Département
du Premier
Cycle

Projet Informatique 2ème Année

Simulation mécanique d'un solide et moteur graphique 2D en Pascal

Tristan Pourcelot
Victor Cleren
Rémi Verschelde
Théo Garcia-Guitton

Responsable : C. Wolf

Groupe 52

Année scolaire 2010-2011

Institut National des Sciences Appliquées de Lyon

1 Introduction

Nous sommes partis d'une idée simple, celle de modéliser la chute d'un objet sur un décor, sachant que l'utilisateur pouvait influer à la fois sur le décor et sur l'objet. Nous avons également voulu intégrer de nombreux paramètres tels que la variation du matériau constituant le décor et l'objet, ainsi que la forme de cet objet. L'utilisateur pourrait ainsi (au prix de quelques approximations) simuler la chute d'un satellite sur la Lune.

2 Spécifications

L'objet principal de notre programme est de modéliser la chute d'un solide dans un fluide, tout en prenant en compte son interaction avec un environnement solide (chocs).

2.1 IHM - *Interface Homme - Machine*

Il est proposé à l'utilisateur de dessiner à la fois le décor et l'objet. Les deux dessins sont séparés en deux fenêtres, munie chacunes de leur interface propre, puisque les outils nécessaires au dessin du décor ne sont pas les mêmes que pour le dessin de l'objet. L'utilisateur peut également ouvrir une fenêtre dédiée aux paramètres physiques, afin d'influer sur la valeur de la gravité notamment.

2.2 Problèmes physiques et méthodes de résolution

2.2.1 Forces

Le programme doit prendre en compte les différentes forces appliquées à un instant t sur le solide. Le mouvement de ce solide est donc déterminé par les équations de la mécanique newtonienne. On peut donc définir les différentes forces appliquées sur le solide :

- La gravité : ..... $\overrightarrow{P} = mg\,\overrightarrow{y_0}$
- La poussée d'Archimède : ..... $\overrightarrow{F_{Ar}} = -\rho_{liquide} \cdot V_{deplace}\,\overrightarrow{y_0}$
- Les frottements lors du contact : ..... $\overrightarrow{F_{fr}} = -f\,\overrightarrow{V}_{\text{Objet/Décor}}$

La majorité des forces sont résolues à l'aide d'algorithmes basiques de calcul, prenant en compte des paramètres telles que le volume de l'objet, sa vitesse ou son poids.

2.2.2 *PFD* - Principe Fondamental de la Dynamique

Dans le cas d'un problème plan, notre problème s'exprime sous la forme :

$$\sum F_x = m\frac{dv_x}{dt} \text{ et } \sum F_y = m\frac{dv_y}{dt} \text{ et } \sum M_z = J\frac{d\omega}{dt} \tag{1}$$

En supposant le Δt du timer suffisament faible, on peut assimiler $\frac{dv_x}{dt}$ à $\frac{\Delta v_x}{\Delta t}$, $\frac{dv_y}{dt} \approx \frac{\Delta v_y}{\Delta t}$ et $\frac{d\omega}{dt} \approx \frac{\Delta\omega}{\Delta t}$.
Avec les mêmes approximations, on obtient : $\frac{dx}{dt} = v_x \approx \frac{\Delta x}{\Delta t}$, $\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t}$ et $\frac{d\theta}{dt} \approx \frac{\Delta\theta}{\Delta t}$.
La vitesse initiale $\overrightarrow{v_0}$ est définie par l'utilisateur. Ainsi, connaissant la vitesse $\overrightarrow{v(t)}$ à un instant t, on peut calculer la vitesse $\overrightarrow{v(t+\Delta t)}$ à l'instant $t + \Delta t$.

$$v_x(t+\Delta t) = v_x(t) + m\Delta t \sum F_x \text{ et } v_y(t+\Delta t) = v_y(t) + m\Delta t \sum F_y \text{ et } \omega(t+\Delta t) = \omega(t) + J\Delta t \sum m_z \tag{2}$$

En prenant en compte les approximations précédentes, on obtient le système suivant :

$$x(t+\Delta t) = x(t) + \Delta t \cdot v_x(t) \text{ et } y(t+\Delta t) = x(t) + \Delta t \cdot v_x(t) \text{ et } \theta(t+\Delta t) = \theta(t) + \Delta t \cdot \omega(t) \tag{3}$$

2.2.3 Collisions

Notons $\overrightarrow{v_c}$ la vitesse du point de contact et $\overrightarrow{v_g}$ la vitesse, connue, du centre de gravité du solide. Pour le calcul de la nouvelle vitesse du solide, nous supposons que le contact entre le solide et le décor lors du choc est ponctuel. On se place dans la base de contact définie par le schéma suivant :

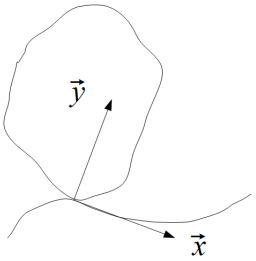

FIGURE 1 – Base locale du contact

Le contact étant ponctuel, on suppose que les forces appliquées au point de contact C sont similaires à celles appliquées dans le modèle de choc d'une particule, on a donc :

$$v_{C,x,initial} = v_{C,x,final} \text{ et } v_{C,y,initial} = -v_{C,y,final} \tag{4}$$

Or, en notant x_G et y_G les coordonnées du centre de gravité dans la base de contact :

$$v_{C,x} = v_{G,x} + y_G \cdot \omega \text{ et } v_{C,y} = v_{g,y} - x_G \cdot \omega \tag{5}$$

On suppose dans un premier temps que l'énergie cinétique est conservée :

$$J\omega_{initial}^2 + m(v_{g,x,initial}^2 + v_{g,y,initial}^2) = J\omega_{final}^2 + m(v_{g,x,final}^2 + v_{g,y,final}^2) \tag{6}$$

On obtient ainsi un système de 3 équations qui nous donne les équations de mouvement après collision. (v_{x0} , v_{x1}) désigne ($v_{G,x,initial}$, $v_{G,y,initial}$), de même pour y et ω .
On obtient ainsi les solutions suivantes :

$$\omega_1 = \frac{-m(x_G^2 + v_{x0}y_G + y_G^2\omega_0 + v_{y0}) + \sqrt{\Delta}}{my_G^2 + mx_G^2 + J} \tag{7}$$

$$\begin{aligned} \Delta \quad = \quad & \omega^2 J^2 - 2Jmv_{x0}y_G\omega_0 + 2Jmv_{y0}x_G\omega_0 + 2m^2v_{x0}y_Gv_{y0}x_G - 4m^2v_{x0}y_Gx_G^2\omega_0 \\ & + 4m^2y_g^2\omega_0v_{y0}x_G - 4m^2y_g^2\omega_0^2x_g^2 + m^2v_{x0}^2y_G^2 + m^2v_{y0}^2x_G^2 \end{aligned}$$

On obtient ainsi :

$$v_{x1} = v_{x0} + y_G(\omega_0 - \omega_1) \text{ et } v_{y1} = v_{y0} + x_G(\omega_0 - \omega_1) \tag{8}$$

3 Réalisation

3.1 Interface

La fenêtre principale affiche un menu utilisateur, une fenêtre de dessin, et un espace d'affichage des principaux paramètres physiques. Outre ces fonctionnalités, c'est également à elle d'effectuer l'animation de l'image, en calculant régulièrement (présence d'un Timer) la position du solide ainsi que sa vitesse et son interaction avec le décor. Les autres fenêtres ont pour mission de gérer le dessin de l'objet et son remplissage, le dessin du décor et la gestion des matériaux. Nous avons également prévu une fenêtre permettant de changer les paramètres physiques.

3.2 Gestion des chocs

La partie la plus compliquée dans la gestion des chocs reste la détection des chocs et le calcul de la tangente locale au décor. Afin d'obtenir ces informations, notre algorithme analyse à chaque itération d'un timer prédéfini les pixels autour du solide, et si jamais ceux ci appartiennent à un des matériaux (tri par couleur), on calcule la tangente locale à l'aide de tout les points de contact. Ceci nous permet de déterminer la direction de rebond du solide, ainsi que d'appliquer les forces de frottement propres au matériau. Par ailleurs, nous appliquons la formule des chocs élastiques afin de calculer la nouvelle vitesse du solide.

3.3 Remplissage de l'objet

Un des problèmes majeurs qui nous fut posé durant ce projet fut le remplissage du solide afin de pouvoir déterminer sa masse et son centre d'inertie. Tout d'abord, il fallut s'assurer que le dessin de l'utilisateur était bien connexe. Pour remédier à ce problème, nous traçons une droite entre le point de départ (X_e, Y_e) et le point de sortie (X_s, Y_s). Nous avons choisi la méthode du ScanLine-FloodFill pour remplir le solide dessiné. Cette méthode utilise une graine (la *seed*) placée sur un point quelquonque de notre solide, et analyse tout les pixels environnants pour déterminer quels sont ceux à l'intérieur du solide, et les autres. Après plusieurs essais infructueux avec un algorithme classique, nous avons adapté un algorithme en C trouvé sur Internet, qui analyse ligne par ligne l'objet, permettant ainsi un gain considérable en efficacité.

4 Bugs - Améliorations

Il subsiste de nombreux bugs et améliorations possibles de notre projet. En particulier, de nombreuses améliorations que nous pensions apporter à notre projet n'ont pas été implémentées par faute de temps, certains problèmes ayant été plus chronophages que prévu.

4.1 Améliorations

- Différents fluides (air, eau, vide ...)
- Variation aléatoire et incidence du vent
- Amélioration du modèle physique (élasticité ...)
- Modification du décor et de l'objet en temps réel par l'utilisateur
- Tracé de la trajectoire de l'objet
- Force de frottement prenant en compte la forme de l'objet

4.2 Bugs

- Lenteur de l'exécution sous Linux (pas de problèmes sous Windows)
- Le décor ne se remet pas à jour si on le redessine
- Calcul des tangentes peu précis (sinon il serait beaucoup trop lent)
- Mauvais fonctionnement en 64 bits (implémentation Lazarus ?)

5 Conclusion

All your base are belong to us !

6 Annexes

6.1 Avis personnels et implications dans le projet

Tristan :

- Implication et réalisations dans le projet :
  - Conception du graphe UML et définition du projet
  - Implémentation UML → Pascal
  - Rédaction du dossier en LaTeX
  - Gestion du café
- Ressenti global :
  - Bon ressenti, mais les problèmes de communication ont nui au travail de groupe.
  - Le fait de coder d'abord le squelette du programme nous a pris du temps qui n'a pas pu être mis à profit pour l'amélioration du programme.

Rémi :

- Implication et réalisations dans le projet :
  - Diagramme de classes et définition du projet
  - Dessin du décor et de l'objet (procédure de remplissage)
  - Détection des collisions et détermination du point de contact
  - Optimisation des procédures
  - Gestion des matériaux
- Ressenti et problèmes rencontrés :
  - Projet ambitieux mais par conséquent contraignant.
  - Les problèmes de non-communication ont nui à la répartition du travail.

Victor :

- Implication et réalisations dans le projet :
  - Diagramme de classes et définition du projet
  - Dessin de l'objet
  - Calcul des tangentes
  - Modélisation, calculs physiques et implémentation
- Ressenti et problèmes rencontrés :
  - Projet très formateur au niveau de la programmation, mais moins au niveau du travail de groupe.
  - Mauvaise gestion du temps qui a nui à l'ambition du projet.

Théo :

- Implication et réalisations dans le projet :
  - Diagramme de classes et définition du projet
  - Recherches bibliographiques
  - Définitions de classes et implémentation
  - Gestion des photocopies
- Ressenti et problèmes rencontrés :
  - La réflexion et le travail de groupe au début du projet étaient très intéressants.
  - Mais les contraintes sur la fin du projet ont contrebalancé ce ressenti.

6.2 Problèmes rencontrés

- Problèmes de communication au sein du groupe (qui fait quoi, quand et pourquoi...)
- Implémentation FreePascal/Lazarus sous Linux
- Conflits SVN avec les fichiers compilés de Lazarus (.ppu)
- Gestion du temps et complexité du projet
- Non compatibilité du programme suivant l'architecture matérielle et l'OS

6.3 Diagrammes de classes

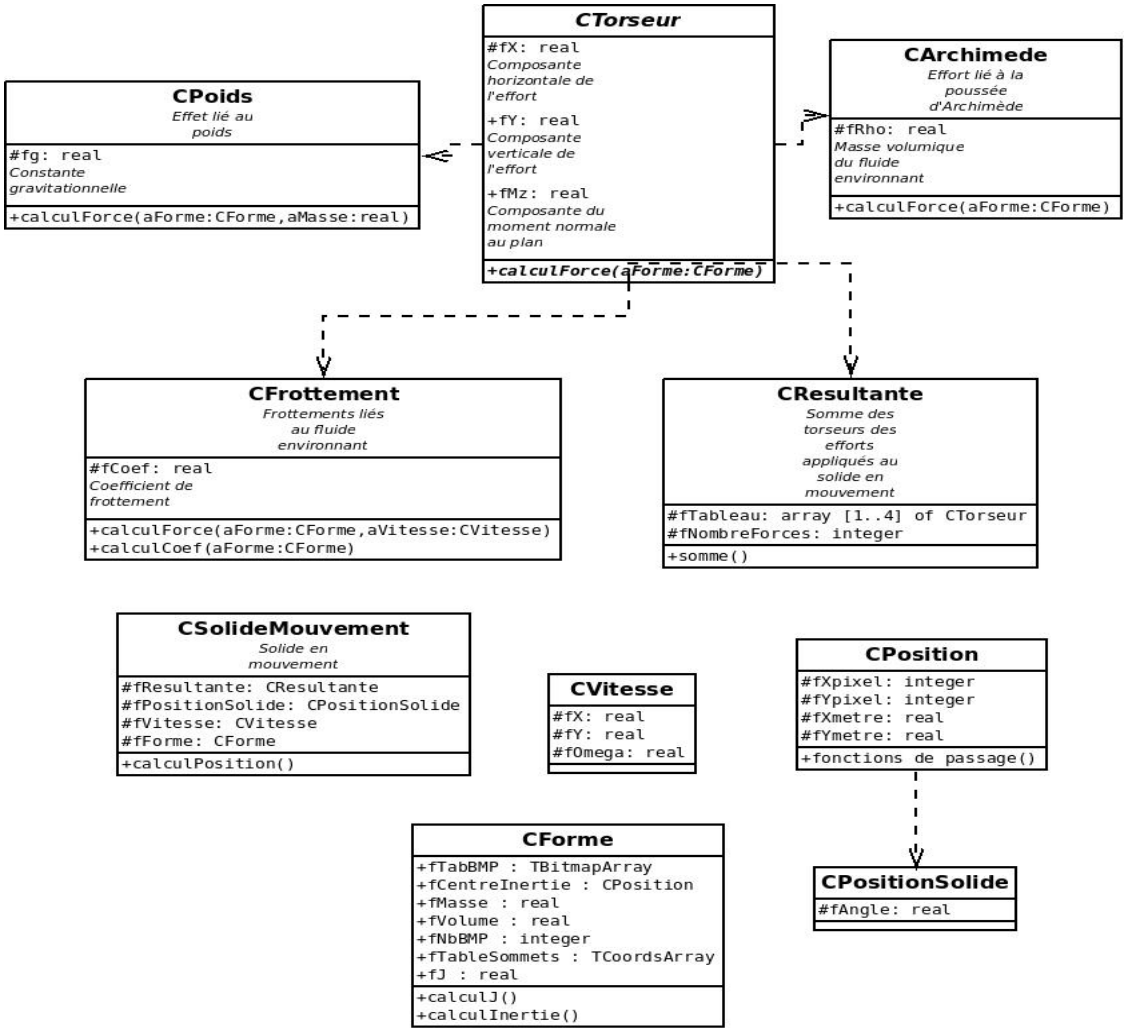

FIGURE 2 – Diagramme de classe utilisé pour notre projet

6.4 Impressions écrans

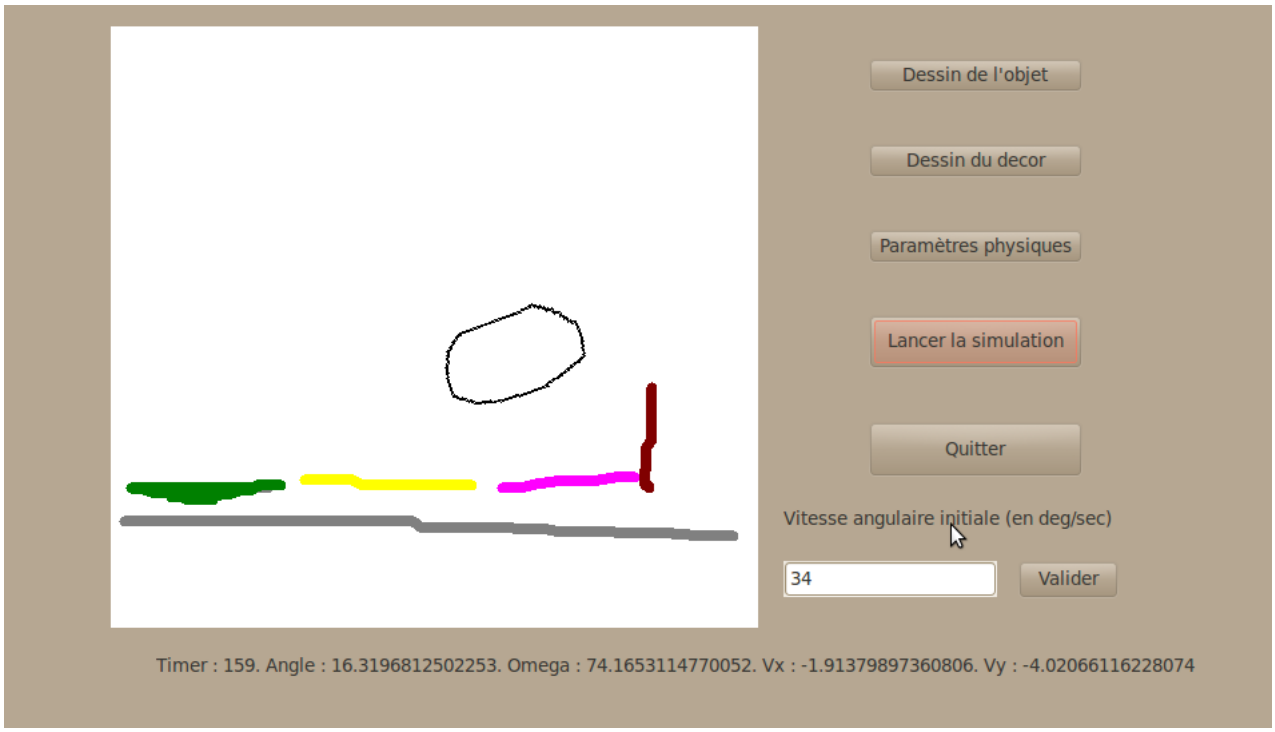

FIGURE 3 – Fenêtre principale

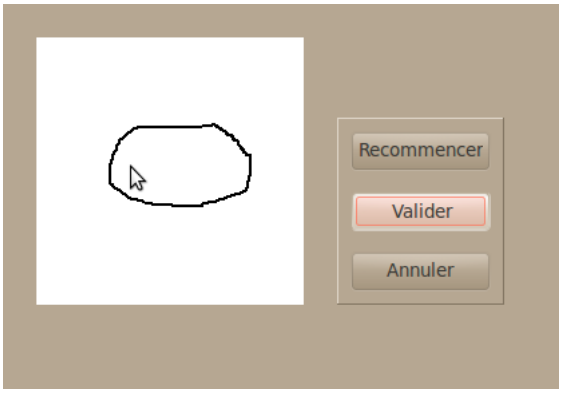

FIGURE 4 – Fenêtre gérant le dessin de l'objet

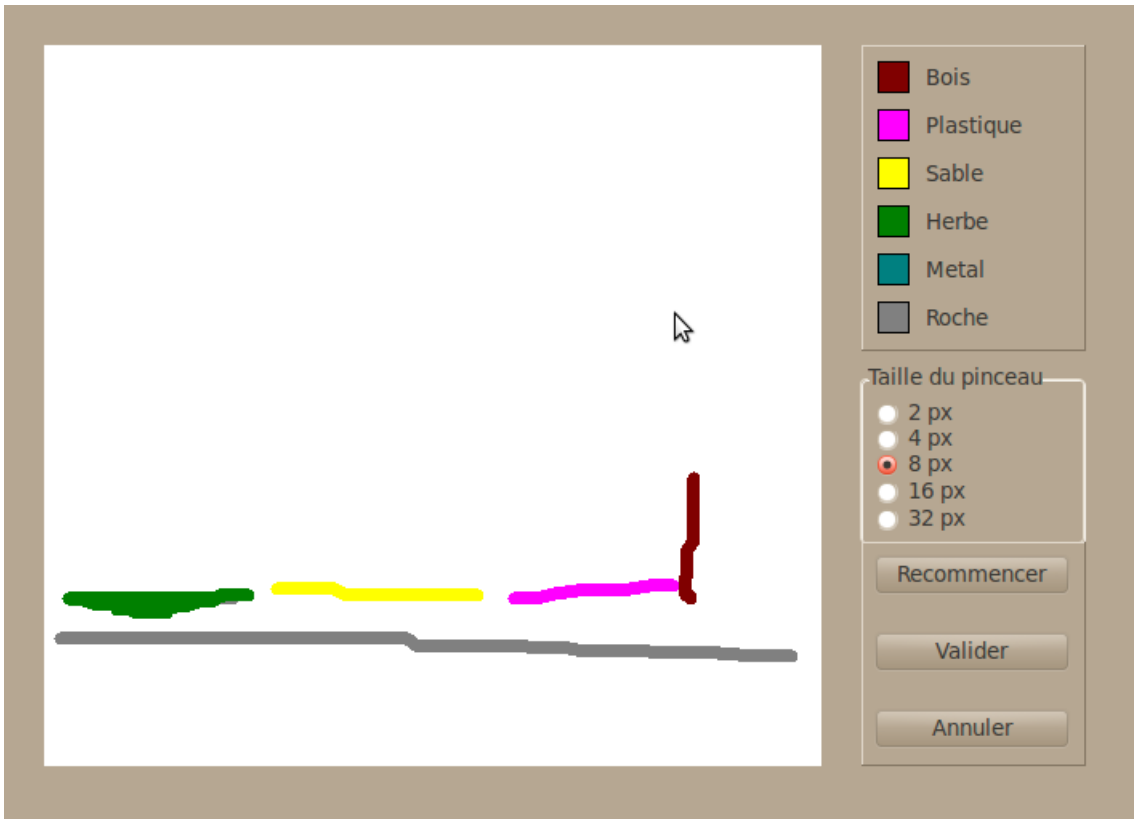

FIGURE 5 – Fenêtre gérant le dessin du décor