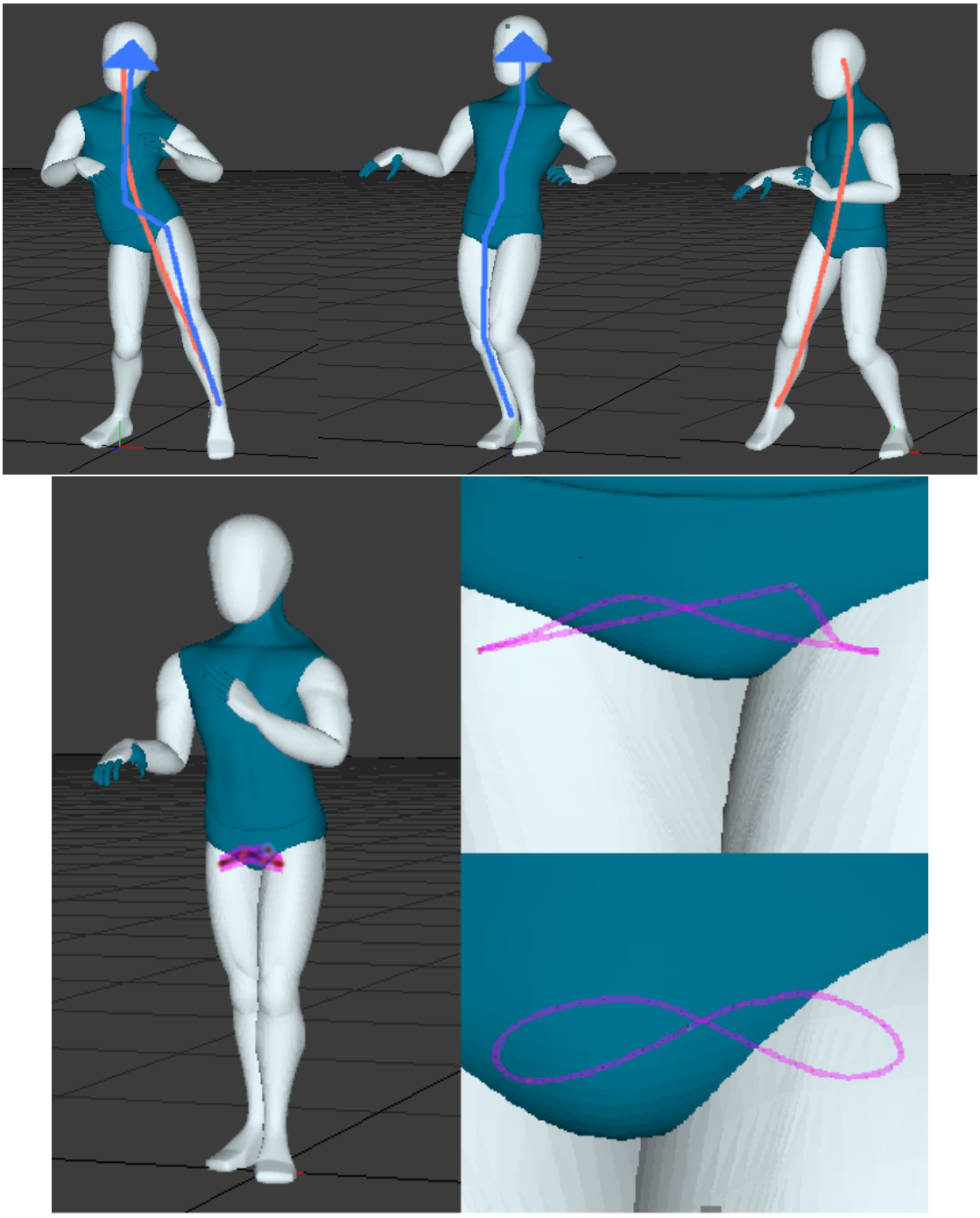
Remarques/Problèmes	**Solution(s)**
Pose : pas de gestion des contraintes d'environnement (IK)	Pour les extrémités qui ne sont pas dans la bodyline courante : rester en IK. Pour les extrémités qui sont dans la bodyline courante, pose sans contraintes puis restauration de la contrainte IK. Utilisation des pole vectors pour conserver au mieux la forme de la ligne d'action tout en respectant la contrainte.
Pose : manque de compatibilité avec les outils « traditionnels » de Rumba	Quand on active l'outil de pose, l'état contraint/libre de chaque extrémité dépend de celui du contrôleur IK associé. Si on effectue des changements sur ces états pendant l'utilisation de l'outil et qu'on le quitte, on restaure les états IK/FK avant l'activation de l'outil. L'objectif est de faire en sorte que tous ces changements d'état soient transparents pour l'animateur.
Pose & Trajectoires : les translations/rotations des contrôleurs peuvent se faire sur tous les axes, ce qui peut parfois être problématique (anatomiquement incorrect, interpolations ...)	Les contraintes anatomiques ne sont pour l'instant pas gérées. Pour s'assurer que seuls certains axes d'un contrôleur soient modifiés par une pose ou une trajectoire, on peut cependant utiliser le *lock axis*, qui remplace la direction de vue de la caméra par un axe calculé en fonction des axes que l'on veut bloquer. On peut demander à bloquer des axes globaux (ceux de la scène 3D) ou des axes locaux (propres au contrôleur).
Trajectoires : difficile de bien appréhender le rythme quand on dessine une trajectoire.	L'option *Keep drawing rhythm* (voir annexe A.2) n'est pas « temps réel » : il est donc difficile de bien gérer la vitesse de tracé en fonction de l'intervalle sur lequel on souhaire éditer la trajectoire. Á partir des remarques de Valérian, nous avons implémenté le « record mode » qui joue l'animation pendant qu'on dessine.
Trajectoires : pas de compatibilité avec les layers d'animation	Compatibilité désormais assurée. Si on édite une trajectoire sur un layer d'animation, on ne tient pas compte de l'animation éventuellement pré-existante, car l'objectif est de pouvoir par exemple ajouter une correction à l'animation de base (correction qu'on peut désactiver à la demande).

7.2.5 *Workflow* : Rumba

Le workflow permettant de réaliser une danse « Rumba » est le suivant :

1. Placer les pieds et le cog/root en parallèle ;
2. Sélectionner et poser une bodyline maximale, du pied à la tête ;
3. Ajuster au besoin pour préciser le posing ;
4. Dessiner les bodylines du/des bras ;
5. Afficher le motion trail ;
6. Éditer la trajectoire du cog en gardant le timing avec *Keep exising keyframes* (voir annexe A.2).

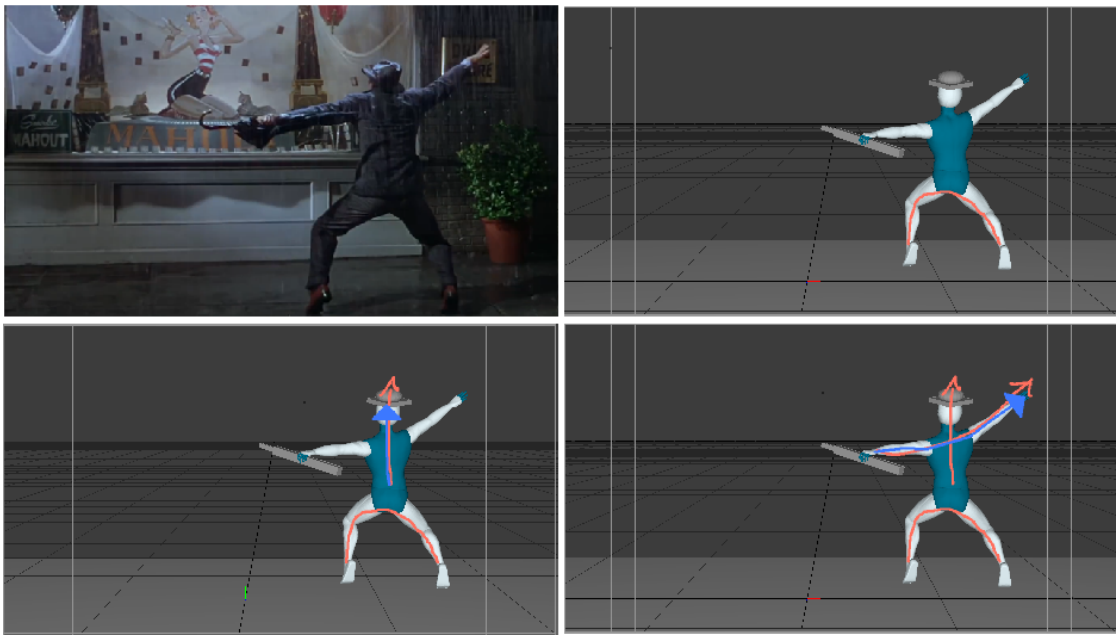
Un temps de nettoyage sur les keyframes et leurs interpolation est à prévoir.



7.2.6 *Workflow* : Gene Kelly, *Singing in the rain* (1952)

Le workflow permettant de réaliser l'animation de Gene Kelly est le suivant :

1. Sélectionner et tracer la bodyline des jambes ;
2. Sélectionner et tracer la bodyline du torse ;
3. Sélectionner et tracer la bodyline des bras ;
4. Ajuster au besoin pour préciser le posing ;
5. Nettoyer les interpolations entre les keyframes.



7.2.7 *Workflow* : Cyd Charisse, Fred Astaire, *The Band Wagon* (1953)

Le workflow permettant de réaliser l'animation de Cyd Charrisse et de Fred Astaire est le suivant :

1. Sélectionner et tracer la bodyline du pied à la tête de Cyd Charisse ;
2. Sélectionner et tracer la bodyline du torse de Fred Astaire ;
3. Sélectionner et tracer la bodyline des quatre bras des deux danseurs (multi-bodyline) ;
4. Ajuster au besoin pour préciser le posing ;
5. Nettoyer les interpolations entre les keyframes.

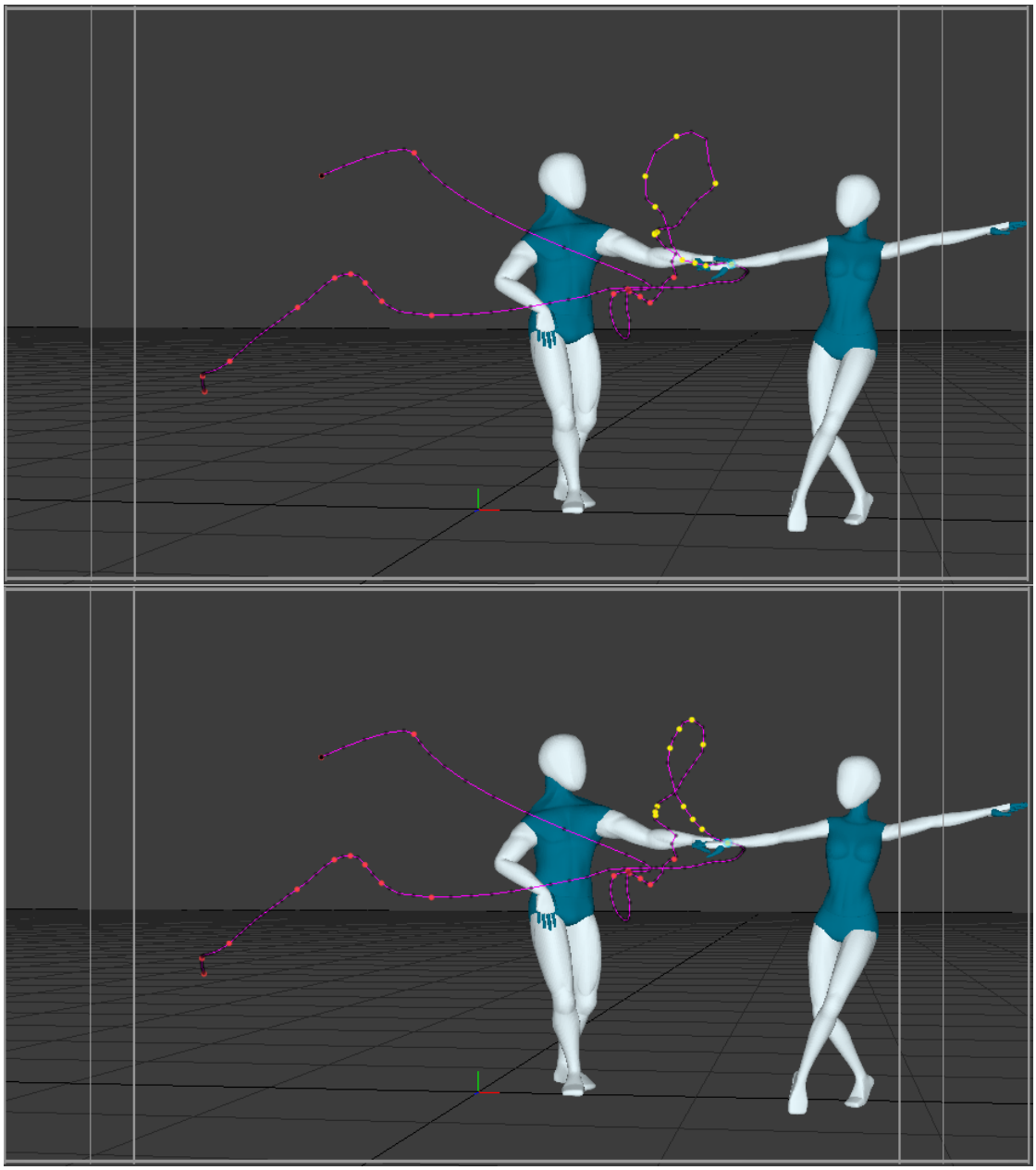


FIGURE 25 – Édition de la trajectoire de la main de Cyd Charisse

7.2.8 Conclusion

Les outils présentés dans ce rapport démontrent bon nombre de qualités : le principe d'animer en dessinant, qui plus est en 3D, est plus intuitif et plus rapide qu'avec une méthode traditionnelle. Cependant, les règles liées à l'animation 3D (gestion de contraintes, interpolation ...) posent des limitations aux animateurs. Ces limitations peuvent être levées avec un temps supplémentaire de développement (et il sera nécessaire de tester ces améliorations).

Durant les exercices effectués par les animateurs, il a été nécessaire de consacrer un temps pour *polisher* des poses, corriger certaines interpolations entre deux keyframes. De ce fait le temps pour produire une animation avec l'outil reste supérieur que d'ordinaire. Toutefois sur du simple posing ou sur des chaines de contrôleurs FK (cordes, queues, doigts, cheveux ...), l'animation par sketch va augmenter la qualité/précision/vitesse de l'animateur.

L'outil de trajectoire s'est montré très utile et puissant en raison de sa compatibilité avec les « layers d'animation », et son option de « lock axis », la création et édition de trajectoire va faire en un tracé, ce que ferait en temps normal un animateur en plusieurs manipulations de contrôleurs.

8 Conclusion

Dans ce rapport, nous avons présenté l'ensemble des fonctions d'animation par croquis développées au cours du projet COLLODI2, ainsi que les résultats de leur évaluation en vraie grandeur, sur des tâches d'animation complexe (chorégraphies).

Ces résultats d'évaluation démontrent que les outils proposés permettent d'obtenir des animations de bonne qualité, et sont correctement intégrés dans le workflow d'animation de RUMBA, sans effets de bord, ni dégradation visible de performance.

En revanche, ces résultats permettent de mettre en évidence un certain nombre de limitations qui ralentissent considérablement le travail d'animation par croquis :

- En posing, il est nécessaire de pouvoir facilement poser des contraintes sur les parties du corps qui ont déja été posées et qui ne doivent plus être modifiées, même si elles appartiennent à la body line en cours d'édition. En particulier, lorsqu'on édite plusieurs body lines à la suite, il serait souhaitable de maintenir par défaut les positions des bones communs au deux body lines (et il devrait être possible également de les over-rider si nécessaire).
- En ghosting, il est nécessaire de proposer un autre mode d'affichage des animations, qui maintiendrait les keyframes en place et ferait défiler les personnage en boucle entre les keyframes. (Aujourd'hui, c'est l'inverse qui se passe, les keyframes défilent autour du personnage animé à la frame courante). Ce mode d'affichage permettrait d'alterner plus facilement entre les modes d'édition de trajectoires et d'inspection des animation produites. Le mode d'affichage actuel est imposé par RUMBA mais ne semble pas adapté aux outils d'animation par croquis.
- En animation multi-personnage, il est nécessaire de prendre en compte les contacts entre personnages (par exemple, deux danseurs qui se tiennent par la main).
- Dans tous les cas (posing et animation), il est nécessaire de mieux prendre en compte les contraintes anatomiques des rigs des personnages. En particulier, il serait souhaitable de limiter les degrés de liberté des genoux et des coudes et de leur imposer des limites angulaires.
- La gestion des "pole vectors" et des contraintes IK peut encore être améliorée. Lorsque l'on rétablit les contraintes IK après une opération de posing ou d'animation, les poses et les animations obtenues ne sont pas toujours compatibles avec celles du solveur IK de

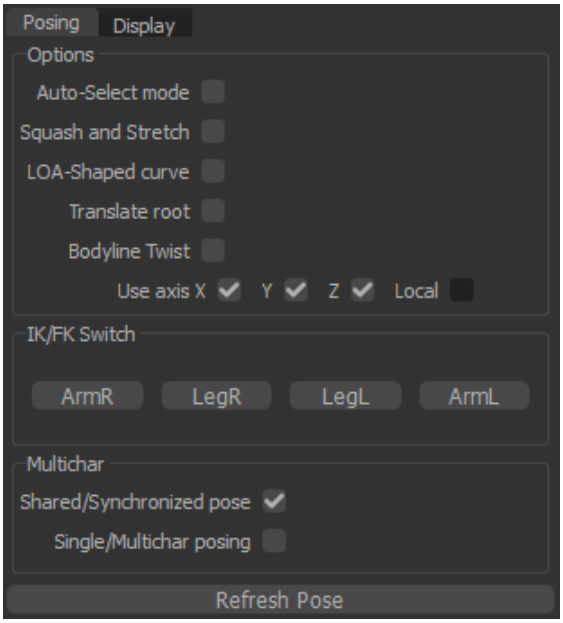
RUMBA. Il semble donc nécessaire de procéder à une évaluation intensive et rigoureuse de la compatibilité entre l'IK de RUMBA et l'animation par croquis. La solution actuellement adoptée est basée sur l'utilisation des "pole vectors" du rig, mais il n'est pas encore parfaitement clair que cette approche soit suffisante. Si cela n'était pas le cas, il pourrait être nécessaire de réviser les algorithmes d'animation par croquis et/ou de l'IK de RUMBA afin d'assurer leur compatibilité théorique et pratique.

Nous estimons que ces limitations peuvent être levées en moins de six mois, et permettraient de grandement améliorer les résultats d'évaluation en vue d'une intégration effective des outils d'animation par croquis dans une version de production de RUMBA.

A Mode d'emploi des outils

A.1 Outil de pose par lignes d'action

L'outil de pose par lignes d'action s'active en pressant la touche **L** dans Rumba. La sélection de bodyline se fait en maintenant la touche **Shift** pendant que l'on dessine la bodyline que l'on souhaite. Pour poser une bodyline, il suffit de dessiner la ligne d'action désirée.



Auto-Select mode
Permet de poser une bodyline sans la sélectionner au préalable : le même sketch sera utilisé pour sélectionner la bodyline et la poser.

Squash and Stretch
Active/Désactive le squash & stretch.

LOA-Shaped curve
Remplace la courbe dessinée par la courbe « ligne d'action » qui l'approxime au mieux.

Translate root
Déplace le modèle pour « coller » à la ligne d'action.

Bodyline twist
Active/Désactive le twiwt par canettes de la bodyline.

Use axis X/Y/Z/Local
Bloque la rotation des contrôleurs selon certains axes. Si *Local* est cochée, les axes sont locaux (repère du contrôleur), sinon les axes sont ceux de la scène 3D. Dans ce cas, la normale du plan d'action est remplacée par un axe calculé en fonction des axes bloqués (par exemple, si on bloque les axes X et Y, les rotations se feront uniquement selon l'axe Z).

Note : l'option ne fonctionne pour l'instant que si deux axes sont bloqués. L'option *Local* n'est pas fonctionnelle.

IK/FK Switch
Pour un personnage, fixe/libère ses extrémités (jambes/bras).

Multichar Shared/Synchronized pose
En mode multi-personnages, pose les personnages de manière « partagée » (la ligne d'action est découpée et chaque morceau est utilisé pour poser les personnages) ou de manière synchronisée (la même ligne d'action est utilisée pour poser tous les personnages).

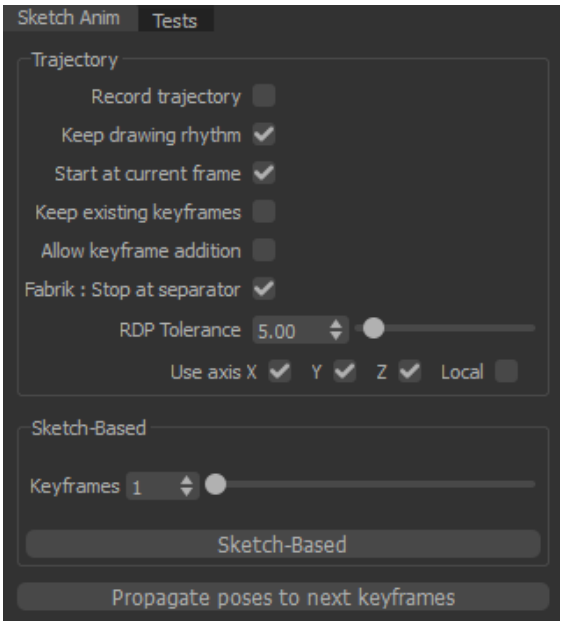
Multichar Single/Multichar posing
Active/Désactive le mode multi-personnnages.

Refresh pose
Ré-applique la dernière ligne d'action dessinée.

A.2 Outil d'animation par trajectoires

L'outil d'animation par trajectoires s'active en pressant la touche **P** dans Rumba. La visualisation des trajectoires s'effectue en activant le motion trail  . Le mode de visualisation locale s'active en appuyant sur le bouton  situé à côté du motion trail. Le *screen space offset* peut se régler via un slider également situé à côté du motion trail.

Une fois un contrôleur sélectionné, on peut définir sa nouvelle trajectoire en la dessinant. Si on veut éditer une trajectoire locale, il faut alors sélectionner le contrôleur **puis** son parent et activer le mode local (même bouton que pour la visualisation).



Record trajectory
Active/Désactive le mode d'enregistrement de la trajectoire.

Keep drawing rhythm
Conserve le rythme de tracé. N'a pas d'effet si *Record trajectory* est cochée.

Start at current frame
Par défaut, les trajectoires modifient l'intervalle d'édition défini par la *timeline*. Si cette option est cochée, l'intervalle d'édition commencera à la frame courante. N'a pas d'effet si *Record trajectory* est cochée.

Keep existing keyframes
Quand on édite une trajectoire, demande à conserver uniquement les clés déjà posées.

Allow keyframe addition
Quand on édite une trajectoire, autorise l'ajout de clés. N'a pas d'effet si *Keep existing keyframes* est décochée.

Fabrik : stop at separator
Indique si une chaîne IK doit d'arrêter à un séparateur (contrôleur ayant plusieurs enfants) ou à la racine. Par exemple, si l'option est cochée, la chaîne IK du bras ira jusqu'à l'épaule (et non pas la racine).

RDP tolerance
Indique l'erreur que l'on tolère entre la courbe que l'on dessine et la courbe résultante. Plus cette valeur est faible, plus la courbe résultante ressemblera à la courbe dessinée (avec cependant plus de clés). N'a pas d'effet si *Keep existing keyframes* est cochée.

Use axis X/Y/Z/Local
Bloque le mouvement du contrôleur selon certains axes. Si *Local* est cochée, les axes sont locaux (repère du contrôleur), sinon les axes sont ceux de la scène 3D. Par exemple, si on bloque l'axe Y, les mouvements du contrôleur se feront uniquement dans le plan X/Z.

Sketch-Based
Adapte l'intervalle d'édition de la *timeline* pour afficher un certain nombre de keyframes (réglable via l'option *Keyframes*) avant et après la frame courante.

Propagate poses to next keyframes
Copie la pose effectuée à la frame courante aux keyframes situées après cette frame.

B Exemple de hiérarchie de contrôleurs

Le JSON suivant illustre un exemple de hiérarchie de contrôleurs pour un modèle de personnage. Nous décrivons sa structure par la suite.

```
{
    "root": {
        "name": "COG",
        "children": [
            {
                "name": "HipL",
                "limb": "LegL",
```

```
        "children": [
            {
                "name": "KneeL",
                "limb": "LegL",
                "children": [
                    {
                        "name": "FootL",
                        "limb": "LegL",
                        "children": []
                    }
                ]
            }
        ]
    },
    {
        "name": "HipR"
        ...
    },
    {
        "name": "Pelvis",
        "limb": "Chest",
        "children": [
            {
                "name": "Spine",
                "limb": "Chest",
                "children": [
                    {
                        "name": "chest_arms_sep",
                        "separator": true,
                        "children": {
                            {
                                "name": "ShoulderL",
                                "limb": "ArmL",
                                "children": [
                                    {
                                        "name": "ElbowL",
                                        "limb": "ArmL",
                                        "children": [
                                            {
                                                "name": "HandL",
                                                "limb": "ArmL",
                                                "children": []
                                            }
                                        ]
                                    }
                                ]
                            },
                            {
                                "name": "HipR"
                                ...
                            }
                        }
                    }
```

```
                        ]
                    }
                ]
            }
        ]
    },
    "fabrik_effectors": {
        "ArmL": {
            "fk_controller": "HandL",
            "ik_controller": "HandL_IK",
            "pole_vector": "HandL_PV"
        },
        "ArmR": {
            "fk_controller": "HandR",
            "ik_controller": "HandR_IK",
            "pole_vector": "HandR_PV"
        },
        "LegL": {
            "fk_controller": "FootL",
            "ik_controller": "FootL_IK",
            "pole_vector": "FootL_PV"
        },
        "LegR": {
            "fk_controller": "FootR",
            "ik_controller": "FootR_IK",
            "pole_vector": "FootR_PV"
        }
    }
}
```

La hiérarchie de contrôleurs est une structure constituée de **noeuds**, eux-mêmes définis par les éléments suivants :

- **name** : le nom du contrôleur ;
- **limb** (optionnel) : indique le membre auquel le contrôleur appartient. Pendant l'étape de pose, on se sert autant que possible de ce champ pour savoir quels membres du personnage font partie de la bodyline ;
- **separator** (optionnel) : indique si le noeud est un noeud de séparation. Lors de l'étape de décomposition de la bodyline en sous-bodylines, on se sert entre autres de ces noeuds pour savoir où découper la bodyline initiale ;
- **flags** (optionnel) : ensemble d'options utilisées pendant l'étape de pose (décrites par la suite) ;
- **children** : les enfants du noeud dans la hiérarchie.

Les principales options de l'élément **flags** sont les suivantes :

- **twist** : indique si le contrôleur peut être twisté ou non (avec les canettes) ;
- **stretch** : indique si le contrôleur peut subir du squash & stretch ;
- **fabrik_fix_to_child** : indique si le contrôleur peut faire partie d'une chaîne IK. Par exemple, si on applique FABRIK pour transmettre la trajectoire du bout d'un doigt à l'ensemble du bras, FABRIK atteindra souvent la cible en pliant le doigt de manière aberrante. Avec cette option, la chaîne IK associée au doigt passe directement du contrôleur du doigt au coude, pour éviter ces incohérences visuelles ;

— **maximal_bodylines** : indique si le contrôleur peut faire partie d'une bodyline maximale (joignant deux extrémités du modèle) ou pas. Par exemple, pour un personnage, on utilise ce flag pour les contrôleurs des doigts : les bodylines maximales (qu'on utilise en général pour donner la forme globale) s'arrêtent alors aux poignets. Pour avoir un doigt dans une bodyline, il faudra alors sélectionner soit un seul bras avec le doigt soit le doigt uniquement.

Le noeud **fabrik_effectors** contient toutes les informations nécessaires au fonctionnement optimal de FABRIK. Pour chaque membre du personnage, on spécifie :
— Le contrôleur FK qui fait office d'extrémité de la chaîne IK ;
— Le contrôleur IK associé au membre (si le rig en fournit un) ;
— Le pole vector associé au membre (si le rig en fournit un).

Références

[1] Andreas Aristidou and Joan Lasenby. FABRIK : A fast, iterative solver for the inverse kinematics problem. *Graph. Models*, 73(5) :243–260, September 2011.

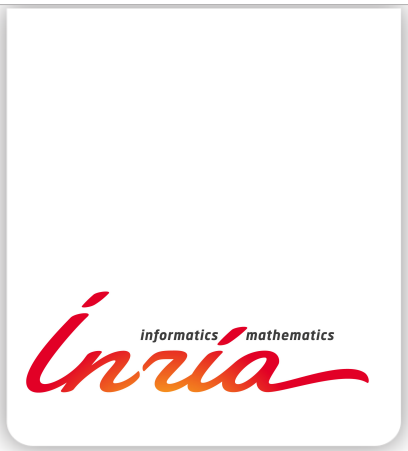
[2] Byungkuk Choi, Roger Blanco i Ribera, J. P. Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Jun yong Noh. Sketchimo : sketch-based motion editing for articulated characters. *ACM Trans. Graph.*, 35(4) :146 :1–146 :12, 2016.

[3] Martin Guay. *Sketching free-form poses and motions for expressive 3D character animation.* Theses, Université Grenoble Alpes, July 2015.

[4] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. The Line of Action : an Intuitive Interface for Expressive Character Posing. *ACM Transactions on Graphics*, 32(6) :Article No. 205, November 2013.

[5] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. Space-time sketching of character animation. *ACM Transactions on Graphics*, 34(4) :Article No. 118, August 2015.

[6] Wikipedia. Ramer–Douglas–Peucker algorithm — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm`, 2019.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803