REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :…….. /M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Optimisation De La simulation Du Tissu (Exploitation des GPU)

Par :
REZIG NOUZHA

Soutenu le : 26/06/2022 devant le jury composé de :

Zerari Abd El Moumène	MCB	Président
Ben Ameur Sabrina	MCB	Rapporteur
ChighoubRabiaa	MCB	Examinateur

Année universitaire 2021-2022

Sommaire

Table des figures

Chapitre 01 : La simulation de tissu

Chapitre 02 :Le parallélisme

Chapitre 03 : Conception et implémentation

Remerciement

Je tiens à remercier d'abord notre dieu qui nous a donné la force et le courage pour continuer et nous avoir aidés et éclairer le chemin pour réaliser ce travail.

Mes remerciement ma gratitude et reconnaissance à mon encadrante Mm :

BENAMEUR Sabrina pour son soutien et son aide précieuse pour la réalisation de ce travail.

Mes remerciement vont également aux membres de jury, Mr **ZERARI Abd El Moumén** et Mm **CHIGHOUB Rabiaa** pour m'avoir honoré par leur participation à l'évaluation de ce travail.

Un grand merci à mes parents, toute ma famille frères et sœurs, mon époux et mes enfants qui mon encouragés et aidés chaqu'un à sa façon tout au long de mes études.

En fin, j'adresse mes remerciements à tous les enseignants qui ont contribués à ma formation durant toutes mes études.

REZIG Nouzha

Résumé

La simulation et la modélisation de tissus est un domaine très riche dans la recherche en synthèse d'image. Les méthodes proposées sont classées en trois approches principales qui sont: les approches géométriques, physiques et hybrides, mais leur but est de considérer deux critères essentiels à savoir le réalisme (fidélité) de la simulation du mouvement de tissu et la vitesse de simulation. Le développement du potentiel des ordinateurs (processeurs, architectures, processeurs graphiques) a permis d'envisager des applications temps réel de tissus.

Ce travail consiste à apporter une amélioration à une simulation de tissu. Pour cela, nous avons essayé d'exploiter les GPU pour accélérer le rendu d'une animation de tissu. Nous avons utilisé un modèle géométrique pour la modélisation de tissu, et en appliquant la texture à notre modèle nous avons jugé qu'il est nécessaire d'utiliser les shaders pour optimiser le rendu. Les résultats obtenus sont satisfaisant.

Mots Clés : Simulation, Tissu, Modèle géométrique, GPU, Shader.

Abstract

Cloth simulation and modeling is a very rich field in image synthesis research. The proposed methods are classified into three main approaches which are: geometric, physical and hybrid approaches, but their goal is to consider two essential criteria namely the realism (fidelity) of the simulation of the cloth movement and the simulation speed. The development of the potential of computers (processors, architectures, graphics processors) has made it possible to envisage real-time applications of cloth.

This work consists of making an improvement to a fabric simulation. For this, we tried to exploit GPUs to speed up the rendering of a cloth animation. We used a geometric model for fabric modeling, and applying the texture to our model we found it necessary to use shaders to optimize rendering. The results obtained are satisfactory.

Keywords: Simulation, Cloth, Geometric model, GPU, Shader.

I. Introduction générale

L'intérêt du domaine de simulation de tissu n'est pas récent; depuis le milieu des années 1980, cette simulation est un sujet très à la mode en synthèse d'animation. Malgré tous les travaux réalisés dans ce domaine, l'animation de tissus demeure très complexe. D'une technique de simulation à l'autre, la qualité des résultats obtenus, le temps de calcul et la facilité d'utilisation varient énormément. De surcroît, étant donné que les tissus et les vêtements se retrouvent pratiquement partout dans notre vie quotidienne, l'œil peut difficilement être trompé lorsqu'il est question de réalisme.[LAP07]

La modélisation de tissu et constitue un axe de recherche particulièrement important, ses nombreuses applications dans le monde industriel auprès des industries de la confection et du tissage, et grâce à son application directe

La simulation de tissu constitue un axe de recherche particulièrement important de part ses nombreuses applications dans le monde industriel auprès des industries de la confection et du tissage, et grâce à son application directe dans les jeux vidéo ou encore dans les films d'animation, en permettant notamment l'habillage de personnages virtuels.. [2]

De nombreux travaux ont abordé ce domaine, ils ont utilisé des modèles géométrique, physique ou hybride. Généralement le tissu est représenté dans les simulations par un maillage, la résolution de ce dernier affecte énormément le rendu,puisque la résolution des maillages polygonaux des tissus doit être assez fine pour capturer les détails de tissu (les plis qui se forment lorsque des forces agissent sur ces derniers), ce qui augmente grandement le temps de calcul nécessaire à la simulation.Dans ce cadre il s'agit de considérer deux critères essentiels à savoir le réalisme (fidélité) de la simulation du mouvement detissu et la vitesse de simulation.

L'objectif de ce travail est d'accélérer le rendu d'une simulation de tissu en exploitant les GPU. Pour cela nous avons réalisé un système qui permet de simuler des pièces de tissu avec un modèle géométrique et afin d'obtenir un rendu plus réaliste nous avons essayé appliquer des textures sur la surface de tissu, mais puisque ceci nous a alourdi l'animation nous avons exploité la GPU. L'utilisation des shaders nous a permis d'accélérer la simulation.

Organisation du manuscrit

En plus de l'introduction générale, notre mémoire est organisé en 3 chapitres,

Le premier chapitre : est consacré à la simulation de tissu, dont nous commençons par des définitions concernant le tissu et ses propriétés mécaniques, par la suite nous présentons les différents modèles de simulation de tissu tout en citant les avantages et les inconvénients de chaque modèle. Nous avons aussi expliqué le principe de la gestion de collision. A la fin du chapitre nous avons décrire le modèle géométrique.

Le deuxieme chapitre : représenteles architectures parallèleset il décrit aussi les modèles de programmation parallèle. Et puisque nous avons exploité les shaders pour l'accélération du rendu nous avons consacré la dernière partie du chapitre à la définition des concepts liés à leur utilisation.

Le troisième chapitre :est réservé à la conception et l'implémentation de notre système. Dont nous commençons par la présentation de la conception générale et détaillée de notre application ensuite nous détaillons les étapes d'implémentation. Les résultats obtenus ainsi qu'une évaluation sont illustrés à la fin du chapitre.

Une conclusion ainsi que des perspectives à ce travail sont présentées à la fin de ce mémoire.

Chapitre 01

La simulation de tissus

Chapitre 01 :

Simulation de tissu

1 Introduction :

Au cours des dernières années, beaucoup de travaux ont été réalisés dans le but de rendre les personnages virtuels plus réalistes. En effet, ces personnages sont présents autour de nous : films, simulations virtuelles, jeux, etc. Évidemment, la génération de vêtements réalistes sur ces personnages est primordiale pour obtenir un niveau de réalisme acceptable pour ces derniers. L'intérêt à ce sujet n'est pas récent ; depuis le milieu des années 1980, la simulation de tissus est un sujet très à la mode en synthèse d'animation. Malgré tous les travaux réalisés dans ce domaine, l'animation de tissus demeure très complexe. D'une technique de simulation à l'autre, la qualité des résultats obtenus, le temps de calcul et la facilité d'utilisation varient énormément. De surcroît, étant donné que les tissus et les vêtements se retrouvent pratiquement partout dans notre vie quotidienne, l'œil peut difficilement être trompé lorsqu'il es tquestion de réalisme.

Au début de ce chapitre, nous présentons quelques définitions concernant le tissu et ses propriétés mécaniques, par la suite nous détaillons la simulation de tissudont nous présentons le principe et les avantages et les inconvénients des différentes approches utilisées dans ce domaine. Nous expliquons ensuite la problématique de cette simulation ainsi que la gestion de collisions. La dernière partie du chapitre est réservée à la simulation de tissu par modèle géométrique.

1.1 2.Tissu réel

Pour parvenir à simuler de manière réaliste les tissus synthétiques, il est important d'en connaître un peu sur les tissus réels. On distingue plusieurs types de tissus en fonction de leur procédé de fabrication, dont les tissus tissés (entrelacement de fils orthogonaux) et les tricots (combinaison de mailles). [3]

Un tissu « surface souple » est le résultat physique de l'entrelacement des fils textiles chaîne « sens longitudinal » et trame « sens transversal ». Le mode d'entrelacement des deux séries de fils, en tenant compte de leur nature, leurs propriétés mécaniques, physiques et chimiques, joue un rôle sur l'effet esthétique et influe largement sur la qualité du tissu [4].

Il existe une très grande variété de tissus tissés, comme le montrent les exemples de la figure 1, ces étoffes tissées sont constituées par l'entrecroisement de fils de chaîne, et de fils de trame. L'apparence et le comportement mécanique (réaction du tissu face à l'application de forces ou de contraintes) du tissu sont intrinsèquement reliés à la structure interne de ce dernier : la nature des fibres (coton, laine, soie, etc.), la structuredes fils (diamètre, fil simple ou torsionné, couleur, etc.), le mode d'entrecroisement des fils de trame et de chaîne (« armure »), et le nombre de fils au centimètre en trame et en chaîne (« compte »). Tout dépendant des choix faits lors de la confection, les tissus peuvent être souples ou rigides, mats ou brillants, lisses ou rugueux, texturés ou plats, etc. [3].

Figure 1.1 : Quelques photographies de tissus tissés [3].

1.2 2.1 Propriétés mécaniques du tissu

Dans la littérature, les propriétés mécaniques des surfaces déformables dont le tissu fait partie sont regroupées en quatre familles :

- Élasticité : qui caractérise les forces internes résultant d'une déformation géométrique donnée.
- Viscosité: qui inclut les forces internes résultant d'une vitesse de déformation donnée.
- Plasticité: qui décrit comment les propriétés évoluent en fonction de la déformation.
- Résilience: qui définit les limites auxquelles la structure se brise.

Les propriétés d'élasticité telle que la traction, le cisaillement et la flexion sont les plus importants, puisqu'elles sont responsables des effets mécaniques des tissus

Pour avoir une simulation dynamique précise des tissus et des vêtements, les paramètres doivent être extraits à partir de mesures d'un tissu réel. Cette tâche n'est pas simple à réaliser parceque le tissu est un matériau très complexe. Pour cela, des précautions particulières doivent être prises lorsqu'on prend des mesures et on les applique dans une simulation virtuelle [5].

Étant donné que les tissus sont omniprésents, tant autour de nous que dans les milieux virtuels (vêtements des personnages virtuels, drapeaux, rideaux, etc.), la capacité de les simuler de façon réaliste affecte énormément la qualité des animations produites. [3]

1.3 2.3La Modélisation de Tissu

La communauté de l'ingénierie du textile a commencé les premières recherches en matière de modélisation de tissu dès l'année 1930. Par la suite et au début des années 80, le tissu était modélisé comme une texture plaquée sur des surfaces rigides. Depuis, les recherches ont beaucoup évolué avec l'étude des techniques de modélisation et d'animation de tissu.

En infographie, l'efficacité, la stabilité et le réalisme visuel sont plus importants que la précision. Afin de rapproche ces caractéristiques, de nombreuses méthodes sont proposées, elles sont classées selon trois approches : géométrique, physique et hybride. Le principe de ces approches ainsi que leurs avantages et leurs inconvénients sont illustrés dans les tableaux 1 et 2.

2.4 Approches dédiées à la simulation de tissus

<table>
<tr><th colspan="3">Approches</th><th colspan="3">Principes</th></tr>
<tr><td colspan="3">Approches géométriques</td><td colspan="3">▪ Simulation des habits avec des méthodes géométriques
▪ Ces modèles géométriques fournissent des techniques qui reproduisent la forme des tissus en utilisant des modèles mathématiques (surfaces paramétriques) pour modéliser les plis des habits.
▪ Ils cherchent à reproduire l'aspect visuel d'une représentation fixe des tissus et ce sans chercher à quantifier les grandeurs caractéristiques de leur comportement physique</td></tr>
<tr><td rowspan="3">Approches physiques</td><td colspan="2">Modèles continus</td><td rowspan="3">•Les mouvements des tissus sont dus:
– à des forces externe et,
– aux réactions définies par les propriétés du textile
•l’animation de tissu est relativement bien définie :
– en connaissant les forces externes appliquées aux tissus et,
– en utilisant un bon modèle de leur réaction mécanique</td><td colspan="2">▪ Basés sur la théorie de la mécanique des milieux continus.
▪ Le milieu, représentant le tissu, est supposé avoir des caractéristiques mécaniques continues.
▪ Des relations lient les déformations aux contraintes subies par le milieu.
▪ Les déformations représentent l’ensemble des modifications de la géométrie du tissu, par rapport à un état donné (origine des temps, état de repos,…).
▪ Les contraintes désignent les forces élémentaires s'exerçant en un point du tissu</td></tr>
<tr><td rowspan="2">Modèlesdiscrets</td><td>Modèles de particules</td><td rowspan="2">▪ Considèrent l'objet à décrire comme un ensemble fini de sous-objets plus simples.
▪ Les sous-objets sont munis de lois qui régissent leurs interactions internes d'une part, et leurs interactions avec le reste du monde, d'autre part.
▪ Ces lois sont souvent issues de la physique et d’un assemblage de sous-objets appelé systèmes de particules</td><td>• Les vêtements sont discrétisés en un maillage polygonal
• Les sommets du maillage sont appelés particules ou masses
• La topologie du maillage définit les interactions et les forces exercées entre les particules.
• Les forces inter-particules sont souvent des lois d’attraction-répulsion qui s’apparentent à celles qui existent entre deux atomes</td></tr>
<tr><td>M. masses - ressorts</td><td>• Les interactions entre les particules sont modélisées à l’aide de ressorts
• L’évolution du système est guidée via la Loi fondamentale de la dynamique: mi .ai = Fi</td></tr>
<tr><td colspan="3">Approcheshybrides</td><td colspan="3">• Combinent les avantages des deux approches précédentes.
• L'idée d'utiliser des méthodes basées à la fois sur la physique et la géométrie vient du constat suivant:
– Les détails de l'habit qui sont les plus coûteux à calculer sont par nature géométrique. Ils peuvent être donc modélisés avec une surface déformée géométriquement sans perdre la qualité de la simulation.
– En clair, le comportement global de l'habit est simulé avec un maillage simplifié et les détails sont générés avec une surface paramétrique dont les points de contrôles sont les sommets du maillage simplifié.</td></tr>
</table>

Tableau 1 : Principes des approches dédiées à la simulation de tissus [6]

Approches			Avantages	Inconvénients
Approches géométriques			• Fournissent des résultats plus rapides que celles basées sur les modèles physiques.	• Incapables de reproduire correctement le mouvement dynamique de l'habit. • Nécessitent une intervention considérable de l'utilisateur pour le dessin des vêtements. • Sont limités à une représentation statique des tissus.
Approches physiques	Modèles continus		• Considèrent le tissu comme un milieu homogène. • Représentent globalement sa surface sans décrire sa mécanique interne.	• Difficulté de déterminer les paramètres mécaniques du modèle. De plus. • Ils nécessitent un lourd bagage scientifique pour la compréhension les équations physiques.
	Modèlesdiscrets	Modèles de particules	• Utilisés dans la simulation de grands nombres de phénomènes physiques, en particulier le comportement des textiles. • Modélisent des phénomènes complexes et globaux à partir de lois comportementales simples et locales agissant sur un ensemble de particules interconnectées.	• L'affichage est plus complexe que celui des réseaux masses-ressorts à topologie fixe. • La complexité vient du fait qu'on ne dispose qu'un ensemble de particules, représentant des objets à topologie variable.
		Modèles masses - ressorts	• Très fréquemment utilisé en informatique graphique • Facile à implémenté et efficace dans la simulation des objets déformables. • Sont Appliqués à l'animation : – de corps inanimés (comme le tissu) ; – de corps organiques actifs (comme les muscles);	• Ne peuvent pas maintenir un volume constant ou quasi-constant pendant la déformation. • Le changement de la densité du maillage, en utilisant les mêmes paramètres des ressorts, est très difficile. • Effet de super-élasticité.
Approcheshybrides			• Méthodes rapides.	• Il existe une très grande variété de la forme de plis et il est très difficile de définir une fonction assez générale qui permette de les modéliser.

Tableau 2 : Avantages et inconvénients des approches dédiées à la simulation de tissus.

1.4 Problématique

Les deux points essentiels à prendre en charge dans la simulation de textile sont le réalisme de la simulation du mouvement de tissu et la vitesse de simulation [7].

2.5 Augmentation de la Fidélité de la Simulation

Le but essentiel dans la simulation de textile est la production d'un tissu non distinguable de celui du réel. Afin d'aboutir à une modélisation réaliste d'un tissu particulier, il faut extraire les propriétés physiques du tissu et les appliquer au modèle du tissu simulé. Actuellement, les mesures objectives des caractéristiques mécaniques et physiques du tissu sont obtenues par plusieurs technologies, en particulier, par celle du KES-F (Kawabata's Evaluation System for Fabrics), le but des travaux utilisant les mesures des machines de Kawabata est de modéliser un tissu numérique qui reproduise les courbes de Kawabata du tissu réel.

D'autres chercheurs préfèrent estimer les paramètres du tissu en comparant le mouvement dynamique du tissu simulé avec le mouvement d'un tissu réel enregistré avec une caméra.

2.6 Amélioration de la Vitesse des Algorithmes

Avoir un temps de calcul raisonnable est un autre but dans la simulation de tissu. La vitesse de la simulation peut être améliorée en employant des modèles physiques et des procédures numériques simples. Dans les modèles physiques, il apparaît que le système de particules est le plus simple pour ce type de simulation. D'un autre côté, différentes méthodes sont utilisées pour l'accélération des procédures numériques, dont la plupart sont basées sur la simplification de la méthode implicite. Cependant, toute simplification exige une dégradation de la qualité de la simulation dans une certaine mesure. En exploitant le fait que les différentes parties du tissu ne doivent pas être simulées avec le même niveau de détails, le temps d'exécution peut être réduit tout en sauvegardant la qualité. L'approche multi-résolution utilise une résolution très fine dans les parties subissant une large déformation et/ou proche de la caméra et une résolution moins poussée pour les autres parties.

L'accélération de la méthode de simulation peut être aussi faite grâce aux nouvelles architectures graphiques, lesquelles combinées avec les algorithmes d'accélération permettraient de produire des simulations de tissus en temps réel.

1.5 3. La gestion de collisions

La gestion de collisions qui se produisent entre le tissu et un autre objet ou entre des parties d'un même tissu constitue un problème essentiel dans la simulation de tissus. Ce problème concerne la détection de collisions, c'est-à-dire comment trouver, d'une façon efficace, le contact géométrique entre deux surfaces complexes, d'une part. D'autre part, il y'a lieu de déterminer la réponse à ces collisions, en d'autres termes, comment ce contact géométrique peut-il affecter le comportement du tissu, et comment ce changement s'intègre efficacement dans la simulation [9].

Une collision entre deux objets est détectée s'il existe un instant où les deux objets ont une intersection non vide [10]. Une collision, dans le cas des surfaces flexibles (comme le tissu), est le contact d'une partie du tissu avec une autre partie du même tissu (auto-collision) ou avec un autre objet. Ainsi, lorsqu'une collision est détectée, il faut pouvoir en tirer les informations qui vont permettre de réagir à cette collision: c'est le calcul de la réponse, qui dépend étroitement de l'application.

Le tissu est un objet déformable, son mouvement peut entraîner davantage de collisions. Généralement modélisé par un maillage triangulaire ou rectangulaire, ce qui rend le test de collision plus coûteux en terme de temps de calcul. De plus, sa structure fine ne permet pas de détecter toutes les collisions durant la simulation. Le problème essentiel dans le traitement de collisions en animation de tissu est que la résolution d'une collision peut provoquer de nouvelles collisions.[11].

4Simulation de tissus par modèle Géométriques

4.1 Le modèle Géométrique

La modélisation géométrique est un ensemble d'outils mathématiques, numériques et informatiques qui se combinent pour construire des modèles virtuels (ou modèles informatiques) d'objets réels. Cet objet peut être plus ou moins complexe, plus ou moins schématisé. Elle peut être le fruit de l'imagination, des tendances, ou plutôt une solution plus ou moins précise à un problème physique donné, voire un compromis entre les deux.

La modélisation géométrique consiste également à comprendre comment reconstruire des objets à partir de la numérisation d'objets existants sans modèle (pour l'ajout à des bases de données, la modélisation par réplication, les domaines médicaux), ainsi que des objets déjà modélisés et fabriqués. Pour vérifier l'écart entre le modèle d'objets virtuels et les objets fabriqués.[12]

Concernant la simulation de tissu, les modèles géométriques fournissent des techniques qui reproduisent la forme des tissus en utilisant par exemple des surfaces paramétriques pour modéliser les plis des habits. Ils cherchent à reproduire l'aspect visuel d'une représentation fixe des tissus, et ce, sans chercher à quantifier les grandeurs caractéristiques de leur comportement physique. Ces approches peuvent être considérées comme des outils de modélisation.

4.2 Travaux antérieurs

Parmi les travaux utilisant les méthodes géométriques nous citons: Celui de Jerry Weil qui a fait paraître un article sur le sujet [13]. Il a développé une approche purement 'géométrique' de la représentation des textiles suspendus. La structure de tissu est modélisée comme une grille bidimensionnelle de points géométriques tridimensionnels. Plus tard, B. K. Hinds et J. McCartney ont présenté dans [14] un système interactif de conception de vêtements tout en permettant à l'utilisateur de créer un modèle géométrique d'un vêtement en indiquant ses contours sur un mannequin, ensuite le vêtement équivalent est généré à partir de la surface de mannequin. Les plis sont ajoutés dans le modèle géométrique. Jean-Michel Nourrit et al. [15] ont modélisé des textiles à base de maille, pour modéliser un liage, ils ont utilisé des courbes Splines qui déterminent la trajectoire du fil, et dont les points de contrôle sont ajustés en fonction de règles dépendant des paramètres du liage. Le modèle proposé par les auteurs permettra de simuler la plupart des liages réalisables sur des métiers industriels, et pourra être utilisé comme outil de conception de nouveaux modèles.

Dans ce qui suit nous citons les travaux et les modèles de référence dans la simulation de tissu par modèle géométrique.

4.2.1 Bump mapping

Blinn [16] est le premier à s'intéresser à la création de bosses et de plis pour simuler les irrégularités des surfaces naturelles. Il a remarqué que l'effet des plis sur l'intensité perçue de la réflexion de la lumière sur une surface est d'abord et avant tout causé par leur effet sur la

direction des normales, plutôt que par leur effet sur la position de la surface. Plutôt que modéliser géométriquement les bosses et les plis, il a montré qu'il est suffisant de générer une fonction de texture (bump map) qui modifie la direction des normales de la surface, avant les calculs d'illumination **(figure 1.2).**

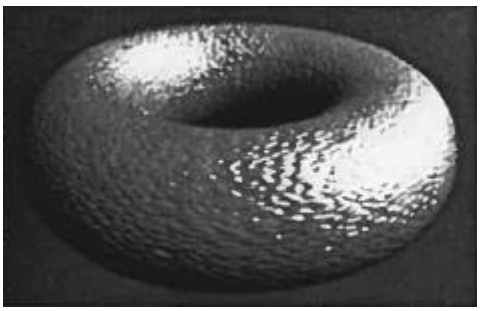


Figure 1.2: Application du bump map sur un tore[16]

4.2.2 Displacement mapping

Volino et Magnenat-Thalmann[17] proposent une méthode géométrique de génération de plis sur des surfaces qui est implémentée comme une couche supplémentaire s'ajoutant à n'importe quelle simulation de tissus. Les plis sont générés dynamiquement avant le rendu en modulant l'amplitude d'une texture de plis (heightmap) prédéfinie, pour simuler la conservation de la surface lors des déformations. Ces déformations sont calculées en comparant la longueur de chaque arête au temps présent, à leur longueur initiale. On obtient l'évolution de l'amplitude des plis lors de l'étirement ou de la compression de la surface (figure 1.3).

Figure 1.3 : Variation de l'amplitude des plis en fonction de la longueur d'une arête. [17]

Où:

L : est la longueur initiale de l'arête,
t : est sa longueur au temps présent,
γ :est un coefficient déterminant le degré de compression requis pour obtenir des plis.

1.6 4.2.3 Modèle de Peirce

C'est le modèle le plus célèbre, et bien qu'il ne soit pas complet, il sert de modèle de référence. C'est une modélisation des propriétés mécaniques des tissus à partir de la texture tissu[19].

Le modèle est défini par une armure toile, composée de fils Section circulaire. Le modèle proposé par Peirce considère que la section transversale du fil est droite et circulaire et utilisez la géométrie (ellipse) associée à cette hypothèse. Cette approximation est raisonnable pour les organisations lâches, mais pas pour les structures serrées. C'est pourquoi D'autres auteurs ont révisé leurs hypothèses sur les sections de fil, par exemple Kemp, Fournit une forme "piste de course" [19]

4.2.4 Modèle de Kawabata

Kawabata et al. Proposent un modèle pour la traction biaxiale de tissu avec une structure identique à celle de Peirce. Cependant, les fils de chaîne et de trame ont été représentés par des lignes droites qui fléchissent en deux Points P1 et P2 sur l'axe perpendiculaire au plan du tissu.[20]

4.2.5 Modèle de Grosberg

Les modèles exposées jusqu' à présent ne traitent que de la traction des tissus. En ce qui concerne la flexion, les travaux menés par Grosberg [21] sont à considérer. En effet cet auteur propose un comportement en flexion des structures textiles en s'appuyant sur un système multicouche. En fait Grosberg considère que la flexion intègre deux composantes : – une composante linéaire (résistance élastique) – une composante non linéaire (hystérésis) En formulant l'hypothèse que pour un fil la pression entre les fibres dans une zone empêche celles-ci de glisser les unes par rapport aux autres, et dans l'autre zone cette pression étant faible, il y a glissement des fibres et donc apparition d'un phénomènes d'hystérésis .[21]

4.2.6Modèled'Abbott

G.M. Abbott [22] propose une autre vision du modèle géométrique de Peirce qu'il modifie en mettant en évidence la relation entre les paramètres géométriques du modèle de Peirce et les paramètres de la composante linéaire du modèle de Grosberg. Dans un premier temps, il a développé son analyse pour une structure non relaxée: les fils extraits d'une telle structure deviennent rectilignes. Ensuite, il poursuit son étude avec des tissus en état de relaxation, cette fois les fils gardent leur déformation ondule quand on les enlevé. Ainsi celui-ci propose que la relaxation du fil soit considérée comme une perte de l'énergie élastique accumulée dans le tissu. A partir des propriétés du fil et des propriétés géométriques du tissu, Grosberg approfondit son étude au cisaillement des structures textiles. B. Olofson a repris les travaux précédents et développe un nouveau modèle de cisaillement [23]. Un autre auteur J. Skelton s'est intéresse en particulier au cisaillement des tissus [24].

2 5. Domaines d'application

Les applications de la simulation de tissu sont nombreuses, Elles sont :

Utilisées dans le monde industriel auprès des industries de la confection et du tissage,

Appliquées directement dans les jeux vidéos ou encore dans les films d'animation, en permettant notamment l'habillage de personnages virtuels.

Exploitées dans les applications de la vente en lignes de vêtements.

Les travaux dans ce domaine essayent d'augmenter le taux de réalisme du rendu et la vitesse d'exécution il cherche aussi à développer des méthodes pour simuler des tissus et des vêtements plus complexes.

2.1 6.Conclusion

La simulation des tissus a une grande importance dans l'animation par ordinateur, elle inclut la simulation de son comportement géométrique et mécanique et la simulation de leurs interactions avec l'environnement.Nous avons présenté tout au long de ce chapitre, les principaux modèles utilisés pour représenter les tissus, afin de simuler leurs formes et leurs mouvements. Les modèles géométriques, qui sont historiquement les premiers à être apparus, sont largement utilisés dans domaine du fait qu'ils fournissent des résultats plus rapides que celles basées sur les modèles physiques.

L'utilité de la simulation de tissus n'est pas limitée au domaine de l'animation par ordinateur. Une fois que la simulation de mouvement de tissu devient plus réaliste, son introduction dans d'autres domaines de la vie quotidienne devient nécessaire, notamment dans la conception de vêtements et les industries de textiles.

Le but principal des travaux de recherche dans ce domaine est de développer des modèles permettant d'accélérer les algorithmes de simulation, qui offrent la possibilité de générer des comportements réalistes de tissu et qui soient capables de concevoir et de simuler des tissus et des vêtements plus complexes. Le prochain chapitre concerne donc l'exploitation des GPU dans ce domaine afin d'accélérer le rendu.

Chapitre 02 :

Le parallélisme

1. Introduction :

Au domaine de l'informatique, le parallélisme consiste à mettre en œuvre des architectures d'électronique numérique permettant de traiter des informations de manière simultanée, ainsi que les algorithmes spécialisés pour celles-ci. Ces techniques ont pour but de réaliser le plus grand nombre d'opérations en un temps le plus petit possible.

Certains types de calculs se prêtent particulièrement bien à l'utilisation du parallélisme: la dynamique des fluides, les prédictions météorologiques, la modélisation et simulation de problèmes de dimensions plus grandes, le traitement de l'information et l'exploration de données, le décryptage de messages, la recherche de mots de passe, le traitement d'images ou la fabrication d'images de synthèse, tels que le lancer de rayon ,la simulation des ombres , l'occultation ambiante , simuler un éclairage directe ect ...[25]

Dans ce chapitre, nous décrivons les architectures parallèles, puis nous présentons des modèles de programmation parallèle, nous citons ensuite quelques travaux exploitant le parallélisme enfin nous définissons les concepts liés à l'utilisation des shaders.

2. Architectures parallèles

Une machine parallèle consiste en un ensemble de processeurs qui travaillent ensemble pour résoudre un problème important [26]. Ils offrent un énorme potentiel de ressources informatiques grâce à leurs processeurs, leur mémoire et leur bande passante. Cette définition inclut à la fois les supercalculateurs parallèles contenant des centaines ou des milliers de processeurs, ainsi que les réseaux de postes de travail et même les machines multiprocesseurs. Les architectures parallèles peuvent être divisées en plusieurs catégories selon la diversité des flux d'instructions et de données. Ainsi, Flynn [27] distingue deux types de machines parallèles : les machines SIMD (Single Instruction Multiple Data), et les machines MIMD (Multiple Instruction Multiple Data). [26]

- Les machines SIMD sont plus simples à utiliser, mais elles ne peuvent pas traiter efficacement tous les types de problèmes. Les machines MIMD ont l'avantage d'être plus polyvalentes et moins chères. C'est pourquoi la plupart des ordinateurs parallèles sont aujourd'hui des machines MIMD.
- Les machines MIMD elles-mêmes sont subdivisées en deux groupes [28], Multi-ordinateurs et multi-processeurs, ils se différencient par leur emplacement mémoire. Les (multiprocesseurs, également connus sous le nom de machines MIMD à mémoire partagée, sont caractérisés par plusieurs processeurs partageant le même espace d'adressage. [26]

2.1 Machines à mémoire partagée

Une architecture de mémoire partagée parallèle possède une seule mémoire contiguë à laquelle son ensemble de processeurs peut accéder directement via un bus ou une hiérarchie de bus (Figure 2.1). La plupart des machines à mémoire partagée sont des systèmes symétriques (SMP, multitraitement symétrique) où tous les processeurs remplissent la même fonction et se disputent uniformément les ressources système. [29]

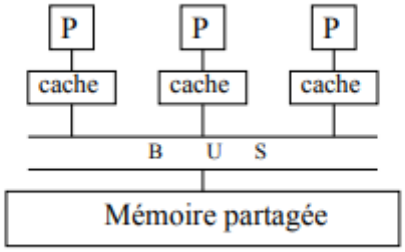


Figure 2.1 : Machine multiprocesseurs à mémoire partagée. Ou le P est un processeur. [29]

2.2 Machines à mémoire distribuée

Une machine parallèle à mémoire distribuée est constituée d'un groupe de nœuds reliés par un réseau de communication. Chaque nœud est constitué d'un processeur et d'une mémoire locale. La mémoire est ainsi répartie entre les processeurs, il n'y a donc pas d'emplacement central. Toutes les interactions entre les nœuds doivent passer par le réseau. La figure 2.2 illustre ce type de modèle d'architecture parallèle. [29]

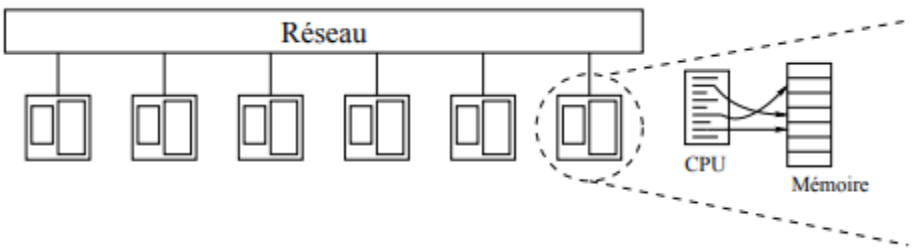


Figure 2.2 : Modélisation d'une machine parallèle MIMD à mémoire distribuée. **[29]**

Ce modèle architectural présente un asynchronisme physique qui doit être pris en compte lors de la programmation, du fait que chaque processeur fonctionne indépendamment avec sa propre mémoire locale. Par rapport à la mémoire partagée, ces architectures peuvent croître considérablement en termes de nombre de processeurs et de capacité de mémoire. [29]

2.3 Machines graphiques parallèles :

En infographie, les machines Origin et Onyx de Silicon Graphics Incorporated (SGI) représentent la norme en termes de puissance. SGI Origin 2000 est basé sur la mémoire partagée et les systèmes distribués. Ce système est appelé Scalable Shared Memory Processor (SSMP). Deux processeurs sont connectés à la mémoire partagée via un "hub" pour former un nœud. Plusieurs nœuds sont ensuite connectés à l'aide d'un réseau interconnecté pour former le système d'origine. La figure 2.3 illustre cette architecture. [29]

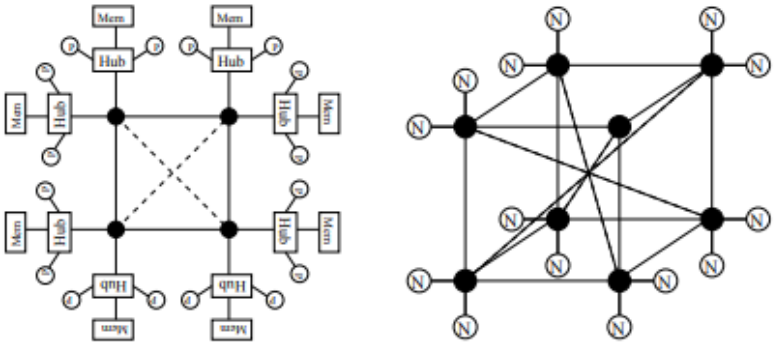


Figure 2.3 : Architecture du système SGI Origin. Deux processeurs sont connectés via un “hub” à une mémoire partagée. [29]

2.4 Grappes de machines ou clusters :

Ces dernières années, des clusters de machines ou clusters [30] ont vu le jour. Le cluster peut être limité à un cluster de PC standard connectés par un réseau haut débit (côté gauche surla Figure2.4), mais il peut également consister en un système de plusieurs machines SMP connectées entre elles via un bus de communication et d'E/Sà hautes performances (côté droit de la Figure2.4). Ce type d'architecture est appelé hybride car il combine les deux types d'architectures précédentes. [29]

Figure 2.4 : (A gauche) Grappe de PCs du projet iCluster (HP, INRIA, ID-IMAG) composée de 226 monoprocesseurs Pentium III à 733 MHz avec 256 Mo de mémoire, interconnectés via un réseau de 100 Mbit/s. [29]

3. Modèles de programmation parallèle :

Les machines parallèles performantes n'ont d'intérêt que si toutes leurs ressources matérielles sont gérables. Le modèle de programmation parallèle vous permet de connecter le développement d'applications avec un type d'architecture particulier sans considérer que les détails techniques peuvent différer. Ces modèles peuvent être catégorisés selon différents niveaux de parallélisme [30].

- Un modèle avec parallélisme implicite. Toutes les activités liées à l'exécution parallèle restent complètement cachées au programmeur.
- Un modèle avec parallélisme explicite. Le traitement parallèle est explicitement décrit sans définir comment l'application est découpée en tâches, leur ordonnancement ou leur communication[30].
- Un modèle avec décomposition explicite. Le langage fournit des primitives explicites pour définir des tâches qui s'exécutent en parallèle.
- Modèle de placement explicite. Le programmeur prend en charge la décomposition en diverses tâches et implémente le déploiement sur le processeur.
- Modèle événementiel. Le mouvement et la synchronisation des données ne peuvent être effectués qu'en générant certains événements qui déclenchent le processus [31]. Ces modèles peuvent être catégorisés selon différents niveaux de parallélisme [31].

3.1Parallélisme de données

Le parallélisme des données est ainsi appelé car il tire parti du parallélisme qui résulte de l'exécution de la même opération sur différents éléments d'une structure de données. Lors de l'expression du traitement parallèle de cette manière, le travail du processeur dépend de la distribution des données. La communication est définie comme la phase de redistribution des données sur le processeur. Un programme parallèle est un ensemble de phases de calcul et de communication pour redistribuer des données. Cette source de parallélisme est souvent utilisée dans la manipulation vectorielle, ce qui est courant dans les problèmes d'algèbre linéaire dense. [31]

3.2Parallélisme de contrôle

Ce modèle décrit l'algorithme parallèle comme un graphe orienté sans cycle ou les sommets de ce graphe représentent les taches qui peuvent être exécuté d'une manière séquentielle et les arcs représentent les règles de précédences entre les taches (taches ordonnées), les taches qui ne sont pas ordonnées dans ce graphe et qui n'acceptent pas les règles du précédence sont exécutées en parallèle.

Le parallélisme de donnée peut être considéré comme un cas particulier de ce parallélisme, et aussi dans ce modèle différents paradigmes de programmation sont usuellement utilisées. [31]

3.3Modèle à mémoire partagée

La programmation avec mémoire partagée convient pour une utilisation avec des machines SMP, c'est-à-dire avec des machines multiprocesseurs. La communication se fait directement via des variables partagées contrôlées par le mécanisme de synchronisation.(Exemple : sémaphore et verrous). La plupart des systèmes d'exploitation mettent à disposition une interface de programmation implémentant ce modèle. L'inconvénient est que vous êtes trop lié au style de l'architecture de la machine, ce qui rend le programme moins portable (l'aspect de portabilité est réduit). [31]

3.4Modèle par échange de messages

Le passage de messages est un modèle de base qui permet la communication entre les processeurs dans une architecture de mémoire distribuée. Il est implémenté à l'aide de primitives d'envoi et de réception avec l'identifiant du processeur du partenaire de communication et le message envoyé comme paramètres. Les problèmes de contrôle du flux de messagerie sont très importants dans ce modèle. Il existe deux types de sémantique : synchrone (utilisant le rendez-vous) et asynchrone (limitation du canal de communication).

4. Travaux précédents dans la Simulations parallèles :

Dans cette partie on va mentionner le résumé de quelques travaux précédemment réalisés, qui ont utilisés le parallélisme dans le domaine de simulation de textiles, et traitement d'image ,nous ne citons ces travaux qu'à titre indicatif afin de montrer les différents travaux réalisé dans le domaine de la simulation textile qui est vaste et varié et aussi inclus dans différents domaines de recherches . [31]

Titre	Réalisé par	Résumé
Algorithmes parallèles de simulation physique pour la synthèse d'images : application à l'animation de textiles.	Florence Zara	• Ce projet de thèse a contribué à l'obtention de nouvelles structures algorithmiques parallèles efficaces avec l'obtention d'algorithmes asynchrones. • Ce travail combine le calcul haute performance à la réalité virtuelle par son apport de méthodes de calcul parallèle pour l'animation d'objets 3D en synthèse d'image.
a. Une Méthode De Parallélisation Des Algorithmes De Traitement D'images	HenniAbderrazak. Saouli Rachida	Ce travail propose une méthode de parallélisation d'algorithmes séquentiels du traitement d'images.

Tableau 2.1 : travaux précédents l'introduction du parallélisme au domaine de l'infographie [tab3]

5. Les shaders GPU

5.1Introduction au shaders :

Les shaders sont également un jeu d'instructions, mais contrairement à l'API de dessin normale, toutes les instructions sont exécutées sur chaque pixel de l'écran. Cela signifie que le code se comporte différemment selon la position des pixels sur l'écran. Comme un moteur

d'impression, votre programme est une fonction qui nous transmet des positions et renvoie des couleurs, et une fois compilé, ce programme peut s'exécuter très rapidement.

Pourquoi utiliser les shaders

Pour répondre à cela, on va surement parler des merveilles du parallèlisme.

Considérez le processeur de l'ordinateur comme un tube. Chaque opération qui sera effectuée est effectuée comme quelque chose dans le tube, comme une usine. Certaines opérations sont sans doute plus importantes que d'autres et nécessitent beaucoup de temps et d'énergie pour être traitées. On dit qu'ils nécessitent plus de ressources et de puissance de calcul. Du fait de l'architecture de l'ordinateur, les opérations sont effectuées en série. Chaque opération doit être terminée avant que la CPU puisse traiter l'opération suivante. Les ordinateurs les plus récents ont généralement plusieurs processeurs qui agissent comme des canaux, ce qui vous permet d'effectuer des opérations de manière séquentielle tout en conservant une certaine liquidité. Ces tubes sont appelés threads.

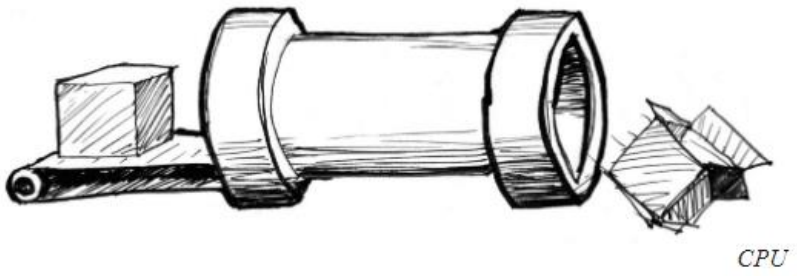


Figure5.1 : la représentation du processeur comme un tube [31]

Les jeux vidéo et autres applications graphiques nécessitent plus de puissance de calcul que les autres programmes. Chaque fois que vous changez l'image, vous devez recalculer tous les pixels de l'écran Chaque pixel à l'écran représente une simple petite opération. En soi, le traitement d'une opération n'est pas un problème pour le CPU, mais (et c'est bien le problème) il faut appliquer cette petite opération à chaque pixel de l'écran !

Figure 5.2 : la façon dont le processeur traite les pixels sur écran[31]

C'est là qu'intervient le traitement parallèle. Plutôt que d'avoir plusieurs gros microprocesseurs ou pipelines puissants, nous préférerions avoir plusieurs petits microprocesseurs fonctionnant en parallèle et simultanément. C'est l'essence même du GPU (unité de traitement graphique).

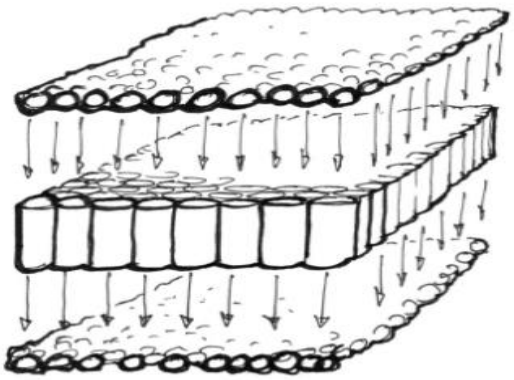


Figure 5.3 : Le traitement de GPU[31]

Les shaders peuvent être exécuter sur certain environnements de travail ,code blocs, eclipse,ogre3D, Visual studio .

Le langage de programmation le plus prêt à sa syntaxe c'est bien le c++ ,grâce à sa souplesse et la variété des bibliothèques qui sont bien structuré avec le Language de programmation utilisés par les shaders (Le glsl).

6. Le glsl

OpenGL Shading Language (GLSL) est un langage de programmation de shader de haut niveau dont la syntaxe est basée sur le langage C. Les shaders permettent un contrôle avancé du pipeline graphique. GLSL a été développé par l'OpenGL Architecture Review Board pour faciliter la programmation des shaders à l'aide de l'API OpenGL sans utiliser le langage d'assemblage ARB ou des langages spécifiques au matériel.[32]

6.1Le pipeline classique

Lorsque tous les sommets d'une primitive sont connus, le pipeline de rendu commence à fonctionner. L'image finale est construite en appliquant un pipeline à chaque primitive. Canalisation classique Décomposé en 5 étapes : [32]

1. Transitions de vue de modèle.
2. Eclairage et teinte du vertex.
3. extrait visuel.
4. projection.
5. Rastérisation

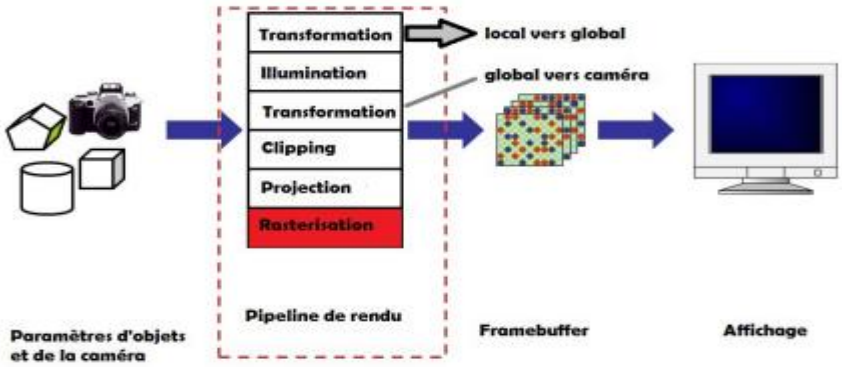


Figure 6.1 : Le rendu classique. [32]

6.2Le pipeline programmable

Contrairement au pipeline classique, le pipeline programmable des GPU modernes fournit des fonctions qui Peuvent être utilisé, géré et modifié par des programmeurs. L'utilité de pipeline programmable est de créer des méthodes d'éclairages très évoluées en établissant la relation entre les différents étape du pipeline.[32]

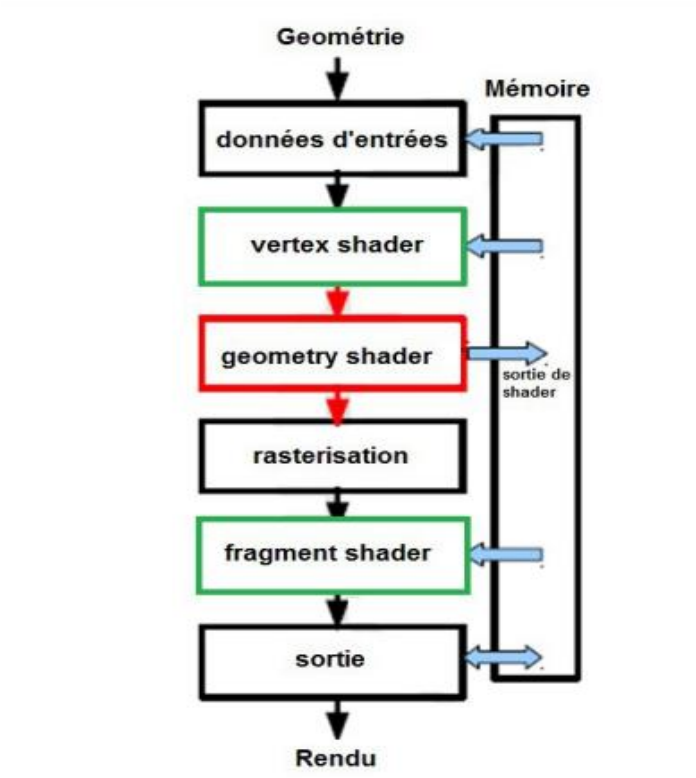


Figure 6.2 : Le pipeline du rendu programmable en utilisant les GPU shaders .[32]

Dans le pipeline graphique programmable, il y a trois nouvelles étapes qui peuvent être gérées par le programmeur :

Vertex shaders, fragment shaders et geometryshaders.

Vertex Shaders : transformations et éclairage.

GeometryShaders : Assemblage de primitives.

Fragment shader : Ombrage et texturation.

Les étapes de pipeline graphique programmable permettent de créer un programme pour paramétrer le rendu de la scène.

Étape 1 : shader de vertex

Dans cette étape, le pipeline traite les sommets et leurs attributs. Les attributs des sommets sont composés d'une couleur, d'une normale et des coordonnées de texture. Cette étape prend un vertex à part pour travailler dessus. Pour un triangle alors le vertex shader sera exécuté 3 fois.Le rôle principal de vertex shader est de transformer les vertex (position, attributs) pour les positionner dans le repère de la caméra.[32]

Étape 2 : Geometry shader

Après la création d'un vertex shader pour chaque sommet de triangle de l'objet, l'étape de geometry shader consiste à l'assemblage des primitives de modèle 3D et de réaliser les opérations d'ajout / retrait de sommets et primitives.[32]

Etape 3 : Fragment shader

Cette étape permet de définir la couleur de chaque pixel de la forme délimitée par les vertex. Elle représente un pixel de l'image résultat ainsi que ces attributs. Le rôle principal du fragment shader est de déterminer l'apparence de l'objet qui se projette sur le pixel traité. La couleur, la transparence et la profondeur du fragment sont des paramètres nécessaires pour réaliser ce calcul.[32].

6.3 Les limites de l'utilisation des shaders

En raison de leur nature parallèle, les shaders ne traitent pas les informations comme le fait un programme classique. Le code du shader s'exécute sur chaque sommet ou pixel de manière isolée. Vous ne pouvez pas non plus stocker des données entre les images. Par conséquent, lorsque vous travaillez avec des shaders, vous devez coder et penser différemment des autres langages de programmation.

.

7. Conclusion :

architectures parallèles et la programmation parallèle. Nous avons aussi détaillé les shaders qui sont un type particulier de programme qui fonctionne sur les unités de traitement graphique (GPU).Les shaders étaient initialement utilisés pour ombrer les scènes 3D mais peuvent aujourd'hui faire beaucoup plus. Nous pouvons les utiliser pour contrôler la façon dont le moteur dessine la géométrie et les pixels à l'écran, ce qui nous permet d'obtenir toutes sortes d'effets. Nous allons donc exploiter les shaders pour optimiser le rendu de tissu, le chapitre suivant sera consacré à la conception et l'implémentation de notre système.

Chapitre 03 :

Conception et implémentation

Chapitre 3 :

Conception et implémentation

1.Introduction

La conception est une phase importante dans le processus de développement des applications, elle permet de décrire l'état interne du système et de donner une vision globale pour ceux qui vont lire ou bien utiliser le système.Pour cela,dans la première partie de ce chapitre nous allons expliquer l'architecture globale et détaillé du système qui est composé de plusieurs modules.

Tandis que dans la deuxième partie du chapitre, nous allons détailler les différentes étapes de mise en œuvre de notre système. Nous commençons par la description des différentes structures de données, et nous décrivons ensuite les algorithmes de base nécessaires à l'implémentation du système. Les résultats obtenus sont illustrés à la fin de ce chapitre avec une évaluation.

2.Objectifs et présentation du système

L'objectif principal de notre système consiste à accélérer le rendu d'une animation de tissu en exploitant les shaders.Pour répondre à ce besoin, nous devons concevoir un système qui prend en charge :

- La représentation d'un tissu sous forme d'un modèlegéométrique.
- Le choix d'un modèled'éclairage.
- La définition deséléments du shader (fragment et le vertex).
- l'animation d'une pièce de tissu (Sur CPU et GPU).
- Le plaquage de texture (Sur CPU et GPU).
- Faire une comparaison entre les résultats obtenus en CPU et GPU.

Pour cela, Notre conception va être décomposé en un ensemble de modules, où chacun possède un ensemble de paramètres en entrée et produit un ensemble de résultats en sortie.

2.1 Laconception globale

La conception globale vise à représenter l'état du système en générale.

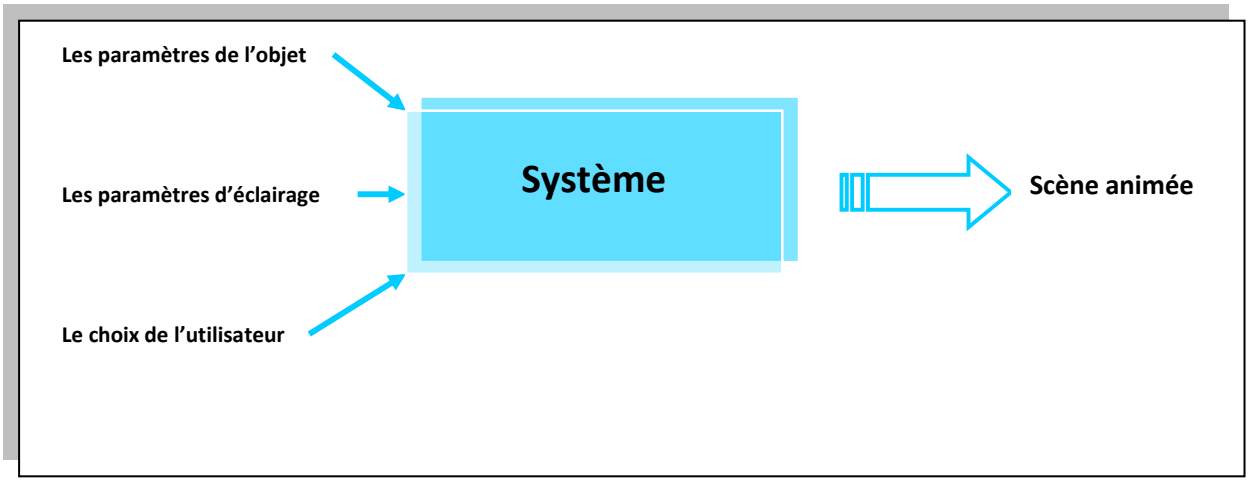


Schéma 3.1 - Présentation du système.

Les entrées du système :

- Les paramètres de l'objet (maillage de points et position des sommets).
- Les paramètres d'éclairage (Le modèle de phong).
- Le choix d'utilisateur qui est le choix de l'exécution (avec ou sans shaders).

La sortie :

- Une scène animée (une pièce de tissu animée).

Dans cette partie on va présenter les modules de notre système qui sont des étapes complémentaire pour avoir le résultat final.

La première étape est la modélisation d'un système d'éclairage, puis la représentation de l'élément à étudier qui est un tissu représenté sous forme d'un maillage de points, ce dernier est représenté d'une manière géométrique, après on va commencer à mettre en œuvre le shader pour que les paramètres de l'objet seront envoyé à ce dernier afin d'appliquer une texture et établir une animation sur l'ensemble de point qui construisent l'objet à simuler .

Le schéma suivant illustre les modules de notre système

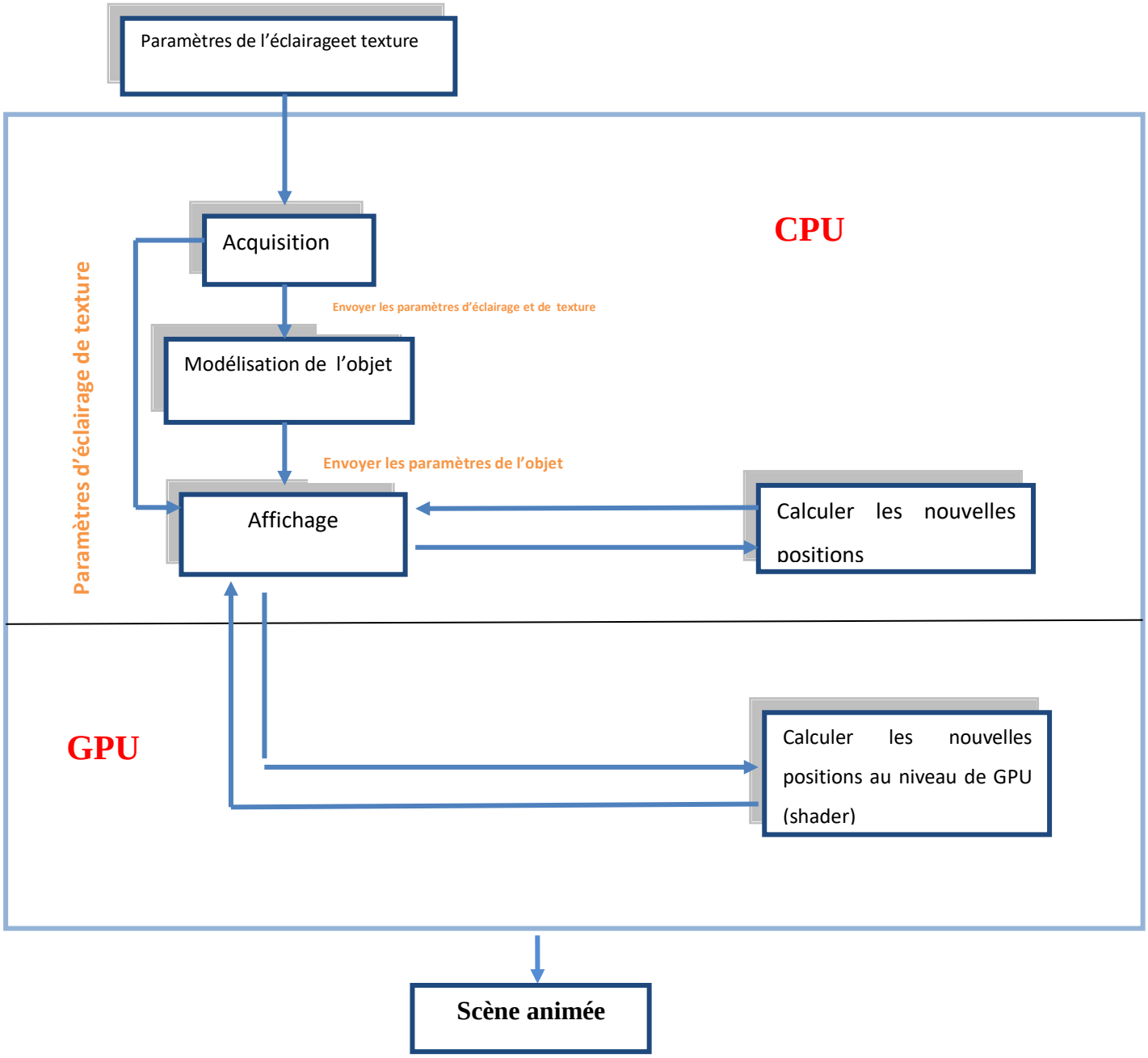


Figure 3.2 : Présentation de l'architecture des modules de notre système

2.2La conception détaillée

La conception détaillée sert à montrer l'architecture détaillée du système en discutant les modules qui le composent.

2.2.1Le module d'acquisition

Ce module est nécessaire dans ce travail, c'est dans ce module qu'on doit initialiser les paramètres de texture et d'éclairage et tous qu'on a besoin d'initialisation.

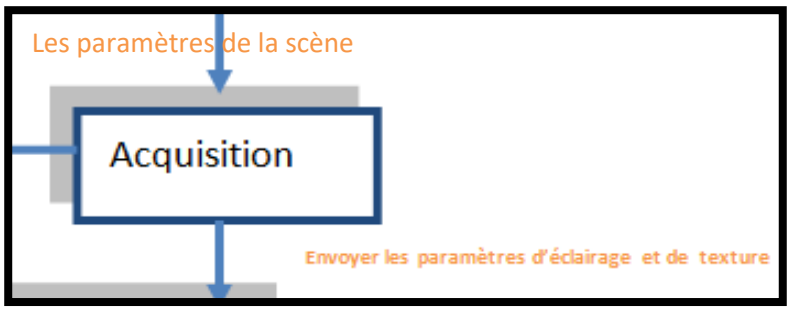


Figure 3.3 : Le module d'**Acquisition**

2.2.2Le module de modélisation de l'objet :

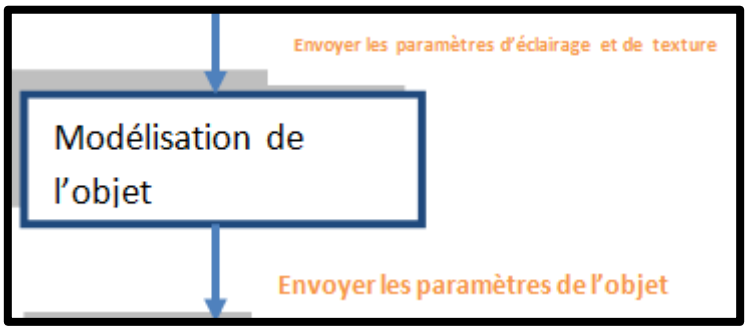


Figure 3.4 : La représentation du module **Modélisation de l'objet**

Ce module consiste à concevoir un objet qui peut être à la suite texturé et animé, l'objet est modélisé par un modèlegéométrique qui est un ensemble de points ou chaque point à ses cordonnées qui vont être en suite utilisées pour faire l'animation et la texture.

2.2.3Le module : Calculer nouvelles positions des points

Ce module va être la partie associée à l'animation qui est appliquée sur le CPU et GPU, après avoir modélisé l'objet,ses paramètres vont être une entrée à ce module puis le premier frame sera affiché sur CPU comme premier résultat puis la méthode utilisé consiste à donner un mouvement aux points en appliquant une formule mathématique qui consiste à modifier la

position des points selon une fonction sinus afin d'avoir un mouvement de tissus sous la force du vent.

Cette fonction est programmée comme suit :

L'initialisation des coordonnées de l'objet ou chaque composante prends une valeur initiale puis en utilisant une boucle afin de parcourir tous les sommets composants l'objet et en appliquant une formule sur ces derniers, un mouvement va être affiché comme animation et sa se répète jusqu'à la fin de l'animation.

Donc notre animation est une boucle qui fait parcourir tous les sommets de l'objet puis lui appliquer une animation.

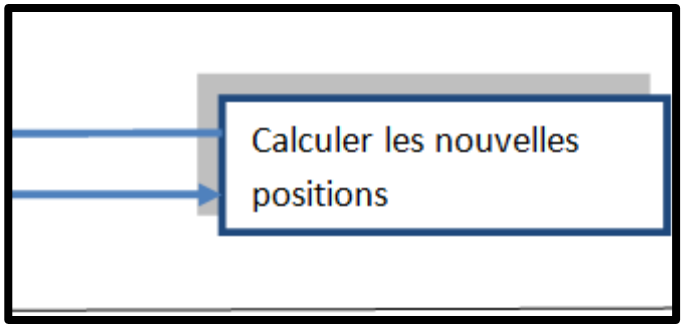


Figure 3.6 : La représentation du module de calcul de nouvelles positions

2.2.4Le module d'affichage

Ce module prend en charge de mettre à la disposition de l'utilisateur de voir les résultats voulus après l'exécution.Il nécessite la description de quelques attributs géométriques et permet de décrire la façon dont la vue est cadrée pour obtenir l'image finale.

A chaque pas de temps et après le calcul des nouvelles positions de l'objet, il faut afficher le rendu pour voir les nouvelles positions des sommets. L'affichage successif des points du tissu spécifie l'évolution de l'objet dans le temps.

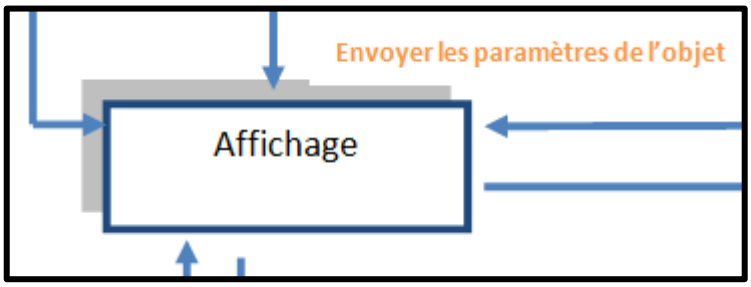


Figure 3.9: La représentation du module Affichage

Ces modules illustrés concernent juste l'animation en CPU

On passe maintenant à expliquer comment le rendu est fait sur le GPU

Au niveau de GPU il ya deux partie qui se collaborent entre eux afin de nous réaliser une animation ou bien un rendu, ces deux parties sont complémentaires et forment un shader, le shader donc est formé de deux parties chacune a son propre rôle.

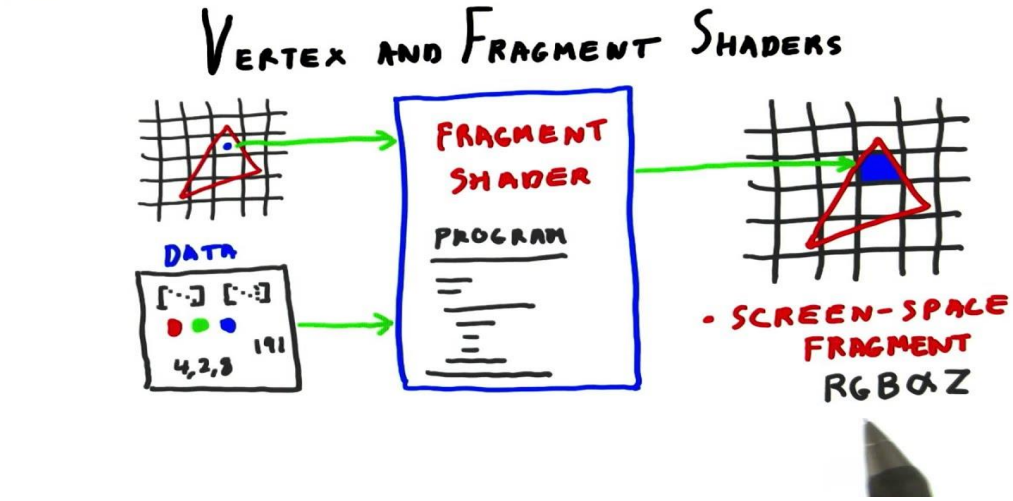


Figure 3.10 : le passage entre un vertex shader vers un fragment shader

- **Vertex shader**

Est une fonction programmable dans les cartes graphiques qui offre une flexibilité au rendu des images il est utilisé pour transformer les attributs des sommets (vertex) comme la couleur la texture la position. Il permet à l'objet original à être redéfinis en plusieurs manières.

La sortie du vertex shader avec une texture map subit une interpolation puis se transforme en un pixel shader qui est une autre fonction programmable qui permet aussi la flexibilité en faisant le dessin d'un pixel individuel et comme le vertex shader est utilisé pour déformer les objets, le pixel shader est utilisé pour changer l'apparence de pixel.

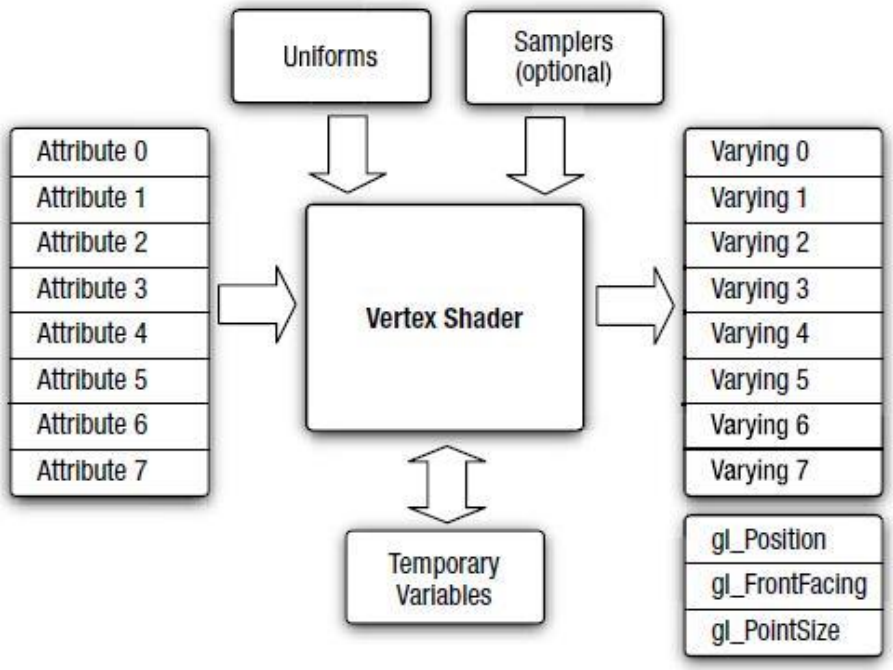


Figure 3.11: Les entrée et sorties de vertex shader

- **Fragment shader**

Un Fragment Shader est une fonction qui traitera un fragment généré par la rastérisation en un ensemble de couleurs et une seule valeur de profondeur.

Le fragmentshaderest l'étape du pipeline OpenGL après qu'une primitive a été pixellisée. Pour chaque échantillon des pixels couverts par une primitive, un "fragment" est généré. Chaque fragment a une position d'espace de fenêtre, quelques autres valeurs, et il contient toutes les valeurs de sortie interpolées par sommet de la dernière étape de traitement des sommets.

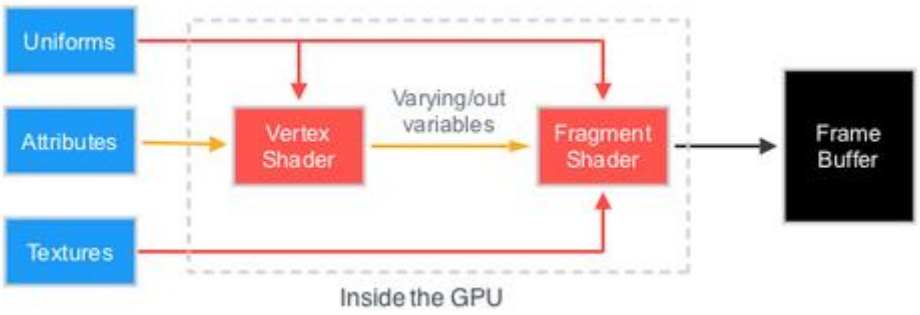


Figure 3.12 : le processus de déroulement d'un fragment shader

3.La mise en œuvre

Dans ce chapitre, on va parler sur les outils exploités afin de réaliser ce travail et aussi voir le contenu de projet (le code, fonction utilisées) et les résultats obtenus par ce projet.

Nous présentons d'abords les outils de développement qu'on a utilisé puis on va détailler un peux pour montrer les différents modules (fonctions) et en fin on donnera un aperçu de la scène via une interface graphique.

3.1Langage de programmation utilisée

3.1.1 L'environnement de travail

- **Microsoft Visual c++**

Visual C++ est un environnement de développement intégré (IDE) pour compléter la programmation C, C++. C'est un logiciel commercial avec une version gratuite disponible. Visual C++ contient divers outils de gestion de base de code et outils de développement à utiliser avec les interfaces de programmation d'application (API) Microsoft Windows et la plate-forme Microsoft .NET.

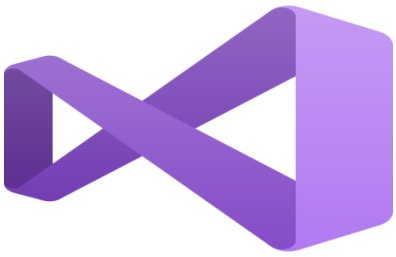


Figure 3.13 : Image illustrative au logiciel Visual C++

3.1.2La structure de données

La structure de donnée utilisée pour représenter le tissu est un tableau de point ou chaque point est un vertex qui est représenté avec trois cordonnées (x et y) ou z est fixe.

```
glBegin(GL_QUADS);
    for (x=0; x<100; x++) {
        for (y=0; y<100; y++) {
            double_x = double(x) / 100.0f;
            double_y = double(y) / 100.0f;
            double_xb = double(x + 1) / 100.0f;
            double_yb = double(y + 1) / 100.0f;

            glTexCoord2f(double_x, double_y);
            glVertex3f(points[x][y][0], points[x][y][1], points[x][y][2]);

            glTexCoord2f(double_x, double_yb);
            glVertex3f(points[x][y+1][0], points[x][y+1][1], points[x][y+1][2]);

            glTexCoord2f(double_xb, double_yb);
            glVertex3f(points[x+1][y+1][0], points[x+1][y+1][1], points[x+1][y+1][2]);

            glTexCoord2f(double_xb, double_y);
            glVertex3f(points[x+1][y][0], points[x+1][y][1], points[x+1][y][2]);
        }
    }
glEnd();
```

Figure 3.14 : la représentation de la structure de donnée utilisée pour la modélisation de l'objet.

Les points sont normalisés pour que l'ensemble de point à créer ne dépasse pas la hauteur et la largeur de l'objet, ou ces derniers peuvent être édités de l'interface graphique. Les points qui sont les bornes de l'objet sont prédéfinis puis la suite des points entre les points de bornes sont générés. Cet objet va être exploité plus tard par le shader pour lui appliquer une texture et l'animé ensuite.

Initialisation des sommets représentant la pièce de tissu

L'initialisation des sommets (x , y et z) illustrés dans cette figure :

```
for (int x=0; x<101; x++) {
    for (int y=0; y<101; y++) {
        points[x][y][0] = double((x / 3.0f) -12);
        points[x][y][1] = double((y / 5.0f) - 9);
        points[x][y][2] = double(sin((((x / 8.0f) * 45.0f) / 360.0f) * pi * 2.0f));
    }
}
```

Figure 3.15 : L'initialisation des sommets de l'objet

L'animation de tissu

La boucle qui représente l'animation de l'objet.

```
    if(choix==1)
    {
        int  val;
    if (wave_count==1) {
        for (y=0; y<101; y++) {
            hold = points[0][y][2];

            for (x=0; x<100; x++) {
                int  val =sin(4.0*x)*0.5;

                points[x][y][2] =points[x+1][y][2];

                }

            points[100][y][2] = hold;
        }

        wave_count = 0;
    }


    wave_count=wave_count+1;
    }

    fps();
}
```

Figure 3.16 : la boucle qui anime l'objet au niveau de CPU

3.1.3L'installation du shader

On commence par ce qui doit être écrit dans le programme main :

- La déclaration de deux pointeurs de type char vers le vertex et fragment shader **(vs,fs)**

```
char *vs = NULL,*fs = NULL;
```

- La lecture de code qui est dans le vertex et le fragment
  vs= textFileRead("Shader1.vert");

  fs = textFileRead("Shader1.frag");

- La création d'un objet programme :

```
p = glCreateProgramObjectARB ();
```

- Creation d'un objet shader

```
v = glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);
      f = glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);
```

- Attacher chaque objet shader à l'objet program

```
glAttachObjectARB(p,v);
      glAttachObjectARB(p,f);
```

- Lier le source à chaque objet shader

```
constchar * vv = vs;
	constchar * ff = fs;
glShaderSourceARB (v, 1, &vv,NULL);
	glShaderSourceARB (f, 1, &ff,NULL);
```

- Libérer les shaders

```
free(vs);free(fs);
```

- Compiler chaque objet shader

```
glCompileShaderARB(v);
	glCompileShaderARB(f);

printInfoLog(v);
	printInfoLog(f);
```

- Lier les objets shaders entre eux dans l'objet program

```
glLinkProgramARB(p);
	printInfoLog(p);
```

- Sélectionner l'objet program

```
glUseProgramObjectARB(p);
```

De cette façon le shader est prêt à être utilisé

4.Les algorithmes :

4.1La procédure On init () :

Entrée : les paramètres de l'objet et ceux de l'éclairage et texture

- Cette procédure permet de faire l'initialisation des paramètres qui concerne la scène :

glEnable(GL_DEPTH_TEST);
Activer de z buffer qui est predefinit sur opengl .
glEnable(GL_TEXTURE_2D);
Activer ma texture afin de pouvoir l'appliquer plutard
glEnable(GL_LIGHTING);
Activer l'éclairage de la scène.
glClearColor(0.0, 0.0, 0.0, 0.0);
Initialiser la couleur de la console .

4.2La procédure changeSize(w, h)

Dsfq$glViewport(0, 0, w, h);
Permet de faire un view port sur toute la fenêtre

gluPerspective(120,ratio,1,100);
glMatrixMode(GL_MODELVIEW);
Permet de faire une projection perspective pour voire la scène animée correctement

4.3La procédurerenderScene(void)

Dans cette procédure se fait un rendu qui va générer à la fin une scène animée

```
glUseProgram(p);
```

Premièrement, il faut faire appel aux fonctions citées dans la figure 3.14 pour paramétrer la camera et l'écran et aussi activer le shader à utiliser glUseProgram(p). .

```
glUniform1i(loc, wave_count);
glUniform1i(glGetUniformLocation(p, "choix"), choix);
glUniform1i(glGetUniformLocation(p, "time"), time);
glUniform1i(glGetUniformLocation(p, "Texture1"), 0);
```

A cette étape un passage de paramètres va êtreeffectuer de programme main au shader pour utiliser ces dernier sur GPU.

Pour plaquer la texture, on a utilisé ces deux instructions

```
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, g_Texture2);
```

4.4Proceduremouse(intbutton, int state, int x, int y);

Qui permet de manipuler la scène à travers la souris .

4.5Procedure processNormalKeys(unsignedchar key, int x, int y);

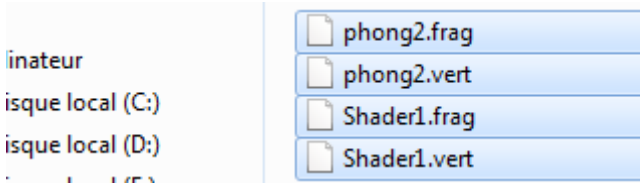
Qui permet de faire une manipulation sur la scène à travers les touches du clavier.

```
if(key=='p')
    choix=1;
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glClearColor(0.35f, 0.53f, 0.7f, 1.0f);
glLoadIdentity();
if(key=='o')
    choix=0;
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glClearColor(0.35f, 0.53f, 0.7f, 1.0f);
glLoadIdentity();
```

Nous avons défini une variable choix pour qu'on pourra voir la différence entre l'animation Sur le CPU et GPU ou :
Si choix ==1 ----------L'animation est sur le GPU et si choix==0 L'animation est sur Le CPU.

5.Les shaders utilisés :

Nous avons utilisés deux shaders un pour le plaquage de texture et l'animation et l'autre pour appliquer un modèle d'éclairage ou ce dernier est le model de PHONG .



Ces shaders sont des fichiers texte qu'on peut manipuler avec tous types d'éditeurs de texte comme NotePad.

- **Shader1 .vertex**

```
varying vec4 tex0;
 uniform int wave_count;
 uniform int time;
 uniform int choix;
void main()
{

    tex0 = gl_MultiTexCoord0;//recuperer les coord _text

    if(choix==1)
    {
        vec4 v = gl_Vertex;
        v.z=sin(4.0*v.z+time*0.08)*1.5;

    gl_Position = gl_ModelViewProjectionMatrix * v;
    }
    else
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

    vec4 pospoint=gl_ModelViewMatrix *gl_Vertex;

}
```

Ce shader manipule les pixels sur l'écran

Ou :

vec4 v :est un vecteur qui va avoir comme valeur le vertex qui est le sommet à traiter **.**

gl_Position : manipule la position du sommet sur l'écran.

gl_ModelViewProjectionMatrix : C'est la matrice de projection ou on doit faire la projection pour que la rastérisation se réalise (transformation du vertex vers un pixel).

- **Shader1.frag :**

```
#version 120
uniform sampler2D Texture1;
 varying vec3 lumiere;
 varying vec3 normal;


varying vec4 tex0;


void main()
{
    
    vec4 image0 = texture2D(Texture1, tex0.st);




    gl_FragColor =image0 ;


}
```

Au niveau de ce fragment, la texture va s'appliquer sur l'objet à travers cette instruction.

vec4 image0 = texture2D (Texture1, tex0.st);

et elle sera afficher par cette instruction :gl_FragColor =image0 ;

qui veut dire que chaque pixel prend la couleur de l'image qui est appliquer sur l'objet

Parlons maintenant sur le modèle de PHONG :

6.Model de Phong :

C'est un modèle d'éclairage direct appliqué sur les polygones ou maillage polygonal : définition par les sommets basé sur l'interpolation des normales entre les sommets. Ce modèle consiste à calculer une normale moyenne pour chaque sommet (lissage)

Et puis calculer l'intensité entre 2 sommets = interpolation linéaire des normales puis le calcul de l'intensité,son principe vise à Interpoler des normales plutôt que des couleurs, c'est une méthode d'interpolation identique, ce model utilise trois composantes nécessaire afin d'avoir un éclairage (composante ambiante, diffuse et spéculaire). L'intensité réfléchie(diffuse) n'est calculée qu'après interpolation des normales.

Comme résultat, il donne un effet spéculaire meilleur que celui de model de Gouraud.

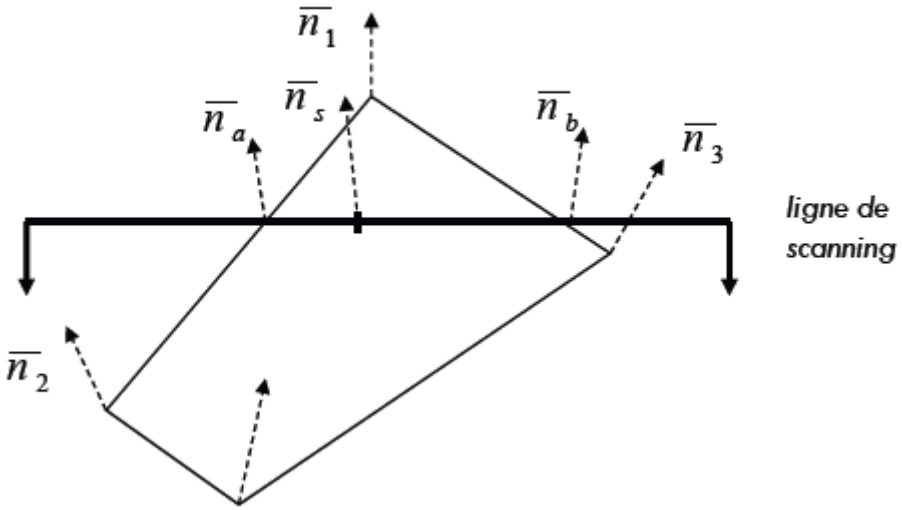


Figure 3.17 : L'interpolation dans le modele Phong

Etapes pour faire l'interpolation des normales calculées aux sommets de la face :

1. Calcul des normales aux sommets.
2. Interpolation des normales lors de la rastérisation.
3. Normalisation des normales.
4. Calcul des intensités par fragment.

- **Phong. Vertex**

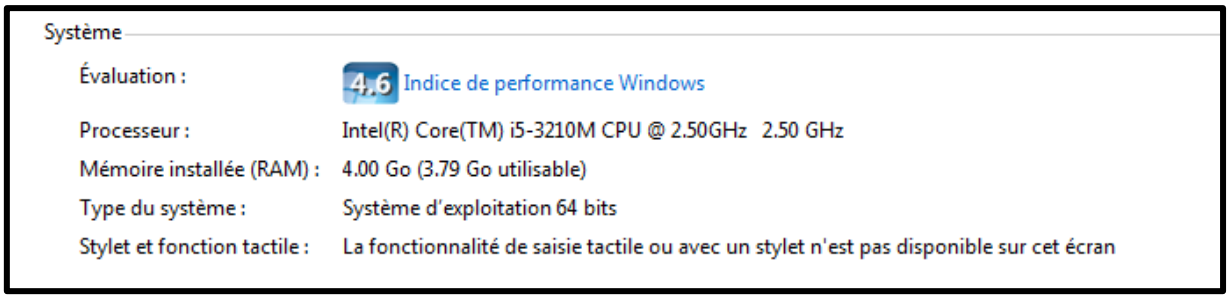
```
#version 120
#pragma optionNV(unroll all)
uniform vec3 position = vec3(0.0,15.0,10.0);
varying vec3 N;
varying vec3 s;
varying vec3 v;
varying vec3 p;
uniform vec3 source;
void main(void)
{
      vec3 v = vec3(gl_ModelViewMatrix * gl_Vertex);
    gl_Position = gl_ModelViewProjectionMatrix *  gl_Vertex;

    N=normalize(gl_Normal);
    s=normalize(vec3 (source- p));
  }
```

- **Phong .fragment**

```
#version 120
#pragma optionNV(unroll all)

varying vec3 v;
varying vec3 N;
varying vec3 s;

 vec4 ld= vec4(1.0, 1.0, 1.0, 1.0);
 vec4 kd= vec4(1.0, 1.0, 1.0, 1.0);//(0.12, 0.09, 0.81, 1.0);
 vec4 la= vec4(0.74, 0.76, 0.13, 1.0);
 vec4 ka= vec4(0.87, 0.05, 0.03, 1.0);
 vec4 ls= vec4(1.0, 0.0, 0.0, 1.0);
 vec4 ks= vec4(0.2, 0.2, 0.2, 1.0);

vec3 r;
void main(void)
{

 //  r = 2.0*dot(N,s)*N-s;
r=-s+2.0*dot(s,N)*N;
     r = normalize(r);
 vec4 amb  = ka*la;
  vec4 diff0 = kd*ld * max(dot(N,s), 0.0) ;
   vec4 d0 = clamp(diff0, 0.0, 1.0);

   vec4 spec0 =  clamp(ks*ls*pow(max(dot(v ,r), 0.0),1.50), 0.0, 1.0 );

 gl_FragColor = amb+d0+spec0;

}
```

7.Résultats

Dans cette partie, on va citer les résultats que nous avons obtenus après l'exécution de notre application, Nous avons implémenté notre système dans un ordinateur ayant les caractéristiques suivantes :



Resultats obtenus

On a déjà mentionnés que notre travail consiste à optimiser la simulation de tissus en exploitatant le GPU (shaders) pour cela , nous avon apliqués l'animation sur CPU et GPU afin de voir l'effet du Gpu sur m'objet et voir la difference avec celle appliquée sur le CPU .

Resultat sur CPU :

L'animation était lourde a chaque fois qu'on fais augmenter la resolution des points (le nombres de sommets ou point construisant l'objet .

Comme il est apparu dans la **figure (3.18)**

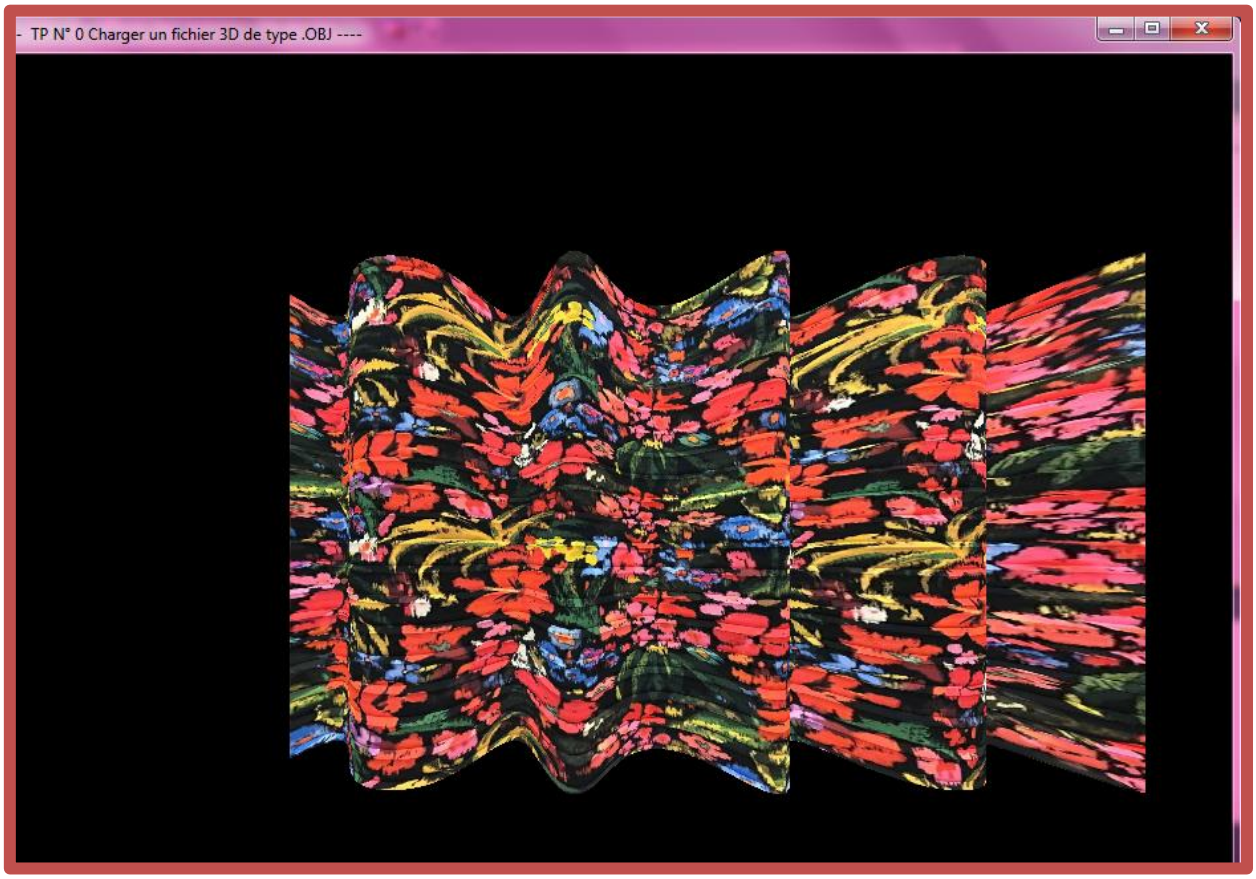


Figure 3.18 : resultat d'animation sur CPU

Resultat pour GPU (Shader)

Dans ce niveau , l'animation de l'objet était rapide ansi qu'elle semble realiste par rapport ce que le resultat precedent a donné.

Figure 3.19 : Le resultat d'animation sur GPU(Shader)

Conclusion

Dans ce chapitre nous avons focalisé sur la conception de système qui consiste à faire une optimisation de la simulation de tissu en utilisant les GPU ,ce dernier est simple ,contient un objet modelisé avec un modele geometrique basé sur une structure de points . Nous avons commencer par par motionner la représentation de la conception globale de système puis nous avons détaillé dans les différents modules qui sont nécessaires pour la réalisation de notre système.

Conclusion generale

La simulation de tissus est un sujet d'actualité dans le domaine de l'infographie. L'importance de lancer des simulations de différents éléments avec différents modèles (géométriques, physiques ect…..) différentes situations réside dans l'impossibilité d'accomplir d'une manière rapide une animation réel avec des humains. Pour cette raison, l'animation des objets textiles concerne beaucoup de domaines d'applications tels que Le cinéma, les jeux video par exemple La production de films et La production du jeu. Ainsi que avoir une animation réelle et qui réponds au temps d'exécution nécessite un potentiel de calcul grace au nombre énorme des sommets construisant l'objet, et un matériel puissant (machine puissante avec des ressources puissantes), ce qui nous a fais penser à optimiser l'animation textile en exploitant la puissance de GPU.

Notre travail s'inscrit dans le cadre d'optimisation de tissus en exploitant le GPU. L'objectif principal était d'implémenter une structure de donnée cohérente qui permet de modéliser un objet géométrique puis appliquer a ce dernier une texture au niveau de GPU et aussi lui appliquer une animation , la simulation d'une animation de tissus d'une façon plus réaliste, Notre implémentation est basée principalement sur le modèle géométrique pour modéliser initialement un objet puis l'animer en exploitant le GPU pour reproduire les mouvement de l'objet et avoir une animation cohérente avec un temps d'exécution réduit afin d'avoir une simulation dite optimisée .

Les résultats Obtenus sont satisfaisants mais il reste quelque perspectives afin d'améliorer le travail.

-appliquer l'animation à plusieurs objets.

-Améliorer l'aperçu de l'animation en modélisant l'objet à animer avec un model physique.

Référence :

Lap[07] M. Laprade. « Traitement interactif de plis dans la simulation de tissus » Mémoire de maîtrise (M.Sc.). Université de Montréal, 2007.

[1] Frédéric Cazals and Marc Pouget. Topology driven algorithms for ridge extraction on meshes. Rapport de recherche 5526, INRIA, 2005

[2]S.BENAMEUR , M DJEDI , une methode d'integration efficace pour la simulation de tissu par un modele physique .Courrier du savoir ,volume 22 numero 1 , page 178-194,janvier 2017 .

[3] Sabrina Benameur and NourEddine Djedi. Intégration de la multi résolution dans un système masse-ressort : Application à l'animation de tissu. TAIMA'05, pages 495–500, 2005.

[4] Fernando Birra and Manuel Santos. Towards efficiency in cloth simulation. In International Conference on Articulated Motion and Deformable Objects, pages 144–155. Springer, 2008.

[5] Jan Bender, Daniel Weber, and Raphael Diziol. Fast and stable cloth simulation based on multi-resolution shape matching. Computers & Graphics, 37(8) :945–954, 2013.

[6] Jan Bender and Crispin Deul. Efficient cloth simulation using an adaptive finite element method. In VRIPHYS, pages 21–30, 2012.

[7] Kwang-Jin Choi and Hyeong-Seok Ko. Research problems in clothing simulation. Computer-aided design, 37(6) :585–592, 2005.

[8] Frederic Cordier and Nadia Magnenat-Thalmann. Real-time animation of dressed virtual humans. In Computer Graphics Forum, volume 21, pages 327– 335. Wiley Online Library, 2002.

[9] Sébastien Delest. Segmentation de maillages 3D à l'aide de méthodes basées sur la ligne de partage des eaux. PhD thesis, Université François RabelaisTours, 2007.

[10] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A fast finite element solution for cloth modelling. In Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on, pages 244–251. IEEE, 2003.

[11] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In ACM SIGGRAPH 2005 Courses, page 1. ACM, 2005.

[12] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A fast finite element solution for cloth modelling. In Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on, pages 244–251. IEEE, 2003.

[13] Jerry Weil. The synthesis of cloth objects. ACM Siggraph Computer Graphics, 20(4) :49–54, 1986.

[14] BK Hinds and J McCartney. Interactive garment design. The Visual Computer, 6(2) :53–61, 1990.

[15] J-M Nourrit, Eric Desjardin, and C Secroun. Modélisation de textiles à base de mailles. Revue internationale de CFAO et d'informatique graphique, 12(4) :373–385, 1997.

[16]James F. Blinn. Simulation of wrinkled surfaces. In SIGGRAPH '78 : Proceedings of the 5th annual conference on Computer graphics and interactive techniques, pages 286–292, New York, NY, USA, 1978. ACM Press.

[17] Pascal Volino and Nadia Magnenat-Thalmann. Fast geometrical wrinkles on animated surfaces. In Seventh International Conference in Central Europe on

[18] Computer Graphics and Visualization (Winter School on Computer Graphics), February 1999.

[19] Peirce (F. T.). – The geometry of cloth structure. The journal of the textile institute, vol. 46, n◦5, 1930, pages : 45–96

[20] Kawabata (S.), Niwa (M.) et Kwai (H.). – The finite deformation theory of plain weave part i : the biaxial deformation theory. The journal of the textile institute, no2, fevrier 1973, page : 62.

[21]Grosberg (P.) et Kedia (S.). – The mechanical properties of woven fabrics part i : the initial load extension modulus of woven fabrics. Textile Research Journal, vol. 36, n ◦1, janvier 1966, pages : 71–79.

[22]Abbot (G. M.), Grosberg (P.) et Leaf (G. A. V.). – The elastic resistance to bending of plain-woven fabrics. The journal of the textile institute, vol. 64, n◦6, juin 1973, page : 346.

[23] Olofson (B.). – A theory of elasto-plastic buckling. The journal of the textile institute, vol. 58, n◦6, juin 1967, page : 221.

[24] Skelton (J.). – Fundamentals of fabric shear. Textile Research Journal, vol. 46, n◦12, 1976, pages : 862–869

[25] R. Revire, F. Zara, and T. Gautier. Efficient and Easy Parallel Implementation of Large Numerical Simulations. In ParSim 2003 (Special Session of EuroPVM/MPI 2003), Venice, Italy, September 2003.

[26] P.-E. Bernard. Parallélisation et multiprogrammation pour une application irrégulière de dynamique moléculaire opérationnelle. Mathématiques appliquées, Institut National Polytechnique de Grenoble, France, Octobre 1997.

[27] M.-J. Flynn. Somme Computer Organizations and their effectiveness. IEEE Transactions on Computers, C-21(9) :948–960, 1972.

[28] G. Authié, J-M. Garcia, A. Ferreira, J-L. Roch, G. Villard, J. Roman, C. Roucairol, and B. Virot. Parallélisme et applications irrégulières. Hermès, 1995.

[29] NOUR EL-HOUDA BENALIA, Exploitation des potentialités des cartes graphiques à l'optimisation des algorithmes évolutionnaires, thèse de Doctorat, Laboratoire LESIA, Université de Biskra, Algérie, 2015

[30] S. Ahuja, N. Carriero, D. Gelertner, and V. Krishnaswamy. Matching language and hardware for parallel computation in the linda machine. IEEE Transactions on Computers, 1988(8) :921–929, August 37.

[31] T. Gautier and J.-L. Roch. Irregular algorithms and on-line scheduling. In INRIA, editor, Proc. of Stratagem'96, Symposium on parallel computing for solving large scale and irregular applications, pages 17–37, Sophia-Antipolis, France, jul 1996.

[32] Ramakrishnan Mukundan, Advanced methods in computer graphics, Springer, 2012.