VOT 74224

THE DEVELOPMENT OF DEFORMABLE BODIES COLLISION RESPONSE ALGORITHM FOR INTERACTIVE VIRTUAL ENVIRONMENT

(PEMBANGUNAN TINDAK BALAS PELANGGARAN OBJEK BOLEH CANGGA UNTUK PERSEKITARAN MAYA INTERAKTIF)

NORHAIDA MOHD SUAIB
ABDULLAH BADE
DAUT DAMAN
MOHD SHAHRIZAL SUNAR

PUSAT PENGURUSAN PENYELIDIKAN
UNIVERSITI TEKNOLOGI MALAYSIA

2007

ACKNOWLEDGEMENT

The researchers would like to express sincere gratitude to all parties and individuals involved, whether directly and indirectly in making this project a success, especially to the Ministry of Science, Technology and Innovation for providing funding for this research, Universiti Teknologi Malaysia (UTM) particularly the Research Management Centre, Faculty of Computer Science and Information System, fellow researchers and colleagues at the Department of Computer Graphics & Multimedia and all students involved. We are truly grateful for your support throughout this research.

Thank you.

ABSTRAK

Kaedah percanggahan berasaskan fizik lazimnya menghadapi masalah kerana memerlukan kos pemprosesan yang tinggi menyebabkan kaedah tersebut tidak sesuai untuk digunakan secara praktikal di dalam aplikasi interaktif, walaupun jika percanggahan hanya berlaku pada kawasan kecik objek boleh canggah. Tesis ini mencadangkan kaedah percanggahan berasaskan pemilihan dinamik untuk objek yang mengalami percanggahan pada kawasan kecil. Ia dilakukan untuk memastikan interaktiviti dengan objek berisipadu yang mempunyai bilangan geometri yang banyak dengan mengurangkan kawasan yang akan diproses untuk percanggahan. Kaedah ini adalah satu bentuk algoritma pengoptimum yang akan memilih kawasan yang akan diproses untuk percanggahan berdasarkan keadaan kestabilan kawasan tersebut. Dengan menganggap tiada tenaga lain yang bertindak ke atas objek boleh canggah selain daripada tenaga menumpu, algoritma pengoptimum ini berjaya mengurangkan pengiraan percanggahan untuk sistem percanggahan berasaskan fizik. Kaedah ini sesuai digunakan untuk aplikasi masa nyata seperti pembedahan maya.

ABSTRACT

Physical based deformation method usually suffers from high computation cost which does not favors practical interactive applications, even if the deformation only occurs in a small area of the deformable object. This thesis proposed a dynamic selection based method for small area deformation to maintain interactivity with high geometric complexity of volumetric mesh by reducing areas for deformation processing. It is an optimization algorithm that selects small areas for deformation processing based on equilibrium state. Assuming no external forces other than concentrated loads, the optimization algorithm succeeded to reduce deformation computation for physical based deformation systems. The method is suitable for real time application like virtual surgery.

TABLE OF CONTENTS

CHAPTER TWO

LITERATURE REVIEW

CHAPTER II LITERATURE REVIEW

CHAPTER THREE
METHODOLOGY

CHAPTER III METHODOLOGY

CHAPTER FOUR
IMPLEMENTATION

CHAPTER IV IMPLEMENTATION

CHAPTER FIVE
ANALYSIS

CHAPTER V ANALYSIS

CHAPTER SIX
CONCLUSIONS

CHAPTER VI CONCLUSIONS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER I

INTRODUCTION

This chapter describes the context of the work, presents the research statement, and provides an overview of the report.

1.1 Introduction

Real time deformation is an important aspect of interactive computer graphics especially in computer animation and medical application. It has been extensively studied since the introduction of global deformation by Barr in 1984 (Barr. 1984). In general, the studies of deformable modeling focuses on diversity of deformable object characterization, accurate material representation and gaining high simulation performance. With the increasing power of 3D hardware, the deformable modeling field has gain a new research direction. The input data for real time application can now contain thousands of polygons ensuring more accurate shape representation than ever before. Although the polygon rendering capacity increased, deforming large number of polygons remains a problem in real time applications. This is due to the high processing resources required by the deformable modeling method. In order to balance the available resources between processing and rendering, recent studies focuses more on deforming object with large number of polygons interactively.

Achieving interactive deformation is a crucial part in computer animation and medical applications. Deformable modeling can assist artist in modeling 3D content for computer animation by enabling higher degree of controls for modeling tools. These tools reduce artist workload and provide better results in less time compared to traditional method without deformation tools. Another form of deformation modeling, physical based deformation, used by computer animation to provide a method to simulate the behaviour of real world materials. The results are visually convincing in terms of realistic depiction of the real world compared to traditional animation method. With physical based animation, artists are no longer required to manually key framed the animation as the task has been shifted to the physical based animation system.

Deformation modeling also has found its way to medical field. It is used mainly to simulate the behavior of soft tissues of the human body. One example of medical application is virtual surgery which allows trainee surgeons to feel and see exactly what they would if they were operating on real patient. This may help improve surgical skills of the surgeons as it would with pilot trained in flight simulator. The use of virtual objects reduces the cost of obtaining real material for surgical training and reduces the offensive nature of using real dead bodies for training. With virtual surgery application, surgeons can plan ahead the surgical procedures and perform surgical test without the risk of failure. However, the complex nature of the human tissue and the demanding accuracy required by medical application makes it a very challenging domain.

The field of deformable object modeling has seen many improvements throughout the years. This will be discussed in Section 1.2 (background and previous works that are related to this research). This is followed by the problem statement in Section 1.3, Section 1.4 lists the research objectives, Section 1.5 describes the domain and scopes and Section 1.6 introduces dynamic selection based method in brief. Results and findings are given in Section 1.7. The final section, Section 1.8, gives a summary of each of the chapters in this research.

1.2 Background

Computer graphics modeling had only been for rigid objects until Barr introduced global deformation technique, more than two decades ago (Barr. 1984). The idea behind this method is to apply another transformation to existing transformation before transformation is applied to the objects. In order to allow more deformation control over the objects, Sederberg introduced free form deformation (Sederberg et al. 1986). The method models non solid object behaviour by changing the object according to the changes experience by enclosing lattices. Both methods have been used extensively in 3D modelling tools and CAD tools. However, both deformation methods lack one crucial feature, and that is physical behaviour.

In order to allow physical behaviour to the deformable object, Terzopoulos proposed an elastic physical based deformation method in 1987 for use in pre-computed computer generated animations (Terzopoulos et al. 1987). Later, he introduces inelastic physical behaviour such as viscoelasticity, plastic and fracture (Terzopoulos and Fleischer, 1988). Then in 1989, he presents a method to model the behaviour of fluid like molten objects (Terzopoulos et al. 1989). Generally, Terzopoulos and his colleagues proposed methods that are based on simplification of elasticity theory to model various physical behaviours for use in pre-computed computer generated animations.

The behaviour of deforming objects is the topic of continuum mechanics, a branch of mathematics that tries to capture physical phenomena of continuous media in precise mathematical formulations. One branch of continuum mechanics, nonlinear elasticity, provides the mathematical description of how objects deform. Finite element method discretize infinite dimensional problem into systems of equations with a finite number of variables, to accurately describe physical based

deformation behaviour. However, due to the nature of the system and the complexity of the method, the method cannot be applied directly to real time animation systems.

Thus, Bro-Nielsen proposed a fast finite element method for use in virtual surgery environment (Neilson and Cotin, 1996). He uses condensation techniques to reduce the complexity of the system equations and thereby achieve a considerable speed-up compared to the volumetric models in (Cotin et al. 1996). The effect of using condensation techniques is low generality of the simulations, i.e. no rapid displacement, no great displacement. Based on the principle of superposition, Cotin proposed a higher generality deformation system (Cotin et al. 1999). Although the method experienced high frame rate, it was implemented on low resolution mesh. Furthermore, pre computation method used does not permit topological changes to the deformable objects.

By using high resolution mesh, the deformation behaviour can be modeled in higher accuracy. The problem with high resolution mesh is that it costs more computational resources. To reduce computational resources, several researchers have opted to use multi-resolution method in the mesh domain. Debunne uses automatic space and time adaptive object representation level of detail technique to allow local refinement or simplification of the computation model based on local error measurement (Debunne et al. 2001). Krysl uses adaptive local finite element mesh refinement using wavelet theory to accelerate finite element deformation (Krysl et al. 2003). Although both methods produce acceptable frame rates on high resolution mesh, they still suffer from high computation required by finite element method.

Another way to reduce deformation processing time is by using simpler physics method, such as mass-spring systems. Mass-spring systems describe the deformable object as nodes connected by springs. It is commonly used in cloth simulation (Breen et al. 1994), (Volino and Thalmann. 1997), (Bridson et al. 2003), (Baraff and Witkin, 1998), (Provot, 1995), (Choi and Ko, 2002) and (Baraff et al.

2003). To model deformation for volumetric objects, the deformable object must first be discretized as one would with finite element method. The prominent problem of mass-spring systems is numerical instability under large time step (Baraff and Witkin, 1998). For large number of mass-spring nodes, the simulation system quickly converges error and became unstable. One solution to the stability problem is Verlet integrator, which capable of maintaining stability even for large number of nodes (Jacobsen 2003). Unfortunately, for deformable object with extremely large number of nodes, mass spring system is still too slow to be used for interactive systems.

By using simple mathematical approach for deformation processing, Gibson proposed a fast deformation method for extremely large number of nodes (Gibson, 1997). The method known as ChainMail, perform deformation based on nodes distance constraint. However, due to the used of simple distance constraints for deformation instead of continuum-based physics, the resulting behaviors are not physically convincing. Plus, it is hard to define real world materials. Nevertheless, ChainMail contributed new approach in the field of deformable object modeling by introducing force propagation method. Force propagation method works like sound wave effect in the sense that areas near contact are first displaced and displacements are propagated throughout the object.

Since the introduction of ChainMail, there are many improvements on the force propagation method made by various researchers. To introduce physical-based deformation on ChainMail, Dusyak and Zhang presented an improvement method to the ChainMail algorithm by combining the ChainMail algorithm with a modified mass-spring system (Dusyak and Zhang, 2004). The result is a high speed simulation of physical-based deformation but the author does not describe multiple contacts handling, which apparently seems to be the problem. Another improvement to the ChainMail is by the work from Grimm et al.; in which deformation behaviors for surface deformations were improved by using dynamic length spring constraint based on distance to the source of collision (Grimm et al. 2004). Like Dusyak and Zhang, the author did not describe how to handle multiple contacts at all. Also, the

algorithm was tested for surface deformation only. Choi et al. used static selection of neighbouring nodes to handle deformation (Choi et al. 2002, Choi et al. 2003). One critical problem of the algorithm is that the deformation area cannot be scaled as needed. A special method is required to handle multiple contacts. Another improvement to the ChainMail algorithm is made by Park et al. who extended ChainMail algorithm by preserving original shape by keeping track of the direction vector from current node position to the original node position (Park et al. 2002).

This research tries to find solutions to the above mentioned problems noted by previous researchers.

1.3 Problem statement

The advent of hardware acceleration rendering support has made geometry a popular choice for real time and interactive applications rendering. With increasing complexity of 3D geometric data and growing demand for realistic deformation functionality, significant effort is being devoted to the design of robust, fast, and scalable algorithms for geometry deformation processing. The problem for volumetric object lies within the fact that it consists of internal structure that requires deformation processing. To achieve high degree of deformation accuracy, classical physics based on continuum mechanics (for computing construction stress) are used for deformation processing. Deformation processing usually consists of dynamics formulation integration throughout the deformable objects. Deformation processing usually involves up to millions of every internal volume elements for a complex geometry object, thus sacrificing interactivity. Managing large sets of data for high speed data transfer under limited available memory storage requires special attention to ensure interactivity.

For small deformation based on concentrated loads, the largest primitive elements' (vertices) displacements are on the area near applied concentrated loads. The further the elements distance from the contact point centre, the lesser force experienced by the elements. This is due to the damping forces conducted by every passing element during force propagation. Similar phenomena can be observed by softly touching a pillow. Notice that only small area near touched area are deformed. Traditional deformation methods perform deformation processing throughout the object even if the vertices did not experience noticeable deformation or if the vertices did not experience any deformation at all. Based on this observation, this research proposed a deformation method where deformation will be process on the areas that are most likely to experience noticeable deformation for non critical interactive applications. This way, the effect upon having high geometry deformable objects seems transparent for total application performance as the system only process deformation for a limited sets of elements.

This research addresses the deformation processing problem for small deformation situations. The hypothesis is stated as:

The cost of deformation processing can be reduced by only deforming small areas (or regions, nodes, vertices etc) that are most likely will undergo deformations.

1.4 Objectives

It is desirable to put computation resources where it will be most beneficial. To this effect, this research outlines the most critical objectives as follows:

1. To inquire into appropriate deformable objects representation.
2. To investigate, analyze and formulate an appropriate technique for collision response encompassing deformable objects adequate for interactive application.
3. To develop an algorithm of collision response for deformable body motion.
4. To design and develop a real-time simulation model based on objects representation and handling of real-time collision response.

1.5 Scopes

The scopes of this project are as follows:-

- Deformable volumetric objects are represented geometrically.
- Objects are manifold and do not experience topological changes such as cutting and fracture.
- Deformations are performed based on concentrated loads.
- Object deformations are fully elastic. Deformed objects should return to its original state after removal of applied external forces.
- Interactions are described as single manipulator tool versus deformable object vertex.
- No volume preservation.
- No frictional forces will be considered.

1.6 Dynamic selection-based method

This research proposed an algorithm known as dynamic selection-based method. In short, the algorithm reduces the cost of deformation processing by dynamically select small areas for deformation. It is used with mass-spring system as the main deformation system.

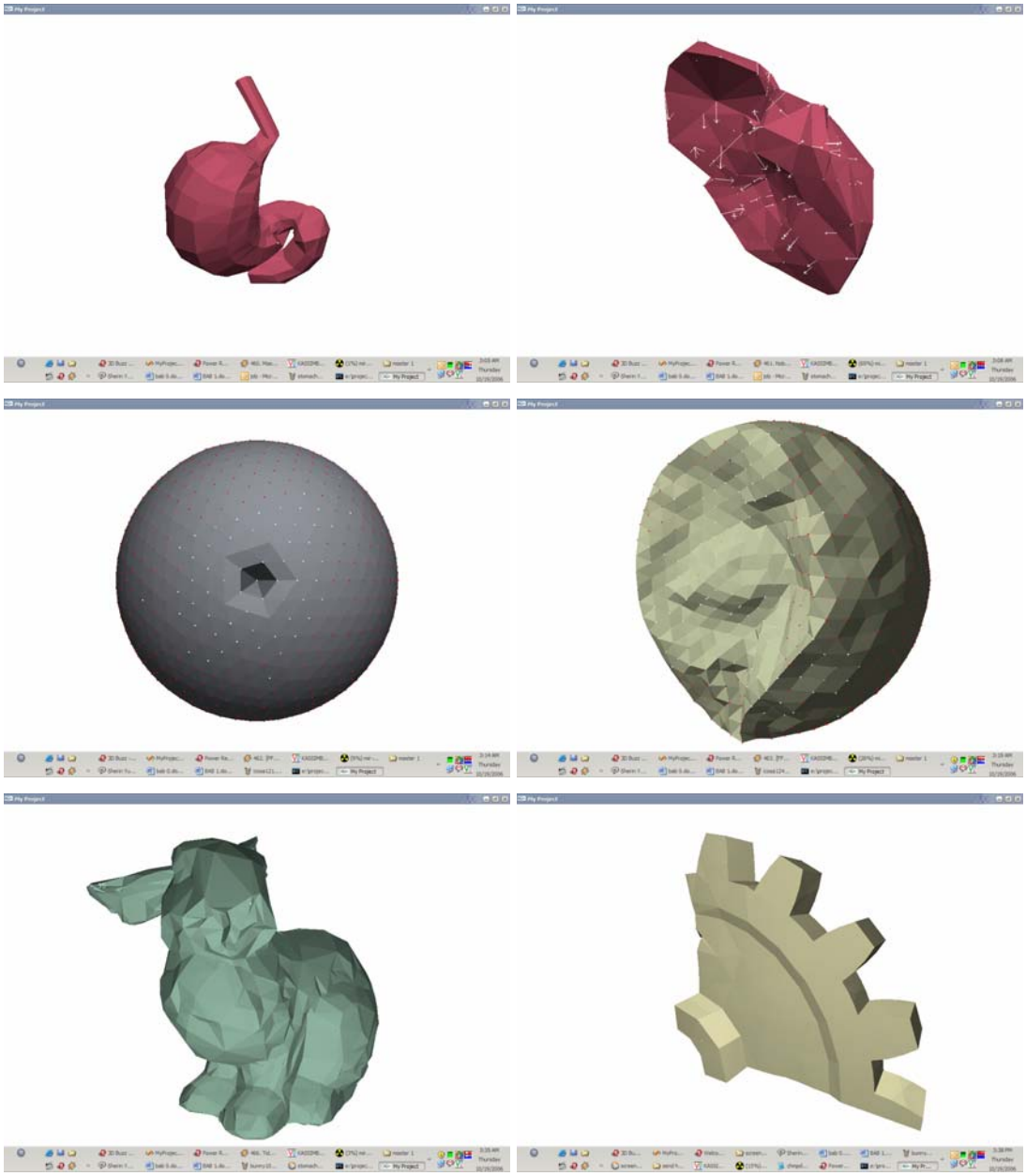


Figure 1.1 Example of dynamic selection-based method.

Dynamic selection based method is highly inspired by force propagation theory. Known as ChainMail, it was first introduced for deformable modeling by Sarah F. Gibson (Gibson. 1997). Based on the reviews, ChainMail doesn't seem to include any physical based justification in its deformation. In depth discussion of this topic are available in Chapter 4.

1.7 Results

The results from this research are summarized as follows:

Object representation: Deformable objects are represented using mass-spring model.

Reduced area for deformation: For every frame, deformable object is evaluated for deformation. The result from the evaluation is a small area of deformable object that will be selected for deformation. Similar in nature to ChainMail (Gibson 1997), this will reduce required deformation processing time as only small areas are actually deformed per frame.

Dynamically enlarge or shrink deformation area: Unlike previous deformation method inspired by force propagation, dynamic selection based method can dynamically enlarge or shrink deformation areas. Previous works usually either resort to static range of areas (Choi et al. 2003) or propagate over the deformable object infinitely (Dusyak and Zhang. 2004). Other methods that can dynamically enlarge or shrink deformation area do not have physical-based justifications in their deformations.

Scalable for either performance or accuracy: In order to tackle broad range of applications, dynamic selection-based method allows the user to tinker with the parameter settings. These settings enable the application to be tuned either for high accuracy or high performance. Chapter 5 provides testing result of different parameter settings.

1.8 Summary of chapters

This section presents a brief overview of the content of this report.

Chapter II: An overview of previous works on deformation method, real-time performance strategy and force propagation-based method.

Chapter III: Implementation planning was outlined here. Acceleration strategies were described along with its justifications. Both hardware and software specification requirements are discussed here.

Chapter IV: Detail discussions on implementation starting from building the data, algorithm loops and algorithm.

Chapter V: Results and benchmarks of the research. It provides analytical performance results, discussion of various issues regarding the performance and quality of deformation behaviors of the proposed algorithm.

Chapter VI: Summary of the report. It reflects on the objectives achievements, contributions and future work.

CHAPTER II

LITERATURE REVIEW

2.1 Overview

In this chapter, elementary theories and techniques that are relevant in volumetric object deformation are discussed. Presented next are literatures for both non-physical-based modeling and physical-based modeling of deformable objects. Then, the discussion will cover previous work on real-time acceleration techniques.

2.2 Introduction

In engineering mechanics, deformation is a change in shape due to an applied force. This can be the result of tensile, compressive, shear, bending or torsion forces etc. Deformable materials can be distinguished by three states of matter; solid, liquid and gas. Some examples of deformable objects are sponge, water and smoke. Two major distinctions on deformable objects modeling are of animation applications and of editing applications. Animation applications usually deal with methods which animates the nature of deformation as a function of time. For example, cloth tools in

most 3D animation package where cloth deformations are simulated for scene environment throughout animation time. Often considered as physical-based deformations, the cloth tools tries to find the equilibrium state for the cloth based on interacting forces. For deformable objects editing, there will be a mechanism or method which facilitates the deformation of object deformations. For example, free form deformation tools available in most 3D animation packages where the objects are deformed to satisfy constraint that are manipulated by user. Often considered as non-physical based deformation, the free form deformation method will displace primitives (with specific constraints) until it reached a new position which satisfies the constraints.

Physical-based deformation for solid objects (non-liquid and non-gaseous matter), based on theory of elasticity, can be either be elastic or inelastic (plastic). Elastic objects are objects that return to their original states after removal of applied forces. Contrary to elastic objects, inelastic objects are objects that do not return to their original state after applied forces have been removed due to atomic plane dislocation in the real materials. Technically, all objects should be considered inelastic, due to the fact that every object can experience atomic plane dislocation. But due to performance reason, most objects that behave elastically during the simulation can be considered as elastic objects. Fundamental measurements of deformed object are by its dimension; length for one dimensional objects, surface area for two dimensional objects and volume/bulk for three dimensional objects. For three dimensional cases, deformable objects are represented with volumes that have both surface structure and internal structure. There are three types of forces; concentrated loads, distributed loads over the body and distributed loads over the surface of on object. Concentrated loads are forces applied at discrete points. Example of concentrated loads is force exerted when a pencil tip is pressed onto a pillow. The second type of force is loads distributed over the body. Such is gravitational force. The final type of force is loads distributed over the surfaces of the object. Air pressure and water pressure is good example of loads distributed over the surfaces.

Volumetric deformable objects can be represented by geometric mesh, iso-surface, voxels, points etc. Geometric mesh-based rendering can be accelerated efficiently by 3D hardware compared to other methods of rendering.

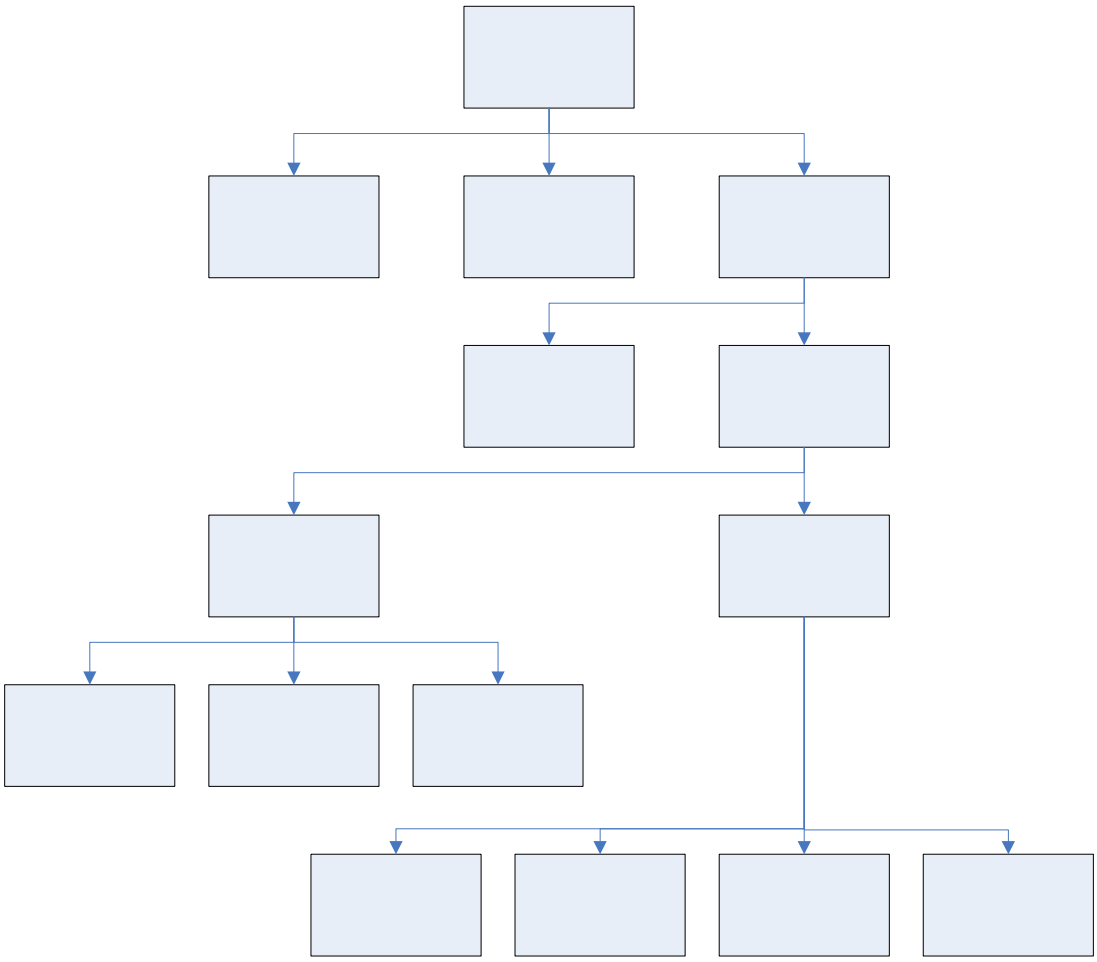


Figure 2.1 Taxonomy of deformable objects for this research.

2.3 Deformable objects modeling

Deformable objects application varies in input data, degree of required accuracy, user interaction and material flexibility. Usually, to suit for specific

application requirements, the application traded off less important simulation features to provide the desired features more computational power.

2.3.1 Input data

Since current 3D hardware has matured enough to support geometric rendering, the focus of this research will be on techniques to acquire geometric data. Geometric data can be acquired by designing, 3D scanner, diagnostic radiology or from mathematical models. Modeling tools such as Autodesk® 3ds Max®, Autodesk® Maya®, NewTek Lightwave 3D®, Robert McNeel & Associates Rhinoceros® NURBS modeling allow designers to create, edit and analyze vertices, lines, curves, planes, surfaces and solids to produce the desired objects. These data can be saved as geometry, mathematical parameters or boundary elements (constructive solid geometry, boundary representations). To acquire data from real world object, one can use 3D scanners available from Cyberware, Northern Digital Inc. and Cognitens Ltd., to name a few. Data acquired using laser scanning, electromagnetic resonance or multiple sets of 2D images reconstruction, are highly accurate compared to artist impression of the objects. Diagnostic radiology enables one to get information of a particular object including both surface and its underlying structure. By analyzing reflection, penetration or emitted energy of transmitted light wave (x-ray), electromagnetic wave, sound wave (ultrasound) or nuclear energy, underlying structure can be constructed without the need to cut the physical objects. These methods of data acquisition are very useful in medical applications as no significant harm done to the patient to get the underlying structure image. Some example of x-ray imaging equipment is computer tomography scan (CT scan), athrography and mammography. Hysterosonography use ultrasound waves to show structures in the human body. The sound waves reflect off internal organs and other anatomic structures to create images. Magnetic resonance imaging (MRI) is a method of producing extremely detailed pictures of body tissues and organs using electromagnetic energy. Electromagnetic energy that is released when exposing a

patient to radiofrequency waves in a strong magnetic field is measured and analyzed by a computer, which forms two- or three-dimensional images that may be viewed on a TV monitor. Nuclear medicine is a subspecialty within the field of radiology. It comprises diagnostic examinations that result in images of body anatomy and function. The images are developed based on the detection of energy emitted from a radioactive substance given to the patient, either intravenously or orally. Voxelman register together data from various sources (CT, MRI, X-ray) to create visualization of human skull anatomy. Data acquired using diagnostic radiology has to be reconstructed as geometric data before performing geometric deformation. Geometric data can also be generated by mathematical models using fractal, implicit method or by spline-based technique.

Figure 2.2 Catheter Angiography : X-ray equipment is mounted on a C-shaped gantry with the x-ray tube itself beneath the table on which the patient lies. Above the patient is an image intensifier that receives the x-ray signals, amplifies them, and sends them to a TV monitor.

Figure 2.3 CTA scan equipment.

Figure 2.4 MRI equipment.

Figure 2.5 Ultrasound (sonography) equipment.

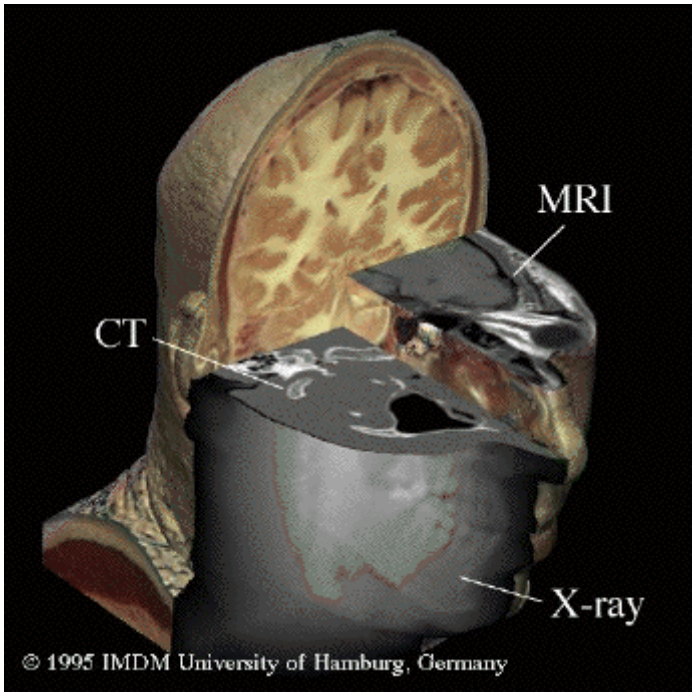


Figure 2.6 Voxelman showing registration of several data sources.

To perform deformation, the object need to have some kind of deformation weight or coefficient. These material properties can either be manually defined or by analytical procedure. Kawabata Evaluation System (Kawabata, 1980) (House and

Breen, 2000) is a standard set of fabric measuring equipment that can measure the bending, shearing and tensile properties of cloth. The equipment measures force or moment that is required to deform a fabric sample of standard size and shape, and produces plots of force or moment as a function of measured geometric deformation. From physics literature (Yong and Nagappan, 2003) (Cutnell and Johnson, 1995), material properties can be divided into five phases; limit of proportionality, elastic limit, yield point, breaking stress and breaks (as illustrated in Figure 2.7).

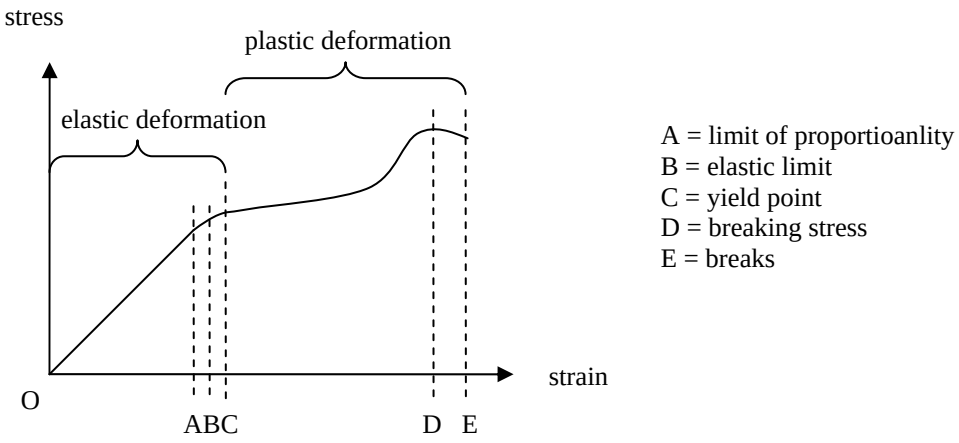


Figure 2.7 Stress strain graph indicates elastic and plastic (inelastic) deformations.

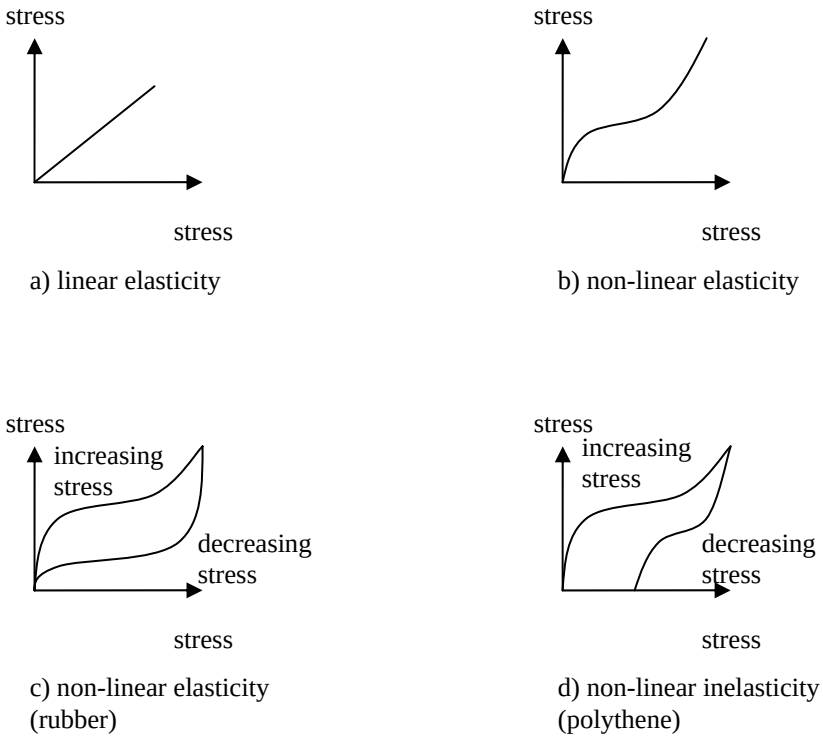


Figure 2.8 Stress strain graph showing multiple type of relation for deformations.

Table 2.1 Values for the Young's modulus of multiple solid materials. (Cutnell and Johnson, 1995)

Material	Young's Modulus Y (N/m^2)
Aluminium	6.9×10^{10}
Bone Compression	9.4×10^{9}
Bone Tension	1.6×10^{10}
Brass	9.0×10^{10}
Brick	1.4×10^{10}
Copper	1.1×10^{11}
Mohair	2.9×10^{9}
Nylon	3.7×10^{9}
Pyrex glass	6.2×10^{10}
Steel	2.0×10^{11}
Teflon	3.7×10^{8}
Tungsten	3.6×10^{11}

2.3.2 Data complexity

Usually, in computer graphics term, complex polygonal object refers to object with large number of polygons. Since computers have limited total triangle fill rate per second, reducing the poly count is always a better choice, as long as the object maintains its original looks. For deformable object modeling, this is not usually the case. Object with low poly counts can still slows the system down. This is due to the cost of deformation calculation for every primitive element of the objects, which is expensive. Different algorithm varies in its computation cost, but at the end, the bottleneck is usually the CPU processing power, not the GPU processing power. Thus, for deformable object, it is recommended to reduce the geometry of deformable object representation as long as it can undergo deformation to the required extent with acceptable results.

Another thing to consider is mathematical complexity of the deformation process. To achieve accurate results, one has to resort to use accurate computation techniques usually originated from mechanical dynamics in physics. These accurate techniques are suitable for highly risky simulation such as virtual surgery. But, a large number of computation tends to prevent the simulation to be performed in interactive manner. Mathematical complexity can be reduced by using simpler ordinary differential equation solver (Teschner et al. 2004), assuming fixed state (Matyka, 2003), pre-compute complex computation (D. L. James and Fatahalian, 2003) etc..

Different deformation techniques vary in its object representation memory consumption. Complex object requires enough memory to store its large geometric structure including internal geometric structure and its material properties. Additional memories are required to store its auxiliary data, such as pre-computed function, coherence cache and temporary variables. This large memory requirement poses challenges on memory storage and data access rate for real-time physical-based simulation and interaction.

2.3.3 Accuracy

Required accuracy varies between different types of applications. Depending on the application requirements and involved risk, end results can either be interactively manipulated, physically realistic or physically plausible. Deformation accuracy can be divided into three; geometric accuracy, mathematical accuracy and physical accuracy. To achieve high geometric accuracy, virtual object must closely resemble its real world counterpart. High mathematical accuracy can be achieved by using accurate computation technique. For example, there are multiple ordinary differential equation solvers, and most of them will accumulate precision error over time. Choosing the most precise technique will delay noticeable inaccuracies and maintain the system stability for a longer period of time. Simulating deformation true to the atom level is very expensive as even the smallest visible object contains large number of atoms. Even when the technologies are able to simulate deformation by displacing atoms, the visual difference is hardly noticeable and less required for mainstream applications. Approximating physical accuracy can be done in various ways such as neglecting small deformation factor, assuming the material is a anisotropic or non-heterogenous material and assuming linear elasticity deformation.

2.3.4 Interactivity

In computer graphics modeling, it is essential to have deformation tools that are robust and fast enough to be used interactively. Simple deformation technique like global deformation (Barr, 1984) is very limited in its deformation ability compared to free form deformation (Sederberg and Parry, 1986). Unlike non-physical-based deformation, physical-based deformation does not really permit user interaction as the object responds to applied forces rather than constraint modifier. If

the interactions are known and limited, deformation can be pre-computed to enable user interaction.

It is desirable to have user control (to some degree) over the deformable objects deformation motion instead of giving the dynamics formulation a total control. This is especially true for computer animation and cartoon animation as it can give animated characters unique behaviors.

2.3.5 Flexibility

Deformation robustness and flexibility are the ability of the deformation technique to support heterogeneous tissues, topological changes and material parameters changes. Human tissues consist of multi-layered, varied stiffness materials. Terzopoulos and Waters apply dynamic mass-spring system to facial modeling by constructing a three layer mass-spring mesh of dermal, fatty and muscle layer (Terzopoulos and Waters, 1990). Under certain conditions such as when elasticity limit are over stress, physically-based deformable objects experienced topological changes to its primitives representation which not only the object never retain the initial shape, but can be either ductile, fractured, tore, brittle or cut. Reconstructing geometry topologies pose a problem for real-time deformation especially for complex geometry. Elasticity coefficient for deformable object sometimes changes under different simulated environment. For example, it is easier to deform a hot plastic than a colder one due to additional energy vibrating the atom plates. Other examples of changing material parameters phenomena are mechanical wear, melting and hardening.

2.4 Non-physical based modeling

3D designers require precise deformation tools which give them total deformation control. These tools usually come as purely geometric modification tools which do not include any physical justifications in its deformation process. The output relies on the skill of the designer and how much control the deformation technique provides. Three most popular non-physical based modeling techniques are discussed as follows; global deformation, parametric representation and free form deformation.

2.4.1 Global deformation

In 1984, Barr introduced global deformation technique by extending the classical linear transformation operation (Barr. 1984). The idea behind this method is to apply another transformation to existing transformation before it is applied to the object. The available deformations are tapering, twisting and bending. Given a function for the transformations:

$X = F_x(x), Y = F_y(y), Z = F_z(z)$,

where (x, y, z) are vertices in undeformed state, and
(X, Y, Z) is the deformed vertex.

Object is tapered by choosing a tapering axis and differentially scales the other two axis components, setting up a tapering function along tapering axis. For example of tapering an object along its z-axis, $X = rx, Y = ry, Z = z$, where $r = f(z)$ is the tapering function either linear or non-linear. To globally twist the object, use differential rotation just as tapering is a differential scaling. To twist an object through an angle θ about the z-axis, we apply

$$(X, Y, Z) = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta, z)$$

By varying the amount of rotation as a function of z, the object will become twisted. This is done by setting $\theta = f(z)$ where f(z) specifies the rate of twist per unit length along the z-axis. To bend an object along y-axis, the deformation transformation is given by

$$X = x$$

$$Y = \begin{cases} -\sin\theta(z - k^{-1}) + y_0 & y_{\min} \le y \le y_{\max} \\ -\sin\theta(z - k^{-1}) + y_0 + \cos\theta(y - y_{\min}) & y < y_{\min} \\ -\sin\theta(z - k^{-1}) + y_0 + \cos\theta(y - y_{\max}) & y > y_{\max} \end{cases}$$

$$Z = \begin{cases} \cos\theta(z - k^{-1}) + k^{-1} & y_{\min} \le y \le y_{\max} \\ \cos\theta(z - k^{-1}) + k^{-1} + \sin\theta(y - y_{\min}) & y < y_{\min} \\ \cos\theta(z - k^{-1}) + k^{-1} + \sin\theta(y - y_{\max}) & y > y_{\max} \end{cases}$$

where $y_{\min} \le y \le y_{\max}$ is the bending region, k^{-1} is the radius of curvature of the bend, the center of the bend is at $y = y_0$, the bending angle is $\theta = k(y' - y_0)$ and

$$y' = \begin{cases} y_{\min} & y \le y_{\min} \\ y & y_{\min} < y < y_{\max} \\ y_{\max} & y \ge y_{\max} \end{cases}$$

Baraff and Witkin (Baraff and Witkin, 1992), uses connected global deformation elements to create flexible object deformation systems.

Global deformation can be easily implemented into existing application since the deformation transformation and classical transformation are similar in nature. The main setback for this method is that the deformation is limited to the three previously mentioned types of deformation. Also, small deformation cannot be performed using global deformation as they always deform the objects as a whole.

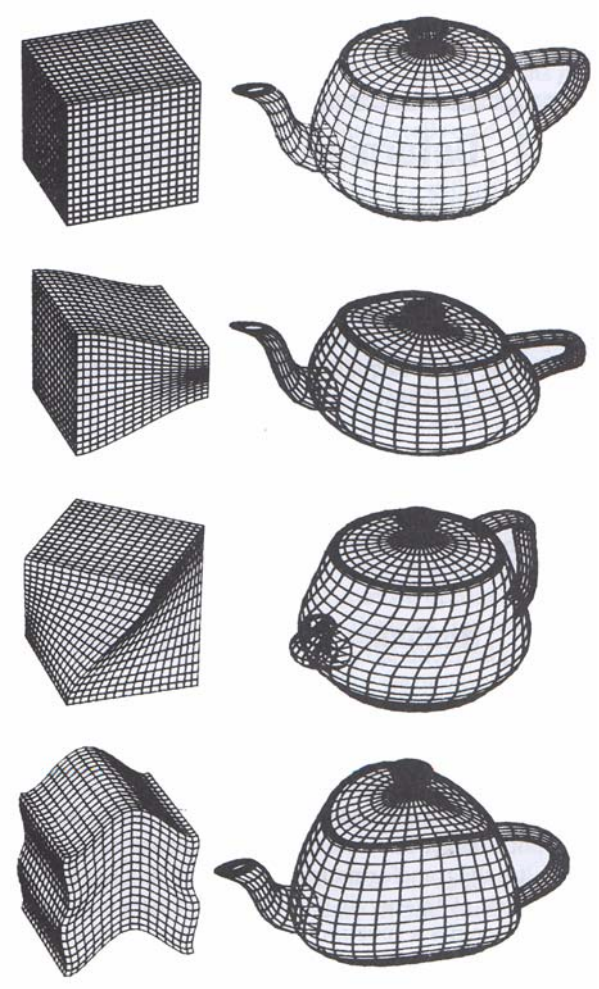


Figure 2.9 Structures deforming global deformation example. Top, original cube and Utah teapot followed by tapering, twisting and bending deformations. (Watt and Watt, 1992)

2.4.2 Parametric representation

By defining the object as parametric surfaces, users are given the ability to deform the surface by altering the functional description of the surface in the sense of displacing the control points. The first representational form or basis is due to Bézier, who was the originator of an early CAD system, UNISURF, used by Renault, a French car manufacturer.

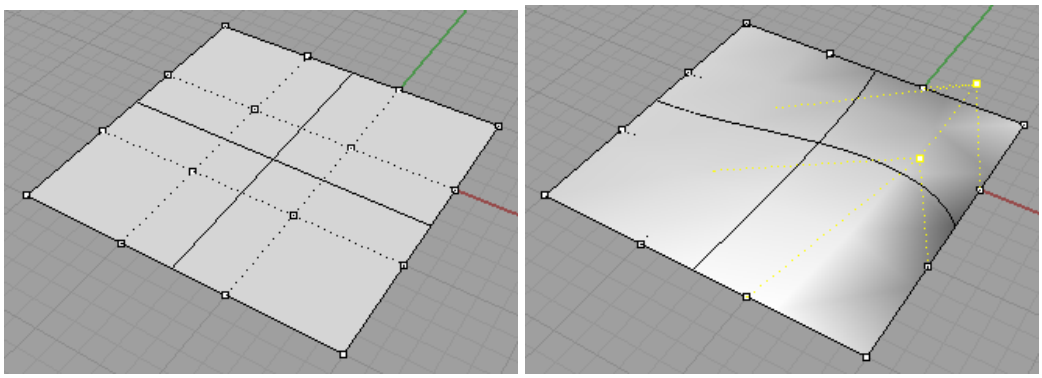


Figure 2.10 Examples of NURBS surface.

Given a set of n + 1 control points $P_0, P_1, ..., P_n$, the corresponding Bézier curve (or Bernstein-Bézier curve) is given by

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t),$$

where $B_{i,n}(t)$ is a Bernstein polynomial and $t \in [0,1]$. These functions are scaled or weighted by P_i, the network of control vertices, to form the surface patch. A cubic Bézier patch, an extension to the Bézier curve, is given by,

$$Q(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} B_i(u) B_j(v) .$$

Bézier patch always passes through the first and last control points and lies within the convex hull of the control points. Undesirable properties of Bézier patch are their numerical instability for large numbers of control points, and the fact that moving a single control point changes the global shape of the patch. The former is sometimes avoided by smoothly patching together low-order Bézier patch. The movements of the control points are constrained by continuity constraint between control points. These continuity constraints introduced two undesirable effects. First, undesirable plateau effect in the deformation is introduced if the deformation only displace the control points and not both control points and the continuity constraints. Second, it is impossible to achieve localize deformation since the continuity constraints may be propagated throughout the patch.

A generalization of the Bézier curve is the B-spline curve. As an improvement over the Bézier representation, B-spline are superior over the Bézier method within the context of deformation as B-Spline does not require continuity constraint and this gives the user the ability to perform localize deformation. Since the absence of continuity constraint, B-spline curve restricted the deformation by control points to only specific known region thus giving better control to the deformation made by the user.

2.4.3 Free form deformation

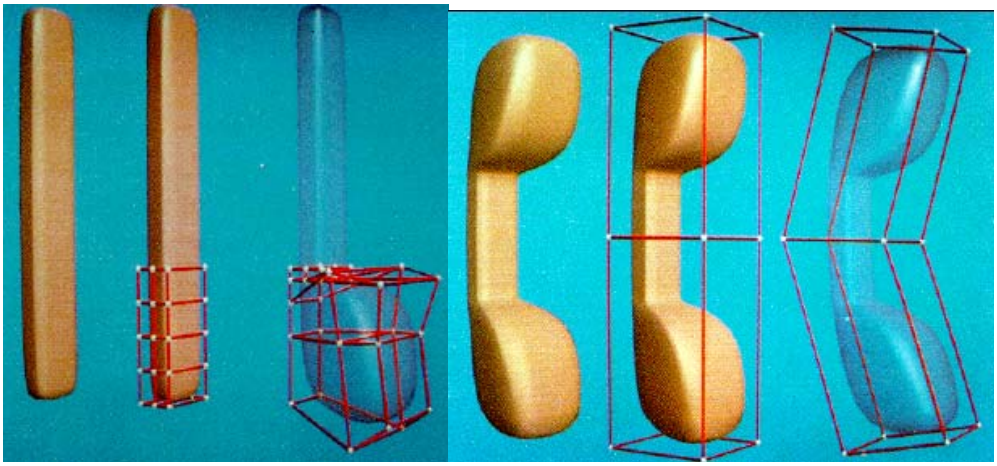


Figure 2.11 Right, local free form deformation. Left, global free form deformation. (Sederberg et al., 1986)

In 1986, Sederberg developed a technique that is more flexible than global deformation known as free form deformation (Sederberg et al. 1986). This technique defines a free-form deformation of space by specifying a trivariate Bézier solid, which acts on a parallelpiped region of space. Instead of deforming the object directly, this technique embeds the object in a defined space that is then deformed. The object is deformed according to the deformation that the embedding space

undergoes. The embedding space called FFD block, are actually hyperpatches connected together to form a piecewise Bézier volume.

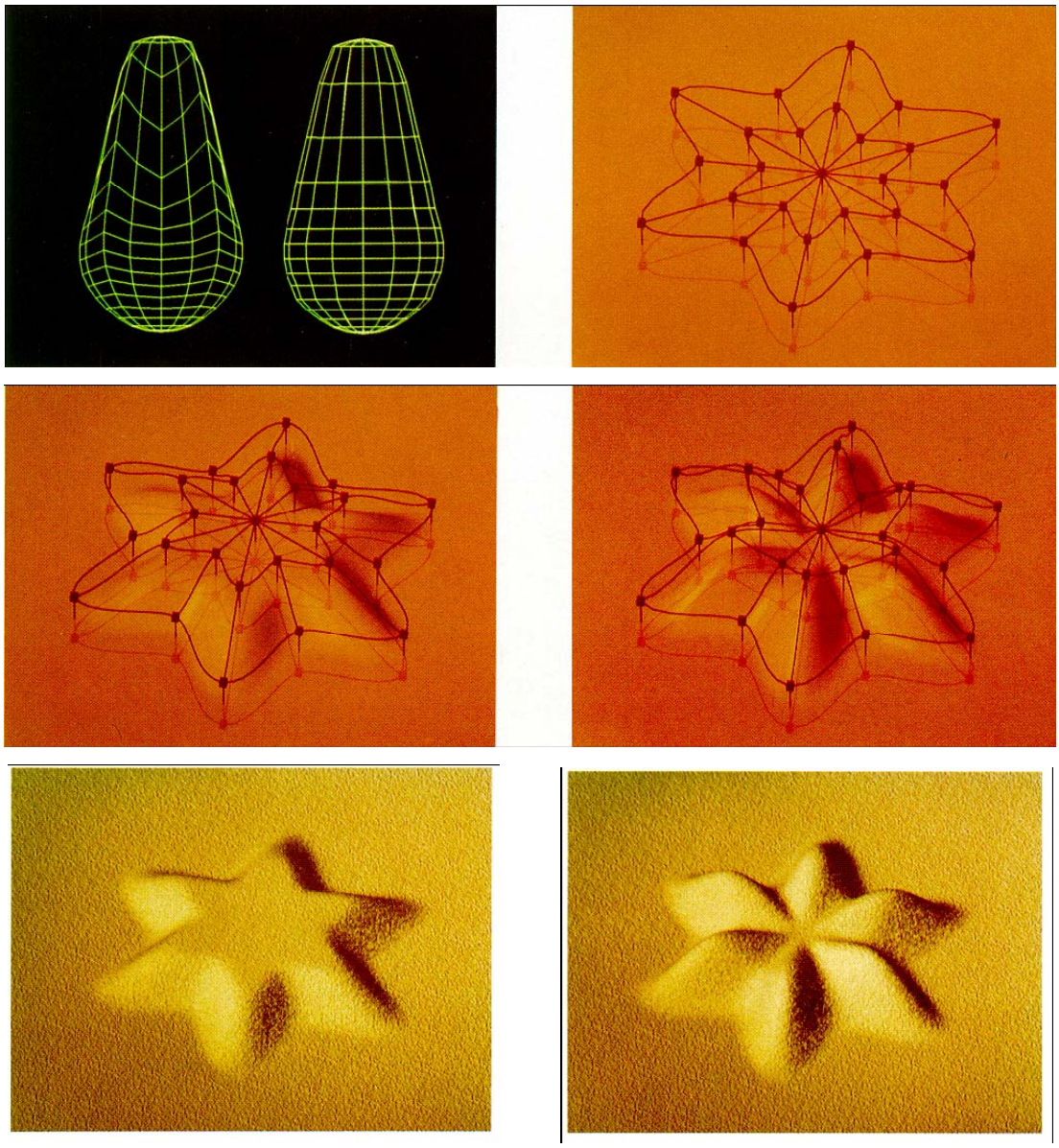


Figure 2.12 Extended free form deformations (Coquilart, 1990). Top left, a sphere deformed with a parallelepiped lattices. Top right, a sphere deformed with a cylindrical lattice. Middle left and right, deformed lattice and the deformed surface. Bottom left and right, resulting sand pie.

A single tricubic Bézier hyperpatch is defined as

$$q(u,v,w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} p_{ijk} B_i(u) B_j(v) B_k(w) ,$$

where $B_i(u)$, $B_j(v)$ and $B_k(w)$ are the Bernstein polynomials of degree 3. The undeformed FFD block consists of a rectangular lattices of control points arranged along three mutual perpendicular axes. The end result is a parallelepiped with lattices as control points attached. To deform an object using free form deformation method, we must first determine the positions of the vertices in the lattice space. Then deform the FFD block by displacing the control points from the undeform lattice positions. Finally, determine the deformed positions of the vertices by finding the relevant hyperpatch within which the vertex is located and convert to the local coordinate system of the hyperpatch.

This method can be used to apply localized deformation or to deform the whole object. Multiple FFD block can be defined in piecewise manner to perform deformation that is not possible to be done by using just a single FFD. For modeling complex deformation and specific small region of deformation, careful placement of FFD block by the user is required. However, the large number of FFD blocks would be inefficient to render.

Unlike free form deformation by Sederberg, Coquilart's extended free form deformation does not define any specific FFD lattice space (Coquillart, 1990). Coquilart states that parallelepiped shaped FFD block puts constraints onto the shape of the deformation and introduced nonparallelepiped lattices as the EFFD lattice space. To construct the EFFD block, users are required to weld several elementary blocks, which is the classic FFD blocks, together. As with FFD, to perform deformation, EFFD lattices need to be displaced. The deformation processing is very similar to that of previously discussed FFD except that unlike FFD, in EFFD, we cannot assume simple connection between the two adjacent spaces because lattice space of EFFD does not aligned with EFFD object space.

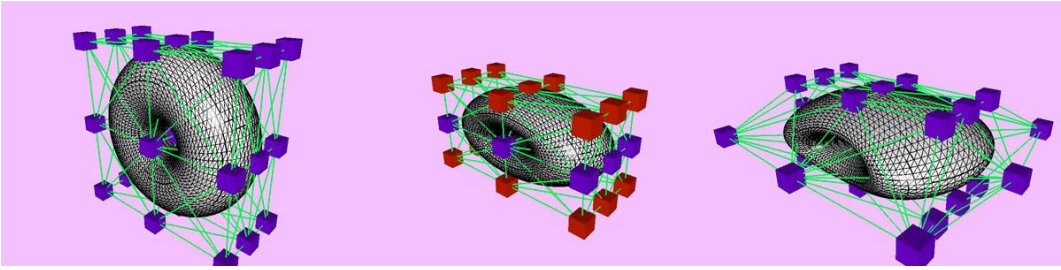


Figure 2.13 Hirota's volume preserving method. Letf, original shape. Center, after free form deformation is applied. Right, unconstrained lattices are displaced to preserve original volume (Hirota et al, 1999)

To preserve the total volume of solids undergoing free form deformation, Hirota uses discrete level of detail representations (Hirota et al., 1999). Given the boundary representation of a solid and user-specified deformation, the algorithm computes the new node positions of the deformation lattice, while minimizing the elastic energy subject to the volume-preserving criterion. During iterations, a non-linear optimizer computes the volume deviation and its derivatives based on a triangular approximation, which requires a finely tessellated mesh to achieve the desired accuracy. To reduce the computational cost, Hirota exploit the multi-level representations of the boundary. This technique also provides interactive response by progressively refining the solution. Furthermore, it is generally applicable to lattice-based free-form deformation and its variants. This method is capable of large deformation, efficiently. It gives designers and engineers real-time visual feedback and an intuitive physical feel of free-form solids, during geometric design and shape modification.

Exact shape and point placement is difficult to achieve with traditional free form deformations. This is due to the free form deformation interface which permits users to deform using only control points. Hsu et al. introduced a free form deformation method that allows user to control a free form deformation of an object by manipulating the object directly instead of using control points (Hsu et al., 1992). The method computes necessary alteration to the control points of the free form deformation spline using least square approach that will induce the point's placements.

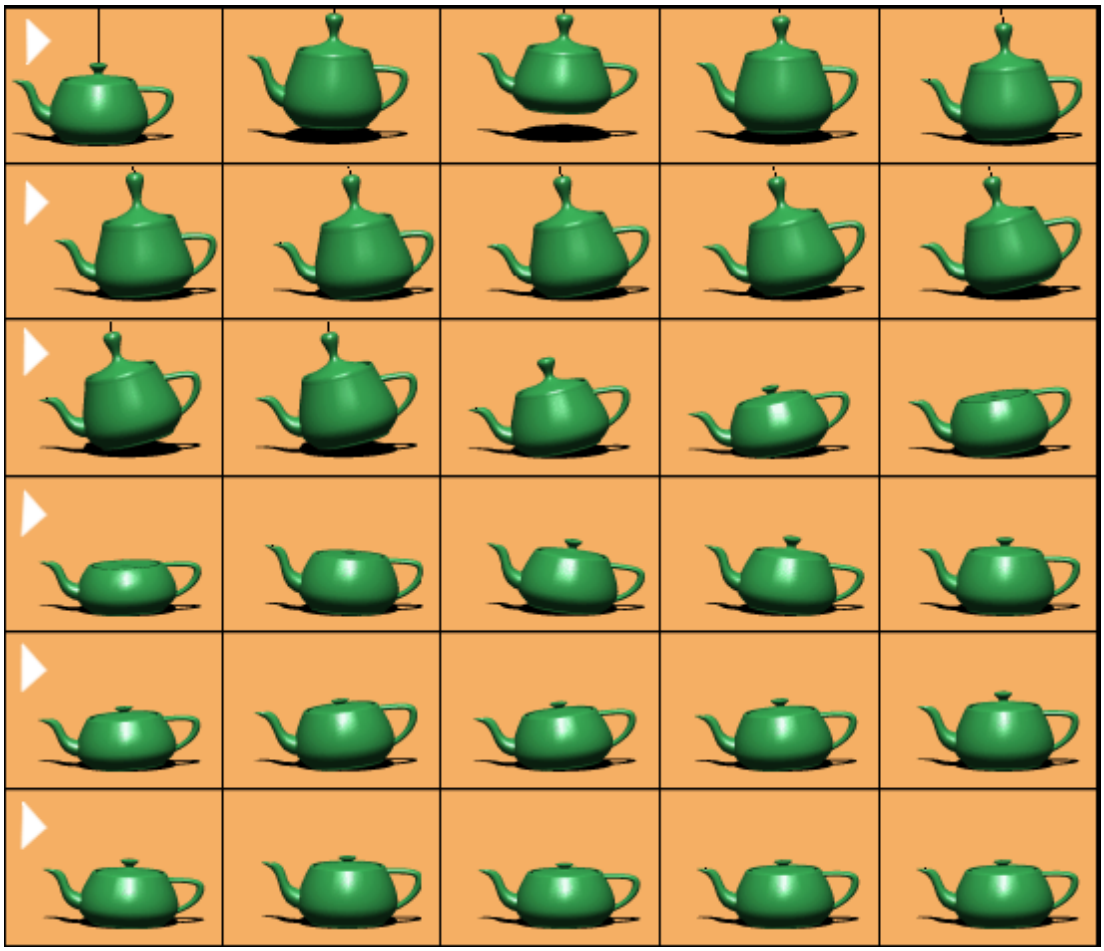


Figure 2.14 Deformable teapot is animated using dynamic global free form deformation. (Faloutsos et al., 1997)

Faloutsos et al. extends the use of free form deformation to a dynamic setting by coupling physical dynamics with free form deformation (Faloutsos et al., 1997). The method is based on parameterized hierarchical FFDs augmented with Lagrangian dynamics, provides an efficient way to animate and control the simulated characters. Objects are assigned mass distributions and elastic deformation properties, which allow them to translate, rotate, and deform according to internal and external forces. First, the dynamics generalization of conventional geometric free form deformation is formulated. The formulation employs deformation modes which are tailored by the user and are expressed in terms of free form deformations. Second, the formulation accommodates a hierarchy of dynamic free form

deformations that can be used to model local as well as global deformations. Third, the deformation modes can be active, thereby producing locomotion.

2.4.4 Pros and Cons

The main strength in parametric representation-based surface deformation is the ability to maintain object smoothness under any deformation complexity. Users are given total deformation control up to the control point complexity level. Due to this feature, parametric-based surface deformation is widely used in computer-aided design and model editing application.

Parametric-based surface deformation is not without its limitation. Since the object representations are defined as sets of parametric surfaces, the deformation detail level depends on the quantity of the control points. It is impossible to apply localized deformation in between control points. Re-meshing the parametric surfaces introduced aliasing that may not accurately reflect the intended deformation due to continuity of the constraints. It is difficult to represent object parametrically especially for objects possessing complicated topology. It is impossible to deform volumetric object while at the same time preserve its volume, since objects represented as parametric surfaces hold no volume information whatsoever. Eventually, simple deformation requires the adjustments of multiple control points or reconstructing the control points altogether which is very tedious.

Global deformation, FFD and EFFD provide higher level control than deformation based on parametric surfaces. While global deformation only provides limited sets of deformations, FFD allows user to manipulate its deformation constraints anyway they like. However, FFD also has its setbacks. The first two techniques are limited in permitting deformations as the techniques constraints the

deformation with its static deformation constraint but the latter provides a powerful tool as it gives the user the ability to construct the deformation constraint.

2.5 Physical-based modeling

Physical-based modeling uses physical principles to model realistic behavior of deformable models. This method uses more computational power than non-physical based method but the result is more convincing compared to the non-physical based method. Integration between physical principle and computer graphics for deformable object modeling was pioneered by Terzopoulos (Terzopoulos et al. 1987) (Terzopoulos et al. 1988) (Terzopoulos et al. 1989). Two most common and well known physical-based methods are finite element method and mass-spring method. On the other hand, two of the most recently proposed methods for physical-based modeling are known as mesh-free method and gas-based method. Here, basic physical-based method for deformable object will be discussed along with each method subsequent extension techniques.

2.5.1 Finite element method

The behavior of deforming objects is the topic of continuum mechanics, a branch of mathematics that tries to capture physical phenomena of continuous media in precise mathematical formulations. One branch of continuum mechanics, nonlinear elasticity, provides the mathematical description of how objects deform.

Continuum mechanics describes materials in terms of partial differential equations. The Finite Element Method (FEM) is a discretization method. It transforms a continuous, infinite-dimensional problem into systems of equations with a finite number of variables. For mechanical problems, the FEM discretizes the equations of motion; hence it delivers a system of ordinary differential equations, i.e., equations where time still has a role. There are two ways to deal with these systems: compute the evolution of the system, or try to find the final equilibrium solution directly. If the final state of the system is all that matters, a static method can be used. By assuming that velocity and acceleration are null, the system of differential equations is changed into a normal system of equations. For many mechanical problems, these equations can be stated in terms of finding minimum energy solutions. If transient effects do matter, then the evolution of the differential equations must be calculated using a time-integration method. Basically, the problems come from the simulation of soft tissue. Although simulating the full mechanical characteristics of soft tissue is not possible in an interactive setting, it is instructive to study exactly what kinds of characteristics are ignored in the simulations. It is not surprising when most implementation tends towards simplifications since the constraints of an interactive simulation do not allow for much sophistication.

To sum it up, the finite element method finds an approximation for a continuous function that satisfies an equilibrium condition which follows from the variation or weak formulation of the problem. The discretization of the problem consists of decomposing its domain into a mesh of carefully selected elements, joined at discrete nodes. The solution of the variational equation is expanded as a weighted sum of finite element basis or shape function on each element. Continuity across element boundaries is achieved by sharing discrete nodes and thus finite element weights. As a next step, the contributions of each element are assembled into a global system of equations which then can be solved for the shape function weights.

To analyze the stress in various elastic bodies, calculate the strain energy of the body in terms of nodal displacements and then minimize the strain energy with respect to these parameters - a technique known as the Rayleigh-Ritz. In fact, this leads to the same algebraic equations as would be obtained by the Galerkin method but the physical assumptions made (in neglecting certain strain energy terms) are exposed more clearly in the Rayleigh-Ritz method.

In all cases, the finite elements steps are:

1. Evaluate the components of strain in terms of nodal displacements.
2. Evaluate the components of stress from strain using the elastic material constants.
3. Evaluate the strain energy for each element by integrating the products of stress and strain components over the element volume.
4. Evaluate the potential energy from the sum of total strain energy for all elements together with the work done by applied boundary forces.
5. Apply the boundary conditions, e.g., by fixing nodal displacements.
6. Minimize the potential energy with respect to the unconstrained nodal displacements.
7. Solve the resulting system of equations for the unconstrained nodal displacements.
8. Evaluate the stresses and strains using the nodal displacements and element basis functions.
9. Evaluate the boundary reaction forces (or moments) at the nodes where displacement is constrained.

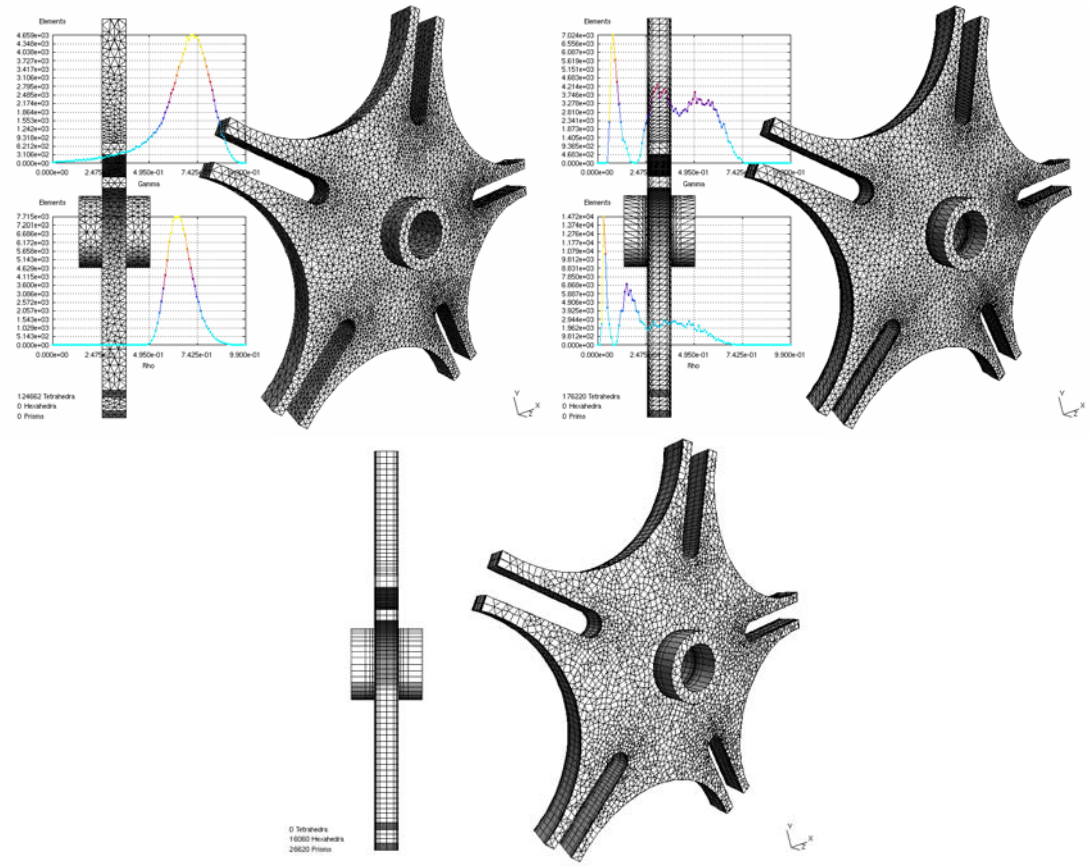


Figure 2.15 Three type of geometry discretization using gmesh (Geuzaine and Remacle, 2005).

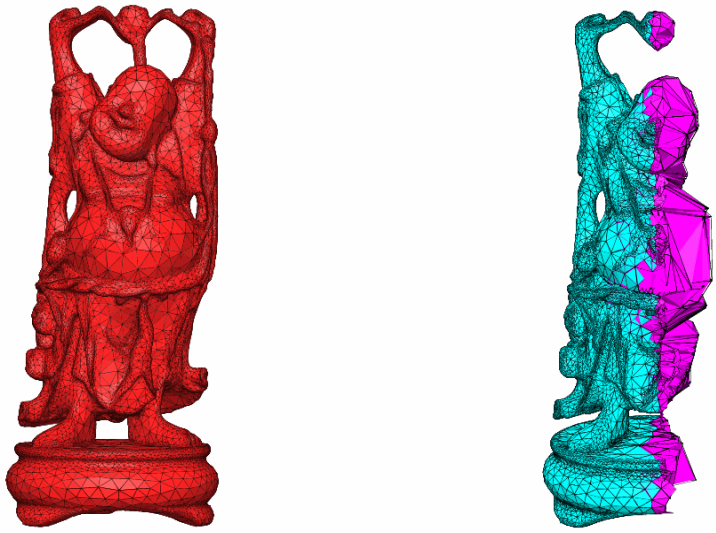


Figure 2.16 **‘**Happy Buddha’ and its sliced tetrahedral mesh version. The model is discretized using tetgen (Si, 2005)

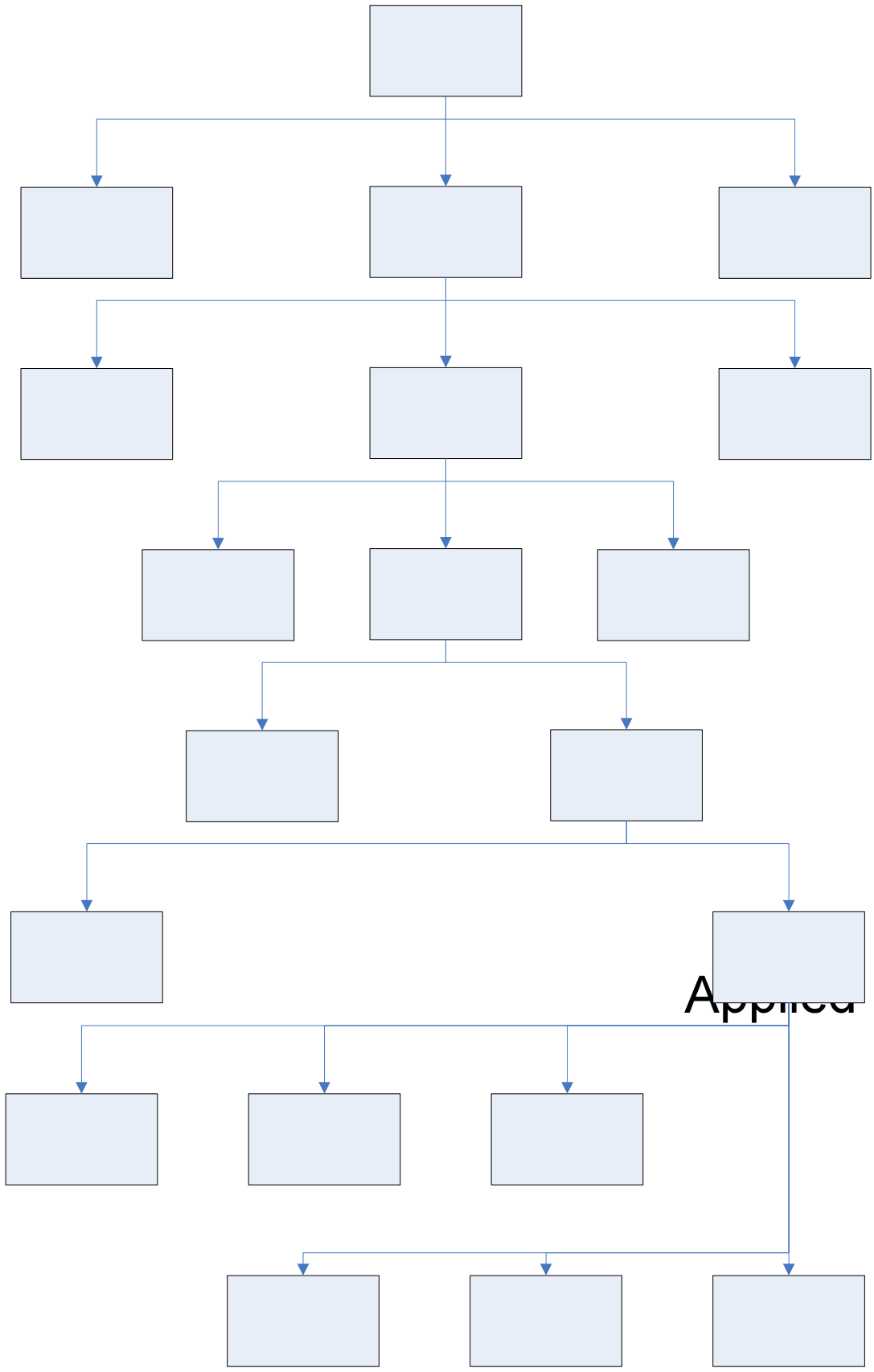


Figure 2.17 Taxonomy for finite element method from mechanical physics view.

Systems

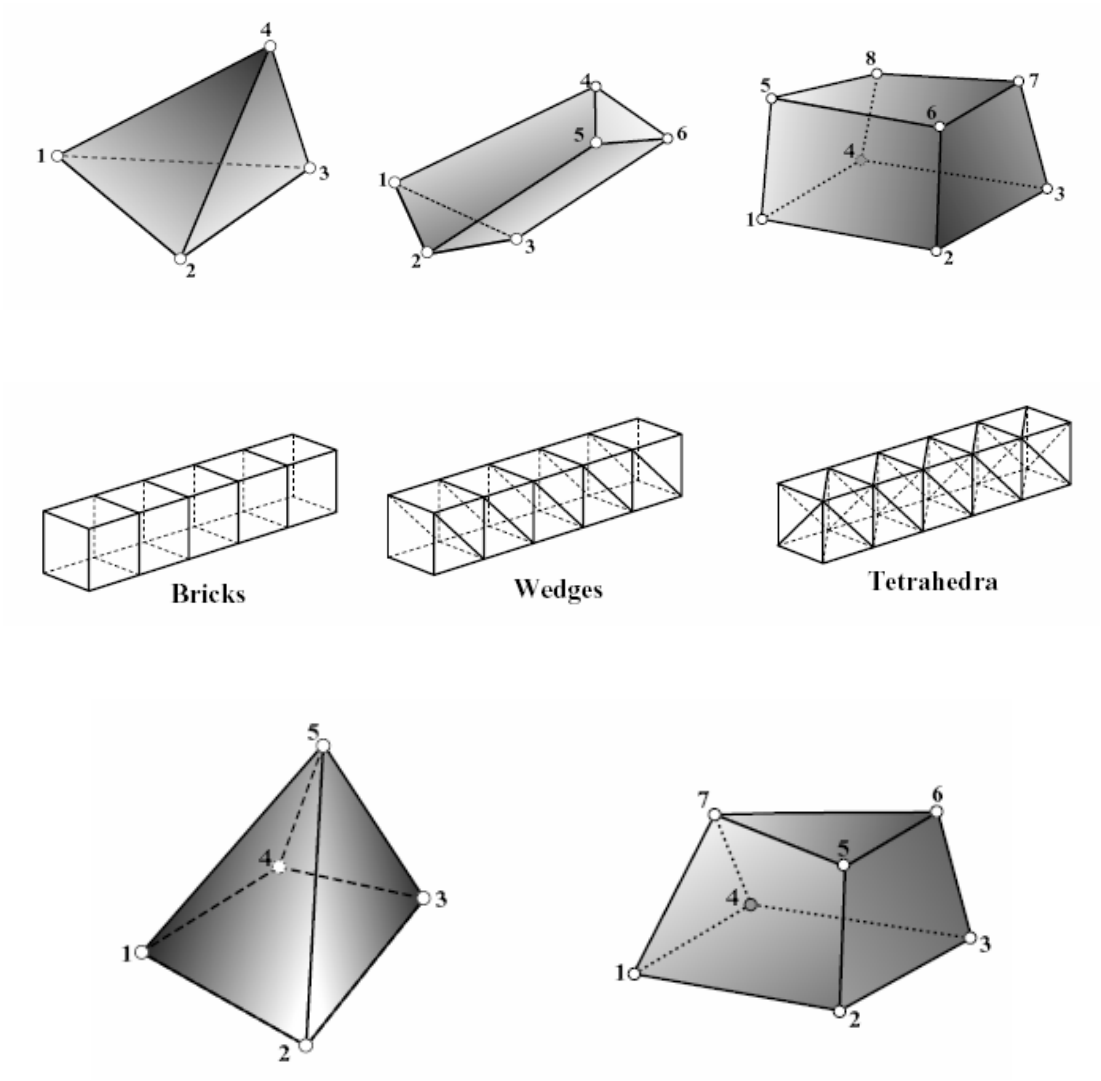


Figure 2.18 Top,the three standard solid element geometries: tetrahedron (left), wedge (center) and brick (right). Only elements with corner nodes are shown. Middle, regular 3D meshes can be built with cube-like repeating mesh units. Meshes are built with bricks, wedges or tetrahedra. Bottom, two nonstandard solid element geometries: pyramid and wrick (w(edge)+(b)rick). Four faces meet at corners 5 and 7, leading to a singular metric.

Solid elements are three-dimensional finite elements that can be used to model solid bodies and structures without any a priori geometric simplification. Finite element models of this type offer the advantage of directness. Geometric and constitutive assumptions required to produce dimensionality reduction, for example to planar or axisymmetric behavior, are avoided. Boundary conditions can be more realistically treated.

Another attractive feature is that the finite element mesh visually looks like the physical system. This directness does not come for free. It is paid in terms of modeling, mesh preparation, computing and post-processing effort. To keep these within reasonable limits it may be necessary to use coarser meshes than with two dimensional models, which in turn may degrade accuracy. Its use should be restricted to problems and analysis stages, such as verification, where the generality and flexibility of full 3D models is warranted.

Two dimensional (2D) finite elements have two standard geometries: quadrilateral and triangle. All other geometric configurations, such as polygons with five or more sides, are classified as nonstandard or special. Three dimensional (3D) finite elements offer more variety. There are three standard geometries: the tetrahedron, the wedge, and the hexahedron or “brick”. These have 4, 6 and 8 corners, respectively, with three faces meeting at each corner. These elements can be used to build topologically regular meshes. There are two nonstandard geometries that deserve consideration as they are occasionally useful to complete generated 3D meshes: the pyramid and the wrick. (The latter term is a contraction of “wedge” and “brick”) These have 5 and 7 corners, respectively. One of the corners is special in that four faces meet, which leads to a singular metric there. This singularity disqualifies these elements for use in stress analysis in highly stressed regions. However they may be acceptable away from such regions, and in vibration analysis. Both standard and nonstandard elements can be refined with additional mid side nodes. These refined elements are of interest for more accurate stress analysis. Of course, the mid side nodes may be moved away from the midpoints to fit curved geometries better. The best choices of elements and interpolation functions depend on the object shape, convergence requirements, degree of freedom, and trade-offs between accuracy and computational requirements. In general, using elements that have more nodes and more complex interpolation functions require fewer elements for the same degree of accuracy.

Consider isoparametric solid elements with three translational degrees of freedom (DOF) per node. Most of the development of such elements can be carried out assuming an arbitrary number of nodes n. In fact a general “template module” can be written to form the element stiffness matrix and mass matrix. Nodal quantities will be identified by the node subscript. Thus $\{x_i , y_i , z_i \}$ denote the node coordinates of the i^{th} node, while $\{u_{xi} , u_{yi} , u_{zi} \}$ are the nodal displacement DOFs. The shape function for the i^{th} node is denoted by N_i . These are expressed in term of natural coordinates which vary from element to element.

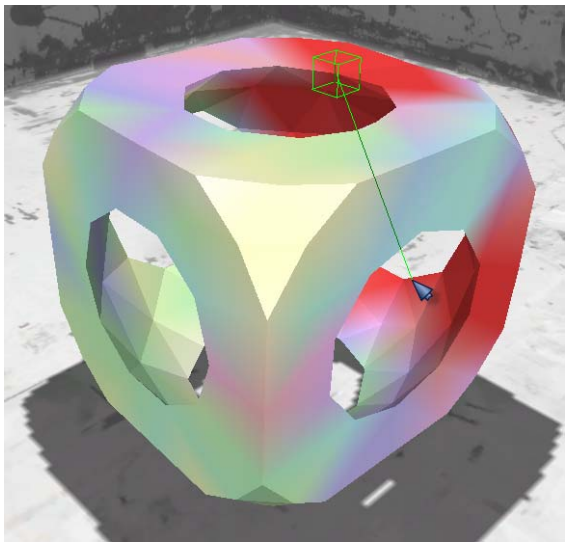


Figure 2.19 A simple finite element method deformable object in action. Image is taken from project Xplodar (Pranckevicius). High contrast red denotes high stress area while bright white denotes less stress area. Even though the simulation is performed in real time manner, notice that the deformable object is low in polygons.

Forces must be numerically integrated over volume or surface at each timestep, requiring a lot of computation. This limits the use of finite element method (FEM) for real time application despite the fact that FEM provides better deformation accuracy. Due to its complexity in nature, it is difficult to implement and optimize FEM. Discretizing the object is also quite difficult. Discretization methods chose for real time applications are based on the ability of the discretizer to maintain high geometrical accuracy with less internal elements using single simple element type (usually tetrahedron). Large deformation and topological changes

requires the system to recompute the large stiffness matrix. Finite element method requires less node points compared to mass-spring systems to achieve similar degree of deformation accuracy. This results to a smaller linear system which can be solved in less time.

Terzopoulos used finite element modeling technique to discretize the deformable objects for its offline simulator (Terzopoulos et al. 1987). The idea is to model deformable objects using differential equation analogous to the standard mass-spring-damper equation. Dynamics are computed from the potential energy stored in the elastically deformed body using finite difference discretization method. Later on, Terzopoulos extends the work to include simulation of inelastic object behaviour such as plasticity, fracture (Terzopoulos et al. 1988), heating and melting (Terzopoulos et al. 1989).

Neilson and Cotin achieved real time finite element method deformation by implementing preprocessing and equation systems condensation (Neilson and Cotin, 1996). By solving a smaller linear system, the implemented systems achieved 20 frames per second for models with 250 nodes on four Mips R4400 processor Silicon Graphics ONYX.

Although fast finite element models have been developed for medical applications (Nielsen and Cotin, 1996)(Berkley et al., 2000), less attention has been paid to displaying time dependent deformations of large size finite elements models in real-time. (Basdogan, 2001) introduces two numerically fast techniques for real-time simulation of dynamically deformable (i.e. time dependent deformations) 3D objects modeled by FEM; modal analysis and spectral Lanczos Decomposition.

Existing techniques of deformable modeling for real time simulation have either used approximate methods that are not physically accurate or linear methods that do not produce reasonable global behavior. Nonlinear finite element methods

(FEM) are globally accurate, but conventional FEM is not real-time. (Wu et al., 2001) apply nonlinear FEM using mass lumping to produce a diagonal mass matrix that allows real-time computation. They proposed a scheme for mesh adaptation based on an extension of the progressive mesh concept, called dynamic progressive meshes to minimize unnecessary computations.

Krysl et al. uses adaptive local finite element mesh refinement using wavelet theory to accelerate finite element deformation (Krysl et al., 2003). The refined mesh is nested in the refinement hierarchy, which simplifies the incorporation of multi-grid solvers. The method exploits refinement of basis functions rather than refinement of elements. It is in spirit much closer to some recent developments in the design of meshless methods. It is suitable in any number of spatial dimensions, and for a much wider variety of finite element types than any standard mesh refinement algorithm.

Finite elements method benefits from a solid background and established technique, books and vast literature. For computer applications, there are a variety of libraries for solving finite elements. Applications to discretize geometric object into sets of elements are also widely available. Compared to mass-spring method, integrating actual tissue properties are easier with finite element method. Solutions for large linear or non-linear systems using numerical techniques already exist. With constraint, some assumption and optimization, real-time computation is possible with current mainstream hardware. Finite element method allows parallel computing techniques for its simulation; enabling scalable simulations.

Finite element method is not without its drawbacks. Simulation time is slow even for linear elasticity deformation. For non linear deformation, it is even slower. To permit real-time performance, multiple accelerating strategies should be implemented. For medical application, some real-time accelerating strategies are not applicable due to limited allowable deformations and inaccuracy introduced. Finite element system is very complex and it is not that easy to implement.

2.5.2 Mass-spring method

Mass-spring method is one of the physical-based methods that have been extensively used in the field of real-time deformable object modeling. The surface or volume is discretized into a set of mass points. Each mass point is linked to its neighbors by one dimensional spring. Deformation is computed by finding equilibrium state between interconnected points after application of external force. The spring is often linear, but non-linear elasticity can be simulated by applying multi-varied stiffness springs. Mass-spring systems can also be modeled as either static or dynamic system (where time has influence).

There are multiple ways to construct the mass-spring lattices. One can construct the springs manually or discretize the object into sets of tetrahedrons (Teschner et al., 2004) (Mollemans et al., 2003) or cubes. Acquired geometry topology (tetrahedrons or cubes) are represented as configuration of point masses connected by springs.

Basically, as spring experiences external forces, the spring is either compresses or extends to the direction of the force and this creates a repulsive force to the opposite direction of the force. The created force is described mathematically by

$$F = -k * \Delta x$$
$$\Delta x = x_c - x_i$$

where F is the resultant force, k is the spring coefficient, and Δx is the distance between the two points (x_c = current distance, and x_i = distance at the inertial position). Inertial position is the distance between two separated points. No force will be generated if the points are not displaced. If the spring is compressed, then Δx will be negative, generating a positive force (expansion). If the spring is expanded, then Δx will be positive, generating a negative force (compression). Elasticity coefficient is represented by k. Also known as Young's modulus, one dimensional deformation coefficient weights the spring final force. Stiffer spring have bigger k as

it creates a larger force from its inertial state. Conversely, a spring with a smaller *k* is more flexible because it creates a smaller force from its inertial state.

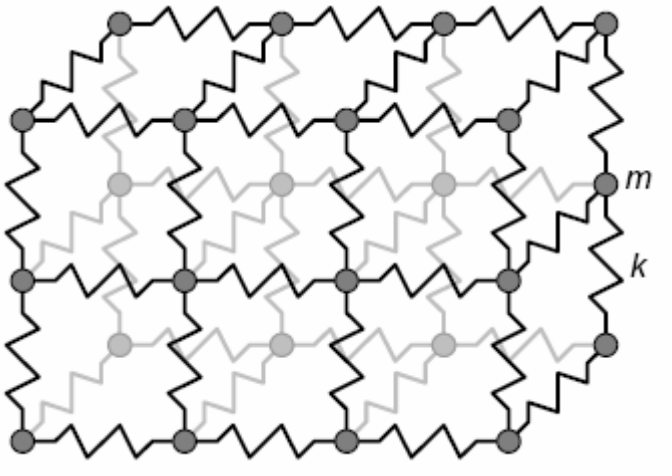


Figure 2.20 An example of mass-spring model. Connected spring exerted forces on neighboring points, displacing the points from its rest position. (Gibson and Mirtich, 1997)

To compute the distance between two points, one can use Pythagoras' theorem. Then, multiply Δx with *k* coefficient and finally use the inverse of this value to compute the force. Spring force alone is not enough to produce realistic simulation. Other forces can be applied into the system such as damping force. This is to simulate the energy loss experience by the springs. This results into an extended equation

$$F = -kx - bv$$

where *b* is the coefficient of damping and v is the relative velocity between the two connected points.

For a networked configuration of mass-spring lattices, when a spring is displaced, the resultant force propagates throughout the entire network. This results into deformable object behaviors. Based on this phenomenon, mass-spring was used in modeling string, cloth, jelly, face, human tissue and various other deformable objects. The difference between these applications is the initial spring configurations.

In a dynamic three dimensional deformable system, where time is integrated into the system, the mass m_j at position $x_i \in \Re^3$ at time t are governed by Newton's second law of motion

$$m_i \ddot{x}_i(t) + \gamma_i \dot{x}_i(t) + f_i^{\text{int}}(t) = -f_i^{ext}(t)$$

where γ denotes a damping factor, $f_i^{\text{int}}(t)$ refers to the internal forces resulting from spring interconnection and $f_i^{ext}(t)$ represents the sum of external forces applied by the user or due to gravity or collision. The equations of motion for the entire system result from assembling the equations of all masses m_i in the lattice. Writing the positions of all m masses component-wise into a position vector x of size 3n, we can state a matrix equation for the entire mass-spring system as

$$M\ddot{x} + D\dot{x} + Kx = -f$$

where M, D, and K are $3n \times 3n$ matrices representing mass, damping and stiffness, respectively. Although possibly large, these matrices are very sparse. M and D are diagonal, where K in a regular lattice is banded according to adjacency between masses. The equation is reduced into two coupled systems of first order differential equations to numerically integrated through time as

$$\dot{x} = v$$

$$\dot{v} = M^{-1}(-Dv - Kx - f)$$

The problem of solving large and complex networked configuration of mass-spring lattices calls for numerical integrators. There are many numerical integrator techniques available, but four most popular integrators are Euler, Midpoint, Runge-Kutta and Verlet. These integrators vary in its accuracy and computational cost. The fastest one but with less accurate are Euler integrator and the most accurate integrator but slow to compute is Runge-Kutta. Verlet integrator, on the other hand, is both fast and accurate integrator compared to other integrators. Accuracy is important to maintain simulation robustness. Although all integrators accumulate errors at each time-step, the highest accuracy integrators will maintain the stability of the

simulation for a longer period of time. Inaccuracy also leads to instability, where the simulation will explode and turn to chaos.

Chadwick et al. coupled multi layered mass-spring system with free form deformation for its computer animation system (Chadwick et al., 1989). The method allows for global and local deformation of articulated character. Teschner et al. approximate the object's shape into uniform tetrahedral meshes of free form deformation constraint (Teschner et al., 2004). Physical based deformation is applied to the tetrahedral meshes using mass-spring techniques where the mass-spring system will deform the free form deformation control points. Deformed free form deformation control points will then deform the underlying vertices. To preserve volume undergoing deformation, volume and surface preserving coefficient is introduced to the mass-spring system. This two fold deformation method which coupled mass-spring system and free form deformation allows for high geometry deformation as the rendering geometry and deformation geometry are independent of each other. Other hybrid method of mass-spring systems is by Christensen et al (Christensen et al., 1995) where the deformable object is approximately wrapped with simple mass-spring lattice configuration. Then physical based deformation is applied to the mass-spring where the lattice configuration will act as free form deformation constraint to the actual object geometry. This method is used for animating characters in 3D animation. Cotin et al. combined finite element method and mass-spring system for virtual surgery application (Cotin et al., 2000). Finite element method is used to model tissue deformation using pre-computed deformations allowing large deformation. To enable volume cutting and topological changes to the tissue, a mass-spring model variant called tensor mass model is applied into the system.

Baraff et al introduced implicit integration for its mass-spring cloth simulations (Baraff et al., 1998). By using implicit integration, the system is much more stable and independent from number of particles used. Fuhrmann et al. describe and algorithm which replaces the internal cloth forces by several constraints and therefore can easily take large time steps (Fuhrmann et al. 2003). Instability, inaccuracy and speed problem for numerical integration can be minimized by using

Verlet integrator. Jacobsen uses velocity less Verlet integration for its real time physic systems (Jacobsen, 2003). Teschner et al. have perform a little experiment on various integrators to find the fastest integrator and have proved that Verlet integrator is the best numerical integrator suitable for mass-spring systems period (Teschner et al. 2004).

Mass-spring systems are easier to implement than finite element method. Computation cost for mass-spring systems are much lower compared to finite element method, therefore mass-spring systems have much wider appeal for real-time applications. Non linear deformable object can also be modeled by mass-spring systems. In addition, mass-spring systems are suitable for parallel processing allowing a scalable simulation platform.

Since mass-spring systems rely on numerical integrators, the systems are vulnerable to convergence and instability. The principle of mass-spring systems defined that force travels according to the spring's links, not by continuum. This physical approximation is too coarse to be applicable for some critical applications. Certain applications requirement such as specific constraint and materials properties cannot be modeled with mass-spring systems. Behavior of incompressible materials and thin object are unpredictable if modeled using mass-spring systems. It is hard to model material stiffness by setting spring coefficient parameter. Sometime the deformation acts differently than desired behavior. Even after successfully tuning the spring coefficient, other coefficient, for example gravity, when changed, the spring coefficient have to be tuned all over again.

2.5.3 Gas pressure method

Matyka and Ollila proposed a novel technique for modeling elastic soft body object (Matyka and Ollila, 2003). Soft body is described as three dimensional deformable meshes which always keep constant volume. The method is based on simple thermodynamics laws and uses the Lausius-Clapeyron state equation for pressure calculation. The pressure force is accumulated into a force accumulator of a 3D mesh object by using mass-spring technique. Behavior of soft body is obtained after the integration of Newton's second law of motion with fixed or non-fixed air pressure inside of it. Simply put, the idea is to create a closed mass-spring cloth represented as manifold mesh object and put air pressure inside it.

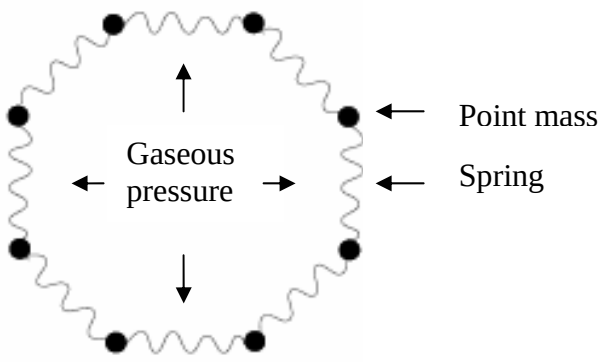


Figure 2.21 Example of gaseous pressure method for simple two dimensional meshes. The mesh must be manifold, represented as wrapped cloth which will have ideal gas pressure inside. (Matyka and Ollila, 2003)

To enable simple pressure formulation, Matyka uses ideal gas approximation which is defined as one in which all collisions between atoms or molecules are perfectly elastic and in which there are no intermolecular attractive forces. One can visualize it as a collection of perfectly hard spheres which collide but which otherwise do not interact with each other. In such a gas, all the internal energy is in the form of kinetic energy and any change in internal energy is accompanied by a change in temperature.

An ideal gas can be characterized by three state variables: absolute pressure *P*, volume *V*, and absolute temperature *T*. The relationship between them may be deduced from kinetic theory and represented by

$$P = \frac{nRT}{V}$$

where n is the number of moles and R is universal gas constant . To calculate pressure for the point of the shape, the expression used is

$$\vec{P} = P \cdot \hat{n} \left[\frac{N}{m^2} \right]$$

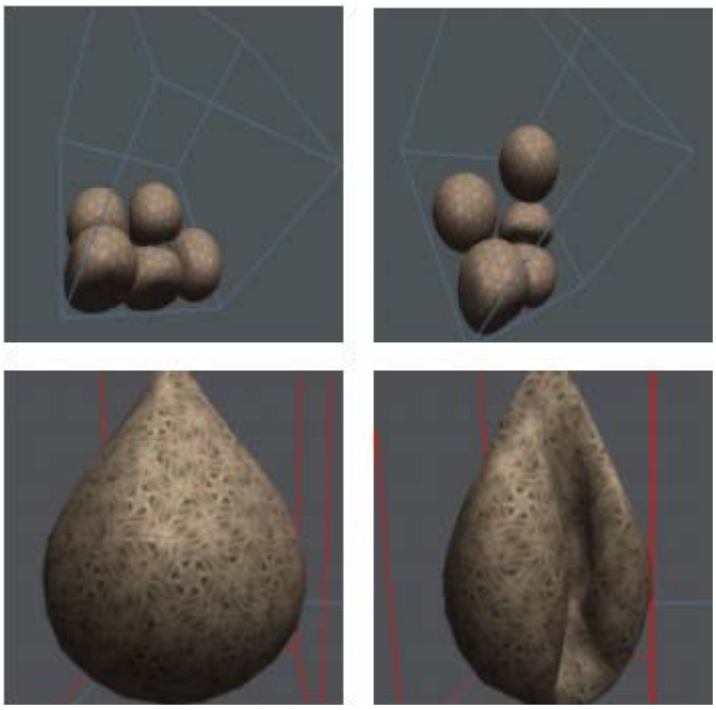


Figure 2.22 Screen shot of of gas pressure method for three dimensional volumetric deformable objects (Matyka and Ollila, 2003). The simulation is fast enough to be performed in real time.

Next, the volume of the deformed body has to be recalculated to measure the gas pressure inside the object. Matyka uses simple bounding geometry such as sphere, box and ellipses to approximate the current volume. A better volume computation method is presented by Owen using Gauss's Theorem. Gauss's

Theorem relates the divergence of a vector field within a volume to the flux of a vector field through a closed surface by the following

$$\iiint_v dell \bullet Fdv = \iint_s F \bullet da$$

where the surface s encloses the volume v. Detail theory and implementation are available at (Owen, 2005).

Deformation based on ideal gas pressure method does proved to be fast(able to perform real time deformation with coupled thousand of vertices) (Matyka and Ollila, 2003). The method is simple to implement and requires no extensive geometry discretization preprocessing (unlike finite element method). Since its volume dynamic is represented as simple ideal gas equation, it does not exhibit complex internal volume structure like volumetric mass-spring method and finite element method to compute internal dynamics. Finite element method and volumetric mass-spring stored invisible internal geometry topology data for dynamics processing while gas pressure method only store visible surface geometry topology data which means less memory footprint.

Albeit all gas pressure method strengths, it's not without weaknesses. It is very hard to define the deformation coefficient (Young's modulus and pressure coefficient) to model desired material. Deformation behavior looks like a balloon filled with water placed underwater. From the available demo, it doesn't look like a balloon filled with gas at all. Since it uses mass-spring technique which consist of numerical integration, gas pressure method inherit mass-spring drawback which is numerical integration accuracy and stability. The deformation is prone to explode if it undergoes huge deformations.

2.5.4 Mesh free method

Numerical methods like Finite Elements, Finite Volumes and Finite Differences are already very well developed. However, there are limitations to these methods. First of all the time an engineer spends on solving a problem, goes mainly into the meshing of his solution domain. Secondly, the mesh is sensitive to large deformations, which can cause accuracy deterioration. To circumvent the meshing as a whole and make the problem more flexible, the so-called mesh free methods are invented.

To give some applications of this method, first the differences between the mesh free methods and the other methods should be clear. Instead of using a pre-defined mesh, mesh free methods only use node generation (giving the points without the need to prescribe the relationship between the nodes) and for each node a shape function is created. Since the mesh less method does not describe point topology explicitly, neighbor search is fundamental in finding the equilibrium state of the deformed object. The lack of topology structure and the ability of the system to self organize provides a system that is able to simulate a wider range of deformable material compared to commonly used deformation technique. The next step is to form a system of equations and solve this system.

Common geometric representations approximate the body by a mesh of nodes of fixed topology which are not adapted to the animation of substances undergoing large inelastic deformations. In this case, the use of mesh less method for object representation and dynamic representation is more appropriate. These systems are unstructured in the sense that interactions between point masses do not depend on a specified graph of connections, but on distance. The need to simulate various complex deformation types such as melting, solidifying, splitting and fusion motivated the use of mesh less method in modeling deformable objects in the field of computer graphics.

To derive inter-point forces, Tonnesen used the pair-wise Lennard-Jones potential energy functions as a dynamics system solution (Tonnesen and Szeliski, 1992). To enable stretching and growing, Tonnesen introduced orientation to the point's properties. Under large deformation, Tonnesen proposed a kd-tree hierarchical data structuring approach to compute forces and torques at reduced number of points. By spatially subdivide the object space within some radius (natural inter-points spacing), all to be deform neighbor points can be efficiently found. To further reduce the computation, this operation is occasionally performed and cache list of neighbors were used for intermediate time steps. New points were added when neighboring points have large enough space between them and still under maximum number of allowable points between the ranges.

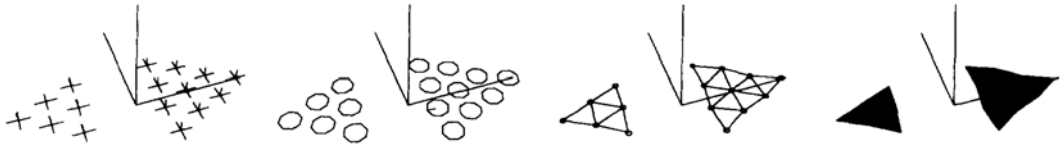


Figure 2.23 Rendering techniques for particle based surface; axes, discs, wireframe triangularion and flat shaded triangulation (Tonnesen and Szeliski, 1992)

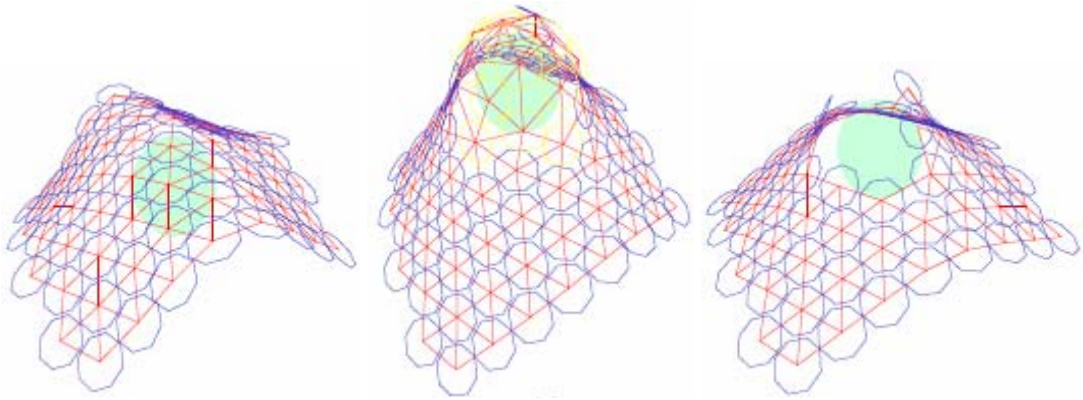


Figure 2.24 Left, deforming. Center, deforming and surface restructuring by adding new points. Right, deforming and tearing. (Tonnesen and Szeliski, 1992)

Each point is given state variables of position and mass for the system to interact with the dynamics. For more complex systems, additional state variables

combined with simple heuristics were formed to create application specific behaviors. The surface is rendered as iso surface which yield an implicit coating of the point which handles topological changes such as splitting and merging by construction.

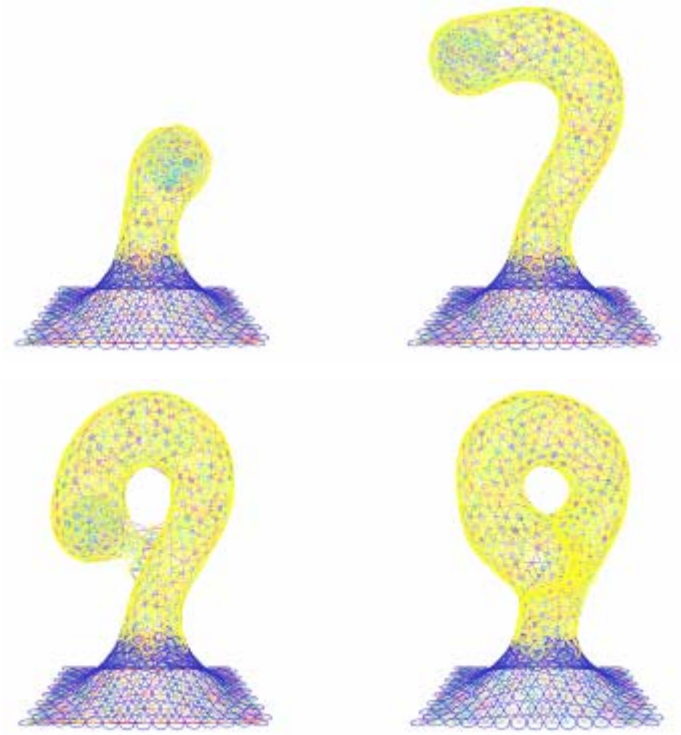


Figure 2.25 Fusioning deformable objects (Tonnesen and Szeliski, 1992)

The Lennard-Jones potential is well known in molecular dynamics for modeling the interaction potential between pairs of atoms. It creates long-range attractive and short-range repulsive forces, yielding particles arranged into hexagonally ordered 2D layers in absence of external forces. Increasing the dissociation energy (magnitude of the potential energy) increases the stiffness of the model, while the width of the potential energy can be varied. Therefore, large dissociation energy and high potential energy exponents yield rigid and brittle material, while low dissociation energy and small potential energy exponents result in soft and elastic behavior of the object. This allows the modeling of a wide variety of physical properties ranging from stiff to fluid-like behavior. By coupling the dissociation energy with thermal energy such that the total system energy is

conserved, objects can be melted and frozen. Furthermore, thermal expansion and contraction can be simulated by adapting the equilibrium separation distance to the temperature.

Desbrun and Cani (Desbrun and Cani, 1995) (Desbrun and Cani, 1996) (Desbrun and Cani, 1999) use smoothed particle hydrodynamics approach used by physicists for cosmological fluid simulation as its deformable dynamics basis. The Smoothed Particle Hydrodynamics (SPH) formalism was introduced by physicists for accurate simulation of fluid dynamics. Simulating a fluid consists in computing the variations of continuous functions such as mass density, speed, pressure, or temperature over space and time. Standard finite element techniques in hydrodynamics use an Eulerian approach: they consist of dividing space into a fixed grid of voxels, and then studying what flows in or out of each voxel. However, this kind of approach requires the division of huge empty volumes and is not intuitive for flows.

SPH belongs to an alternative approach, called the Lagrangian approach that consists of following the evolution of selected fluid elements over space and time. The particles can be viewed either as matter elements or sample points scattered in a soft substance. Each of them represents a small volume of inelastic material that moves over time. In practice, smoothed particles are used to approximate the values and derivatives of continuous physical quantities, such as local mass density or pressure that need to be computed during the simulation. Smoothed particles ensure valid and stable simulation of a state equation describing the physical behaviors of the material. It is also used for deforming the surface of the substance in a coherent way using the level sets of the mass density function. To reduce computation time, adaptive time steps for integration is used according to a local stability criterion along with efficient data structure for neighbor search. Desbrun further the research for rendering the point particles using implicit surface rendering method (Desbrun and Cani, 1996).

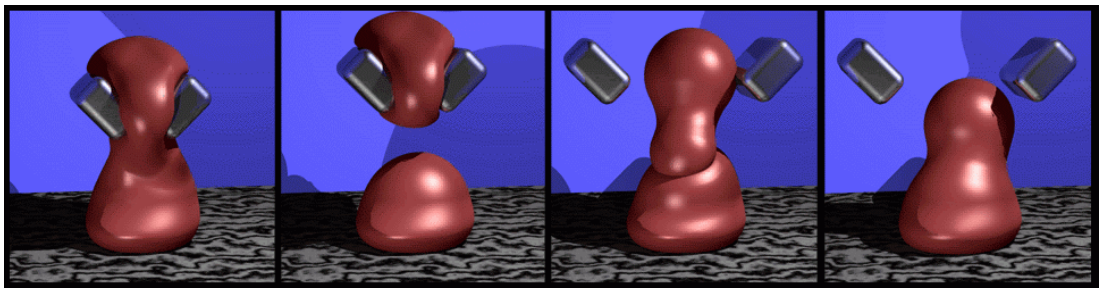


Figure 2.26 Deformable object are splitted and then fused together. (Desbrun and Cani, 1996)

Using mesh less method, dubbed point based method; Keiser et al. were able to simulate wide range of material properties such as stiff elastic to highly plastic using a single application framework (Keiser et al., 2004). By using points for both volume and surface representation, arbitrarily large deviations from the original shape can be simulated. In contrast to previous mesh less based elasticity in computer graphics, the physical model is derived from continuum mechanics, which allows the specification of common material properties such as Young's Modulus and Poisson's Ratio.

In each step, spatial derivatives of the discrete displacement field were computer using a Moving Least Squares (MLS) procedure. It is from these derivatives that strains, stresses and elastic forces at each simulated points were obtain. Equations of motion for these forces were solved using both implicit and explicit integration. Point sampled surface were rendered dynamically adaptive for scalable and faster performance. Although material anisotropy can be simulated, only linear elasticity are implemented in the dynamic system. MLS only works if there are at least 3 neighboring points within non-degenerate locations. This makes it only suitable for volumetric objects, not two dimensional or one dimensional object. The nature of the system is close proximity points always interact with each other. This makes it difficult to model fracture and brittle materials. Even with stiff coefficient, hard edges are difficult to achieve.

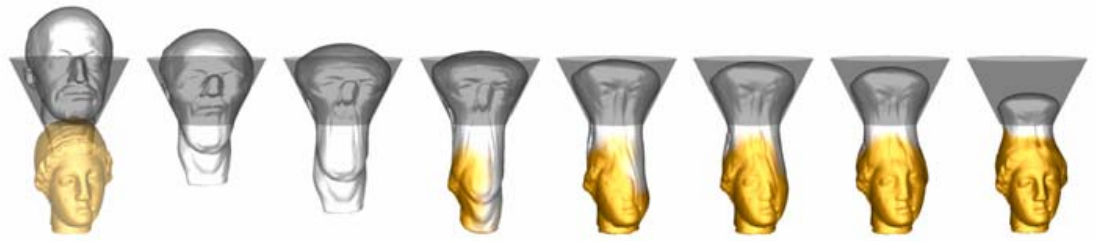


Figure 2.27 Target morph using point based method. (Keiser et al., 2004)

Deformable object ranging from stiff elastic to highly inelastic objects can be modeled efficiently using mesh less method due to its natural properties of not having topological properties explicitly. Surfaces are easy to shape, extend, fusion and split. Material properties such as stretching, bending or variation in curvature can be controlled by adjusting strength of various potential energy functions. Input model doesn't have to be discretize into elements which is a requirement for finite element method.

Mesh free method application in computer graphics deformable object simulation is quite new. The first idea implementation was seen in 1995(Desbrun and Cani, 1995). With this method, objects are easy to deform and new deformed shape are easy to construct for the purpose of rendering (no topology needed). Material stiffness and other properties such as resistance to stretching, bending can be controlled by adjusting strength of various potential energy functions.

One problem of mesh free method is that the surface is not explicitly defined thus poses a problem rendering the points. The points cannot be rendered using trivial geometry rendering technique. It is harder to achieve exact control of the shape. Usually, sampled points are shape approximation of the original object shape. Hard edges are also hard to preserve during point sampling of the object. Accurate dynamic computation is expensive. To enable real time performance, implementation must include heavy optimization. The lack of precise control and shape degeneration due to point sampling makes it unsuitable for engineering purpose.

2.5.5 Pros and Cons

Physically based deformable models have seen wide application in many fields of computer graphics. The ability to simulate real world various material behaviors does prove to be useful in the field of medical and engineering. Physical based model limits the direct user controls of the deformation process. Deformations are computed using approximations of physical dynamics. Sometimes deformation behavior is unpredictable due to gross approximation of dynamics. This can be seen when tuning mass-spring system spring stiffness for specific materials. Unlike most non-physical based deformation technique, deformation parameters for physical based technique are much more complicated to configure. With limited computing power, computing complex dynamics is very expensive. For finite element method, internal geometry structures are required for dynamics computation. Gas pressure method on the other hand, does not have this internal geometry structure for its dynamics computation thus making it less memory footprint requirements. Physical based method does not appeal to some computer graphics application especially in the field of object modeling and editing because of it gives user limited control of deformations.

2.6 Real time modeling technique

Physical based deformable object behavior simulation requires lots of complex dynamics computation. This phenomenon burdens the processor and it is very hard to achieve robust physically realistic behavior in real time. Earlier work on deformable object animation focuses on modeling deformable object on the computer platform (Terzopoulos et al., 1987) (Terzopoulos et al., 1988) (Terzopoulos et al., 1989) (Witkin and Baraff, 1997) (Baraff, 1996) (Baraff and Witkin, 1992) (Foster and Metaxes, 1996) (Szelinski and Tonnesen, 1992) (Stam, 1993) (Tonnesen, 1991) (Tonnesen, 1992) (Breen et al., 1994). Most of them is too

complex and requires huge computation per frame thus not suitable for real time and interactive applications. This section will discussed techniques, idea and implementation from previous researcher to accelerate deformable object behavior simulations.

For better performance, it is highly desirable to construct adaptive discretizations, allocating resources where they can be most profitably used. Usually this is constructed by adaptively refine either by object complexity or deformation complexity. Adaptively refine object complicity is perform by refining or coarsening the mesh resolution accordingly. Adaptively refine deformation complicity is perform by using either more complex or simpler deformation functions accordingly.

Adaptive finite element computations rely on adjustments of the spatial resolution of the domain discretization to deliver higher accuracy where it is needed. When the domain is discretized into a finite element mesh, a possible option, albeit somewhat expensive and in some cases complex, is to create a new mesh with the desired resolution, known as remeshing. Another alternative is to adjust the density of the mesh by performing local refinement of the existing mesh so that in some regions finite elements are split to decrease their “size”, in other regions they are merged to reduce the resolution. Both remeshing and refinement have their advantages and disadvantages. For detail discussion on this topic, please refer to work by Grinspun et al (Grinspun et al., 2002).

Debunne et al. uses automatic space and time adaptive object representation level of detail technique. It allows local refinement or simplification of the computational model based on local error measurement. (Debunne et al., 2001) (Debunne et al., 2000) (Debunne et al., 1999). Object is partitioned in a non-nested multi-resolution hierarchy of tetrahedral meshes (Debunne et al., 2001) (Debunne et al., 2000) or adaptively refined particle resolution (Debunne et al., 1999). At each deformation step, object sampling is refined to concentrate computation on region with the most deformation. Local contact area is deformed with highly detailed

object representation while further areas are computed with grossly approximate object representation. This method reduces computation time while at the same time preserve object and deformation complexity.

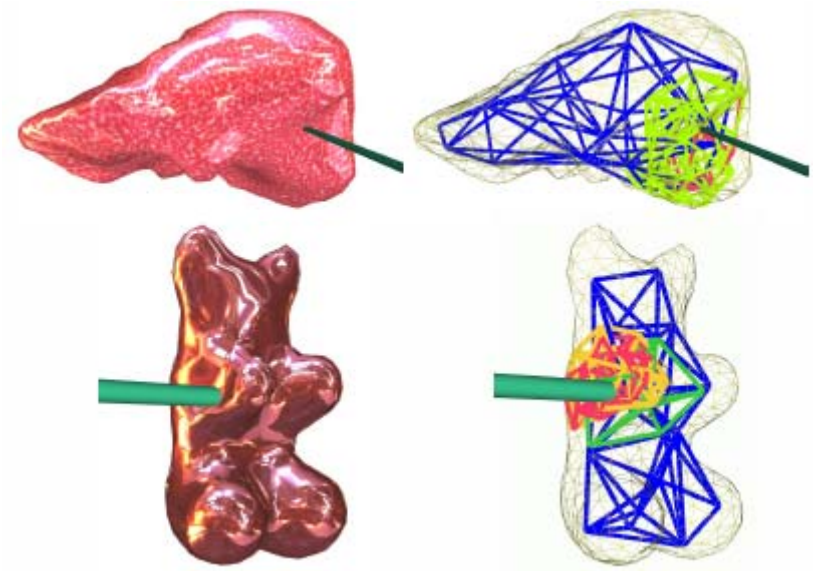


Figure 2.28 Debunne et al. uses local refinement of multiresolution models to reduce computation time by reducing geometry for run time dynamics processing. (Debunne et al., 2001)

Instead of adaptive refinement of object representation, Grinspun et al. prefers method based on adaptive refinement of finite element basis function (Grinspun et al., 2002) (Grinspun et al., 2003). Dubbed CHARMS(conforming, hierarchical, adaptive refinement methods), this method removes a number of implementation headaches associated with other approaches(geometry reconstruction and merging between multi resolution representation) and is a general technique independent of domain dimension (2D and 3D), element type (triangle, quad, tetrahedron, hexahedron), and basis function order (piecewise linear, higher order B-splines, loop subdivision, etc).

Wu et al. uses progressive meshes to simplify object surface geometry for his surface based nonlinear finite element simulations (Wu et al., 2001). Since it uses

Hoppe's progressive meshes, mesh refinement hierarchy can be pre-computed and stored for online fetching. Integration of finite element solver and mesh hierarchy are described in detail in the paper.

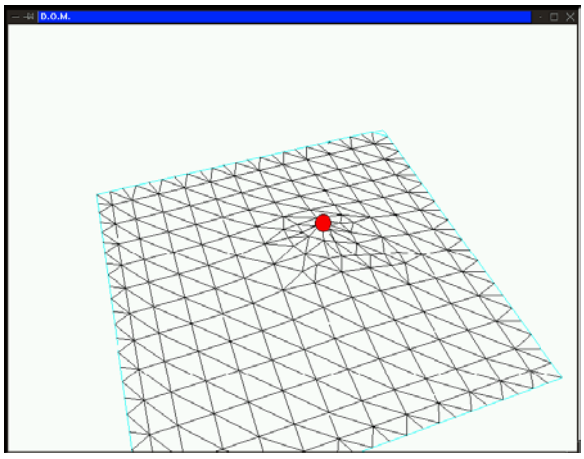


Figure 2.29 Dynamic progressive meshes is used to refine local contact area to enhance dynamics computation (Wu et al., 2001)

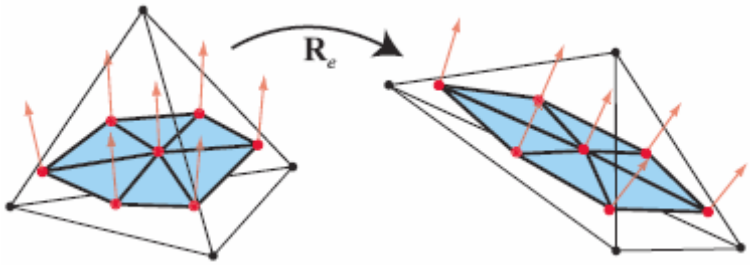


Figure 2.30 Vertices of the surface mesh is displaced according to the displacement field of the tetrahedron in which they lay using barycentric coordinate system (Muller and Gross, 2004).

Muller and Gross achieve interactive rates for its deformable simulator by using two different representations for the same deformable object. A low resolution volumetric mesh for the finite element method simulation and a high resolution surface mesh for rendering. To animate a surface mesh consistently with a volumetric mesh, Muller linked every vertex of the surface mesh to the closest tetrahedron in the volumetric mesh and store its barycentric coordinates with respect

to that tetrahedron. During the simulation, the position of each vertex of the surface mesh is interpolated from the positions of the linked tetrahedron using the stored barycentric coordinates (Muller and Gross, 2004).

Another way to reduce computation for deformable object simulation is by pre-compute complex computation and stored in a database system for online data fetching. Or by performing possible displacement (usually under some sort of constraint) and stored the displacement data so that in real time simulation, displacement need not to be computed. James and Fatahalian pre-computed data driven models of interactive physically based deformable models (James and Fatahalian, 2003). The method pre-computes impulsive dynamics by driving the scenes with parameterized interactions. By using data driven tabulation of the system's deterministic state space dynamics, and model reduction efficient low rank parameterizations of the deformed shapes are built. Storage spaces are constraint by projecting the state space models into very low dimensional spaces using least squares approximations motivated by modal analysis. Phase space dynamics are sampled using parameterized impulse response functions. Interactions are defined in discrete impulse palettes to constrain the range of user interactions.

James and Pai implement a pre-computation method for its deformable object simulation for haptic devices (James and Pai, 2001). The method pre-computed Green's functions and fast low rank updates based on Capacitance Matrix Algorithms. This method is from the fact that linear models allow many systems responses (Green's function) to be pre-computed. Coupled with boundary element method, the deformable object simulation can achieve high frame rate (by pre-computation) with high accuracy (by boundary element method which is sibling of the finite element method) (James and Pai, 1999).

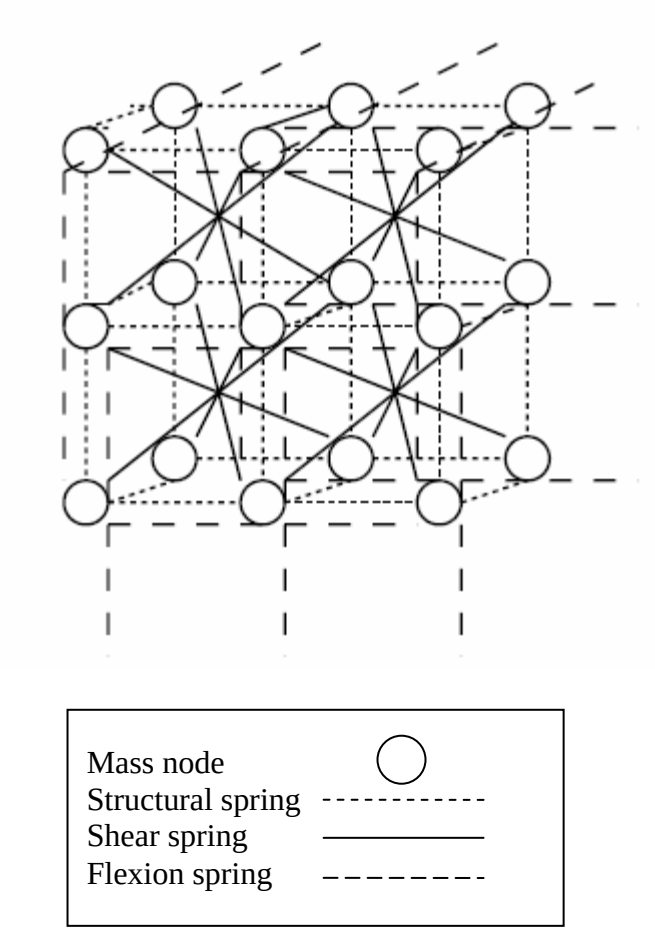


Figure 2.31 Chen et al. mass-spring systems lattice configurations adapted from Provot cloth mass-spring configurations (Chen et al., 1998).

Finite element method is accurate but it does not favor the available computation resources. Another method to accelerate deformable object simulation is by using much simpler dynamics. Based on one dimension dynamics, mass-spring systems are used extensively in the field of deformable surface modeling. Some researchers have extended the use of this method for volumetric objects. Mollemans et al. developed a tetrahedral soft tissue model that can be used in surgery planning systems consisting of mass-spring systems (Mollemans et al., 2003). Object is discretize into sets of tetrahedral. Points are described as mass points and tetrahedral topology are described as springs connecting two points. Another variant of mass-spring systems for volumetric object can be seen from the work of Chen et al. (Chen et al., 1998). The approach is a 3D extension of the discrete mass-spring meshes of Provot (Provot, 1995). Multiple types of springs are introduced namely structural

springs, shear springs and flexion springs. Not all springs stiffness are computed in each step. Under pure shear stress, only shear springs are constrained. Under pure compression, only structural springs are constrained. Under pure flexion stress, only flexion springs are constrained.

Teschner et al. uses uniform tetrahedral volume discretization for its mass-spring systems (Teschner et al, 2004). In contrast to Chen et al. method, Teschner et al. introduce six distance preserving forces between all pairs of points; four area preserving forces and two volume preserving forces. To even more accelerate the deformation simulation, spring configurations are much coarser than actual object geometry. To preserve geometry complexity, Teschner et al. embed actual geometry vertices into the tetrahedral using spline based free form deformation principles. Verlet integrations are used as numerical integration for its speed and stability.

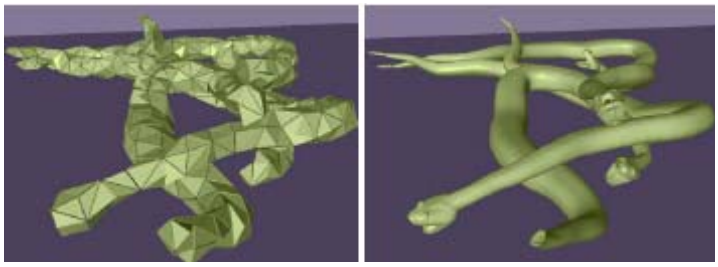


Figure 2.32 A low resolution uniform tetrahedral mesh and a high resolution surface mesh of a snake. Deformation is computed for low resolution tetrahedral mesh using mass-spring systems and high resolution mesh is used for rendering.(Teschner et al, 2004).

Another hybrid method for deformable object simulation is by Cotin et al (Cotin et al., 2000). The method works two fold. First, pre-computation of finite element method deformations are used as base to deform large size meshes in real time. Although this method can perform faster deformation, it doesn't permit topological changes to the deformable mesh. To combat this limitation, Cotin integrate another method to the deformable simulation system; a mass-spring model where topological changes can easily be made.

Gibson presented a deformation algorithm for object with high polygon count (Gibson, 1997). The idea is by using simple mathematical function for its dynamics and deformations are propagated from contact area. The systems works by finding distances between neighboring points and displace the points if it reached constraint limits. Using simple data structure, high speed deformation is achieved in regard to force propagations. Gibson extended the work to introduce anisotropy material in (Gibson et al., 1998).

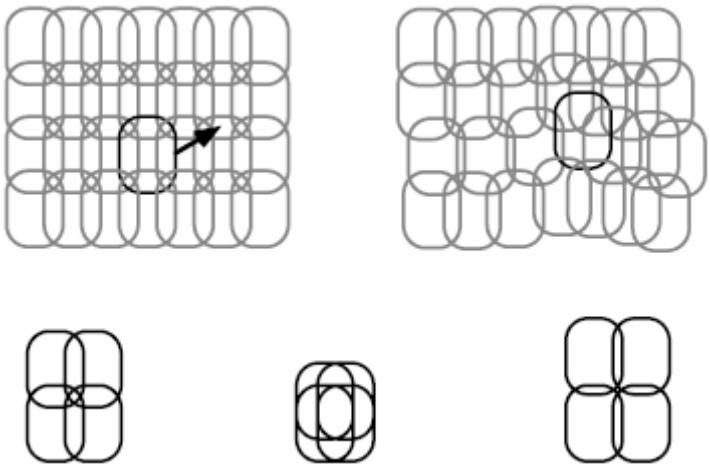


Figure 2.33 Chainmail works by constraining distances between neighboring points (Gibson, 1997). Upper left image shows initial state of the chainmail systems. Upper right image shows deformed chainmail systems. Lower left image shows chainmail systems at its initial state. Lower middle image shows maximally compress chainmail and lower right shows maximally stretch chainmail.

Bro-Neilsen and Cotin compress the linear matrix systems resulting from the volumetric finite element model to a system with the same complexity as a finite element surface model of the same object (Bro-Neilsen and Cotin, 1996). By simulating only the visible surfaces nodes, they achieve speed increase compared to traditional volumetric based finite element method. The condensation method used allows volumetric deformation behavior despite the use of only surface finite element systems.

The process of breaking complex vibration into its component modes of vibration, very much like frequency domain analysis breaks vibration down to component frequencies is called modal analysis. For deformable objects, modal analysis is the process of taking the nonlinear description of a system, finding a good linear approximation, and then finding a coordinate system that digitalizes the linear approximation. This process transforms a complicated system of nonlinear equations into a simple set of decoupled linear equations that may be individually solved analytically. Hauser et al. developed a system that models deformable objects using hybrid formulation that combines rigid-body motion with deformation computed using modal analysis (Hauser et al., 2003). Modal decomposition is through the process diagonalizing general nonlinear physical equation.

Faster real time deformable object simulations can be achieved by multiple types of acceleration techniques. The difference between each type of acceleration strategy differs in its results. Some are accurate, some are able to simulate anisotropy materials, some can simulate non-linear deformation, some are for limited or small deformations and some support topological changes. Whatever the results are, real time deformations are crucial for broad field of applications.

CHAPTER III

METHODOLOGY

3.1 Project planning

This chapter describes how the research was conducted. Firstly, theoretical framework for this project will be discussed. Descriptions of software development, testing methodologies, software specifications and hardware specification will follow.

3.2 Theoretical framework

Basically, the end application is a physical based deformation system. The system takes an object, performs deformation on the object and renders it on screen. Deformation is performed based on mass-spring system method. An additional algorithm is added before any deformation is performed in order to select nodes (vertices of the object) to be deformed.

The simulation system is divided into two main phases, preprocessing phase and run time processing phase.

The objective of preprocessing phase is to provide a suitable data for the run time processing phase. In this context, suitable data is data that does not require any more data processing during run time. It consists of two modules. The first module, tetrahedral discretization, will discretize input data, in this case original geometry of deformable objects, into tetrahedral meshes. Geometric based representation is chosen over other representation such as voxels to enable full hardware support of polygons rendering. The second module will built a data structure for mass-spring system from the tetrahedral meshes provided by the first module.

The second phase, run time processing phase, is the mass-spring system rendering loop. It consists of three main modules (standard mass-spring system) with one additional module. The three main modules are collision detection and response module, deformation processing module and rendering module. The first module will detect and solve collision for the deformable object based on applied concentrated loads. Based on the collision response, the simulation system will select area for deformation in the selection of nodes module. When the area for deformation is defined, the selection of nodes module will provide deformation processing module information of area to be deformed. This way, actual deformation is performed on smaller area compared to traditional method where deformation processing is performed on the whole object. Finally, rendering module will render the deformable object. The second phase will loop itself until terminated by the user. For this research, the focus is on the selection of nodes for deformation module.

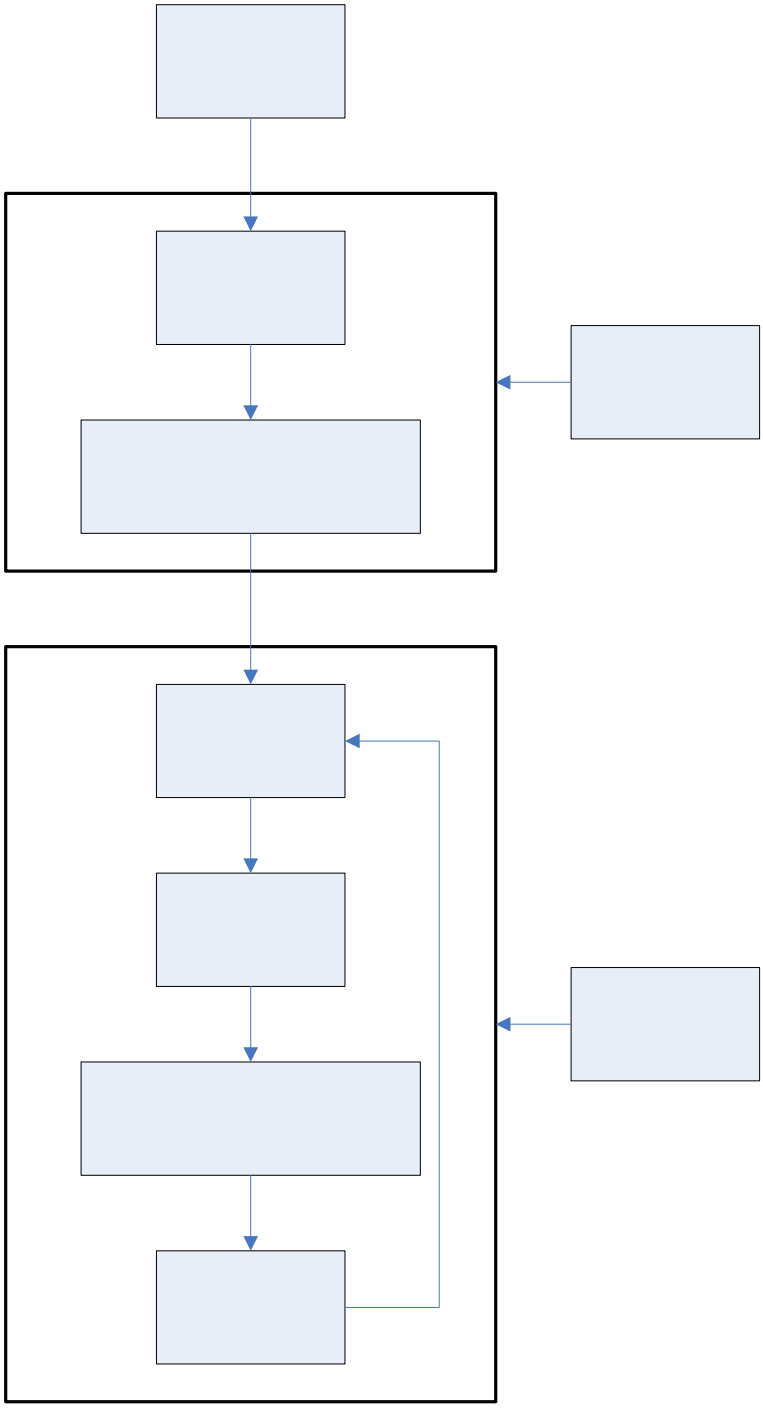


Figure 3.1 Conceptual diagram of the deformable object systems.

To achieve real time deformable object simulation, this project will follow some basic strategies described in previous chapter. Some interesting ideas are;-

1. Adaptive geometry for dynamics computations
2. Hybrid; mass-spring systems and volume embedding

3. Simple dynamics; either by dynamics simplification or use existing simple dynamics
4. Force propagations
5. Modal analysis or subdivide nonlinear elasticity systems into sets of linear elasticity
6. Velocity-less numerical integrations (Verlet integrators); faster yet more stable

The basic idea for nodes selection is based on force propagation idea. If force applied to a node of the object, the force will propagate throughout the object. By propagating deformation similar to force propagation nature, deformation processing time can be reduced as number of nodes for deformation is reduced. Conceptual idea of the systems is visualized in Figure 3.1.

3.3 Software development

Actual implementation will use structured C as its main programming language although some features available only to C++ implementation are not restricted to be used. C is a matured language and lots of freely available libraries can be use with the language. Microsoft Visual Studio .NET 2003 was chosen as the main Integrated Development Environment (IDE) for the project implementation. Included in the packages is Microsoft Visual C++ .NET 2003. It is a powerful tool for creating Microsoft Windows®-based and Microsoft .NET-connected applications, dynamic Web applications, and XML Web services using the C++ development language.

The IDE provides a robust development environment comprises compilers that are conformant to the International Standards Organization (ISO), a Standard

Template Library (STL) implementation, industry-standard Active Template Library (ATL) and Microsoft Foundation Class (MFC) libraries, WinForm .Net libraries and powerful integrated development environment features enabling efficient editing and debugging of source code. The most important integrated library is STL which provide easy, robust, optimized way for managing dynamic data. The latest Microsoft STL implementations are fully compatible across various platforms such as Linux and SGI's.

User interface is designed visually using the provided tools and Windows Forms and components. It has a powerful debugger and advanced compilers, offering advanced options for code generation on 32- and 64-bit platforms. Other alternatives for user interfaces are Fast Light Tool Kit (http://www.fltk.org/), Fox toolkit (http://www.fox-toolkit.org/), QT toolkit (http://www.trolltech.com/products/qt/index.html) and wxWindows (http://www.wxwindows.org/) which are freely available and cross platform.

Visual C++ .NET 2003 enables developers to build entirely unmanaged Windows-based applications and components. The compiler is enhanced with several new and improved optimizations and capabilities, including Whole Program Optimization, the ability to generate optimized code for recent processor technologies (including the Intel Pentium 4), and the ability to better optimize for processors with Streaming SIMD Extension (SSE and SSE2) support. The latest multi-threading features for both compiler and processor are a big plus as mass-spring systems support parallel processing natively.

The Visual C++ .NET 2003 compiler is conformant with the ISO C++ language definition, and can easily builds modern C++ code and library sources. Visual C++ .NET 2003 provides wide range of libraries, including a fully ISO-compliant STL implementation (providing generic container classes and algorithms). This is very useful to our implementations as an efficient, error free data structure is highly required.

Fully integrated is the Visual Studio Debugger, an advanced tool that enables multi-language debugging, managed and unmanaged debugging, and remote debugging. Enhanced Edit and Continue features exist for unmanaged C++ code. The debugger also supports mini-dump technology, enabling developers to quickly identify and correct problems in deployed applications.

OpenGL was chosen as the rendering application programming interface (API) technology for this project. It is the environment of choice for developing portable, multi-platform interactive 2D and 3D graphics applications. OpenGL incorporates a broad set of rendering, texture mapping, special effects, and other advance visualization functions. OpenGL is supported in wide variety of popular desktop and workstation platforms, ensuring wide application deployment.

It is a matured technology with optimize driver supported by three dimensional hardware developers. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard. OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

OpenGL is well structured with an intuitive design and logical commands, similar concept to traditional structure programming paradigm (in this case, C language). Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features. Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

OpenGL does proved to be an invaluable rendering asset but it's not without it weakness, or lack of actual basic implementation features. It lacks camera navigation systems, user interface, math function, geometric processing and tools for manipulating rendered objects. For this project's cause, vertex picking, arcball rotation, dynamics computation and geometry discretization is required.

There are many available libraries and tools filling the gaps of OpenGL implementations. Such are 3d engines, utility toolkits, physics library, math library and computational geometry tools. Some interesting 3d engine which provides the some of the required features are OGRE (Object Oriented Rendering Engines) (http://www.ogre3d.org/), Irrlicht engine (http://irrlicht.sourceforge.net/) and G3D 3d engine (http://g3d-cpp.sourceforge.net/). Each of them works as a graphics toolkits by providing camera systems, advanced rendering systems (shaders, LOD, etc), user interface, object management and others useful stuffs with extensive documentation and community support. Some of them go a long way to provide a useful framework for physical based simulation by integrating with physics library. Such implementation is can be seen from OGRE addons, named nogredex (http://www.ogre3d.org/index.php?option=com_content&task=view&id=17&Itemid=70). Other toolkits worth mention is GLVU (http://www.cs.unc.edu/~walk/software/glvu/) which serves as a common platform for common graphical tasks. Novodex (http://www.ageia.com/novodex.html) and Meqon (http://www.meqon.com/) are two most impressive rigid body dynamics library that are available freely and heavily documented. Although these libraries doesn't explicitly provide and support deformable object functions, these libraries have optimized numerical integration function that is useful for mass-spring systems. Useful computational geometry tools used for discretizing the geometric mesh are tetgen (http://tetgen.berlios.de/) and gmesh (http://www.elysiun.com/~theeth/gmesh/). Both provide an easy way to construct tetrahedral from geometry input of various forms and shapes. Boost (http://www.boost.org/) and Numerical Recipe (http://www.nr.com/) are two great math libraries that are very useful for this project. Both provide mathematical

function that can be used for mass-spring systems. For solving large sparse matrices, there is Matrix Template Library (http://www.osl.iu.edu/research/mtl/). It is freely available, fast, stable and accurate to use for solving large or very large sparse matrices.

3.4 Testing methodologies

The method must be fully tested before it can be used by other users. The software is planned to be tested based on its performance, total memory footprint, visual acceptance and functionality and the correctness of the results.

Performance measurement is performed by recording the time used by specific task. Testing will be performed under various situations including higher geometry scenario. Processing time will be recorded for preprocessing geometry, selecting elements for deformations and performing deformation. For each task, memory usage is monitored.

Deformation is justified by its visual acceptance and comparing the result between virtual and real counterpart. This evaluation is performed for various simulated materials to ensure algorithm robustness.

For benchmarking purpose, testing is performed with different input data, different deformation method and different parameter settings.

CHAPTER IV

IMPLEMENTATION

4.1 Introduction

The simulation system was built using C/C++ languages in Windows platform using OpenGL. This chapter provides discussions including data preparation, various techniques implementations, algorithms and data structure.

4.2 Preparing data

The goal of data preparation is to have a tetrahedral mesh data, suitable for mass-spring system based deformable object simulation. The basic steps are preparing geometry data for tetrahedral mesh generation, generate tetrahedral mesh from the geometric data using tetrahedral mesh generation library, and preprocess the tetrahedral mesh data for simulation specific needs.

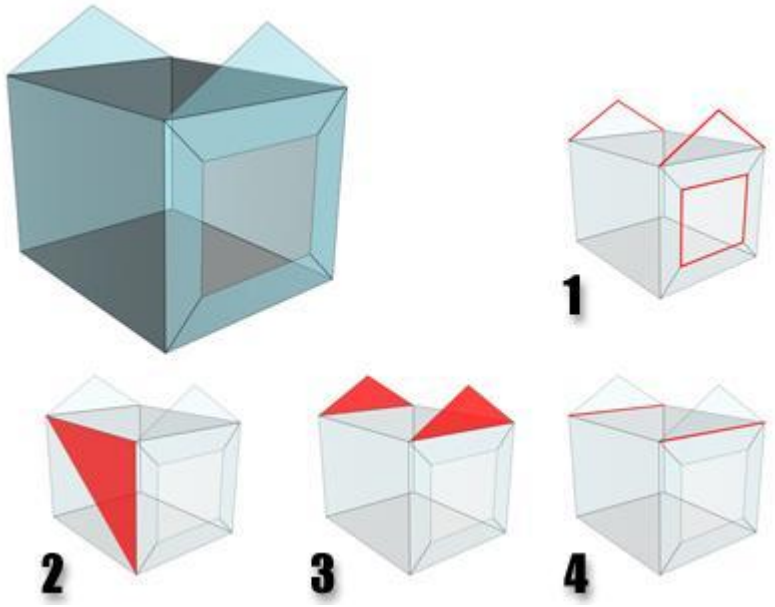


Figure 4.1 Stereolithography file format requirements. 1. No open edge. 2. No double face. 3. No spike. 4. No multiple edges. (Images from 3D Studio Max 7.0 Reference Manual)

There are two freely available tetrahedral generators that seem very suitable for the simulation systems needs, Tetgen (available at http://tetgen.berlios.de/index.html) and NETGEN (available at http://www.hpfem.jku.at/netgen/). Provided as low level library, Tetgen was chosen due to its good documentation, ease of use and the ability to refine the generated tetrahedral mesh.

Geometric data preparation is no trivial task. The geometry must have met various geometric criteria before tetrahedral mesh can be generated. The inability of Tetgen to automatically fix the geometry requires the use of third party tools to fix the geometry.

The geometric data must have these criteria (some of them are stereolithography format requirements):-

1. no open edge

2. no double face
3. no spike
4. no multiple edge
5. no orphan vertex
6. no collided face
7. consistent normals

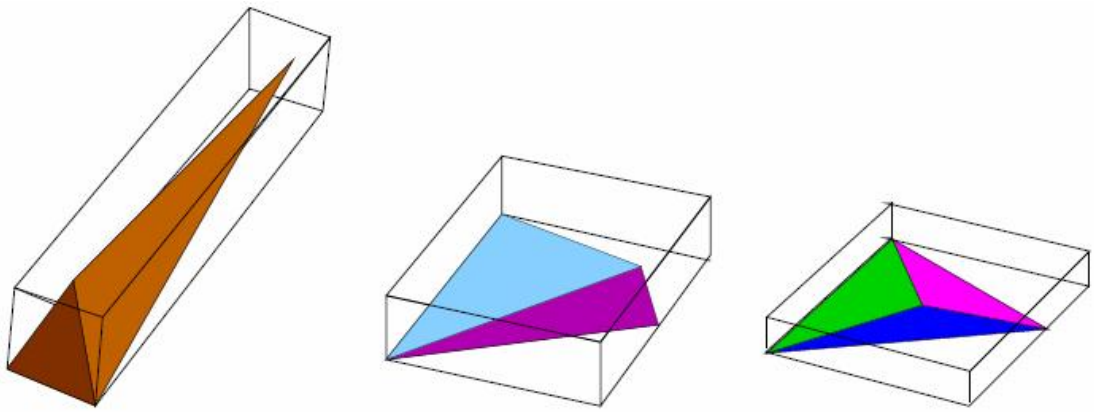


Figure 4.2 Example of tetrahedral with no quality enforcement. (Images from Tetgen 1.3 Manual)

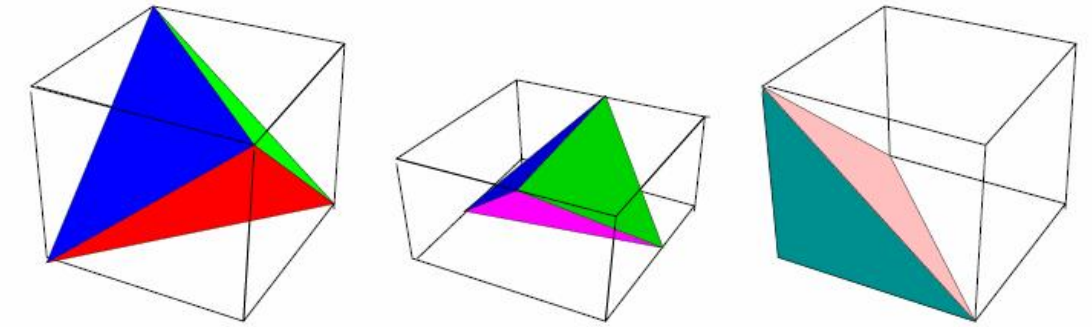


Figure 4.3 Example of tetrahedral with quality enforcement. (Images from Tetgen 1.3 Manual)

For geometry that didn't have these requirements, third party tools such as Floating Point Solution's MeshWorks 1.0, Okino Computer Graphics's PolyTrans for Max and NuGraf, Right Hemisphere's Deep Exploration CADTools plugins and discreet 3ds Max 7 can be used for fix these problems. Most of the problems can be fixed with 3D Studio Max 7.0 using its modifier tools such as STL check, cap holes, data exporters and vertex weld or by manual vertex fix.

When the geometric data is ready, the tetrahedral meshes are generated using Tetgen. Tetgen provides various settings to tweak the desired tetrahedral output. One of them is a tetrahedral quality constraint setting which is used to ensure radius-edge ratio greater than 2.0. Enforcing quality constraint will increase the number of vertices and tetrahedral immensely. Since most of sample geometric objects composed of high number of vertices, quality constraint is of no practical use as of current hardware processing power limitation. Thus, all sample geometric objects are created using no quality constraints.

Sample geometric objects are either freely downloaded from the internet or built using discreet 3D Studio Max 7.0. The objects are classified as simplified range scan, human organ, primitive objects, concave data, extreme data (extreme length, flat object, etc) and contains hole.

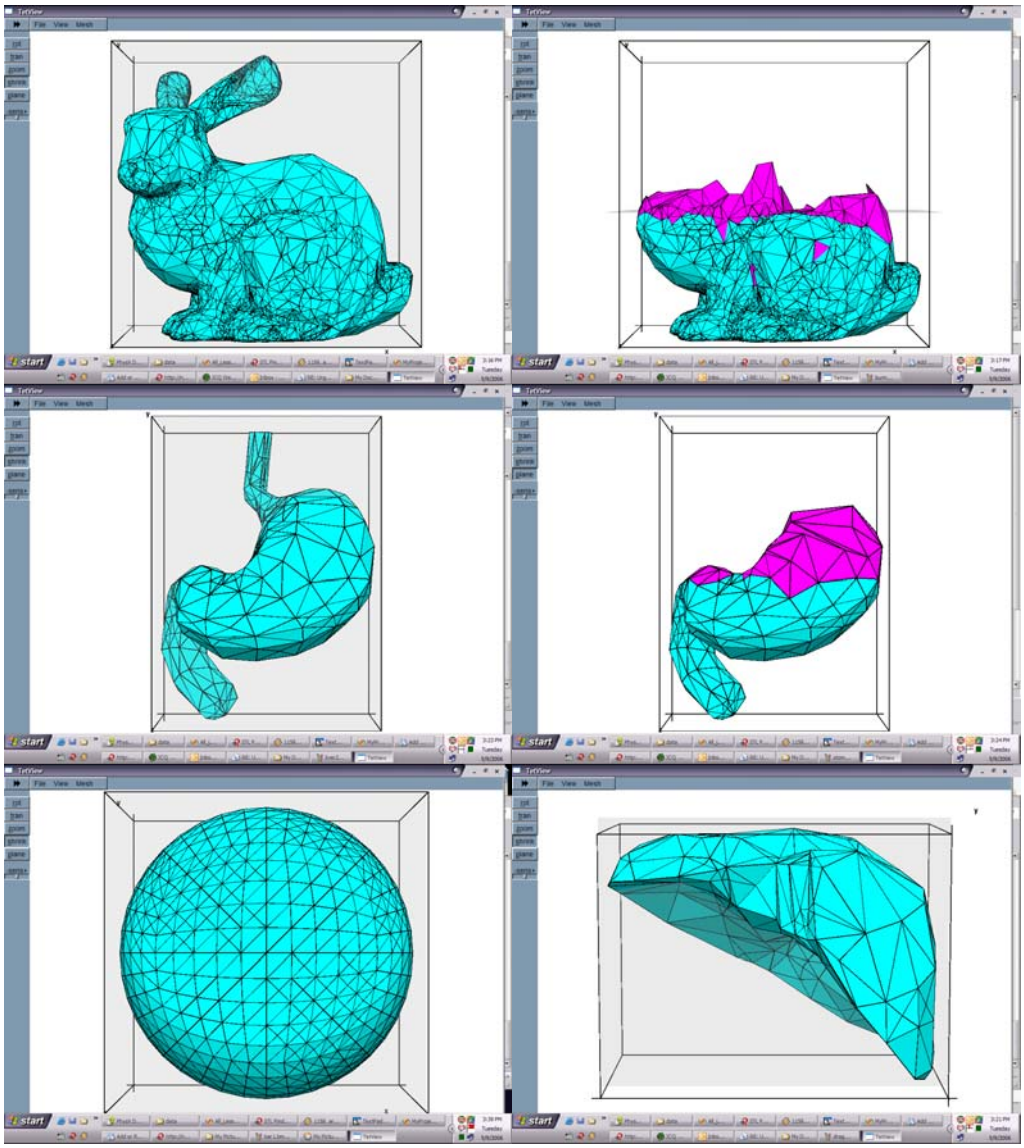


Figure 4.4 Some example of tetrahedral meshes viewed with Tetgen viewer. In top-left to top right order, the data are Stanford bunny, Stanford bunny internals, human stomach, human stomach internals, sphere and human liver. All of them are freely available on the internet except for sphere which is generated using discreet 3D Studio Max 7.0. Screenshots were taken using Tetgen Viewer.

4.3 Building the framework

Building an optimal data management system for simulation comprises of high number of vertices poses a big performance problem. The same goes for simple mass-spring solver, which poses visual artifact problem and stability problem. Instead, physics library is used for its mass-spring simulation solver and its data management system. This makes implementation easier and multiple experiments can be performed quickly. Other features of physics library such as visualization tools, profiling and collision detection makes it very interesting to use. Since this research's main interest is an additional algorithm to reduce deformation processing time and not building a complete mass-spring systems, using physics library allows the comparison of classical mass-spring deformation with or without the additional algorithm in a very fair and unbiased way. Also, using closed source library proved that the algorithm is general enough that it can be used in physics library with little knowledge of underlying library structure. The new algorithm, denoted as optimization algorithm, main purpose is to select small sets of nodes to be solve by physics library. The optimization algorithm takes input from collision response and provides small sets of nodes to be deformed by simulation system.

Figure 4.5 Conceptual flow of common physical simulation after inserting optimization algorithm.

There are various physics library that is suitable for this cause ranging from open source, closed source, commercial and non-commercial. The physics libraries that are freely available for non-commercial use are:

1. Open Dynamics Engine
2. Newton dynamics
3. Tokamak physics library

4. True Axis physic library
5. Meqon Dynamics
6. Ageia Physx

Ageia Physx was chosen due to multiple reasons. Comparisons are loosely based on technology demo provided by each physics library. In terms of number of nodes per simulation scene, Physx are able to sustain the highest number of frame rate for scene with high number of nodes. Couple with a physic processing unit (PPU) card, the simulation performance would be much higher. As of current writing, PPU is exclusively supported by Ageia Physx. Other great aspect of Physx is that it has good documentations, technical support, active community and matured code.

4.4 Performance issues

Each simulation frame consists of collision detection and response, deformation processing, and rendering. For least acceptable visuals, simulations must run at least 24 frames per second. This frame rate leaves 42 milliseconds to prepare for each simulation frame. For best possible visuals, simulations must run more than 75 frames per seconds. This requires 13 milliseconds or less per simulation frame preparation. Under these constraints and depending on the application requirements, deformation processing may contain 50% or less processing time for each frame. For the optimization algorithm to be effective, it must use very small processing time. This includes iterations, caching, searching and mathematical operations.

4.5 General strategy

For large number of nodes, the deformation systems usually suffer even with large memory. Assuming a processing system with 1 GB of RAM, the deformation processing remains a bottleneck while the system RAM utilizations are no where near maximum capacity. The same goes for rendering capacity. Current graphic cards have huge rendering pipeline. It would be beneficial to utilize this feature by featuring full geometry complexity. Elastic deformable object which does not change topology, gives advantage in terms of data structure and geometric reconstruction compared to non-elastic deformable objects. Fast static data structure and preprocessed search result can increase run time performance. Since simulation of deformable objects requires heavy computations, it would be wise to use cheap computation for its optimizations in order to bring more room for deformation processing. Computation cost can be reduce by using simple mathematical function, reuse computed data, preprocess, putting a computation or time cap for each frame or using smaller sized variables. Using some mathematical function known to have high computation cost includes division and square roots should be avoided if possible.

4.6 Building algorithm template

Algorithm template is a general algorithm that will be the basis of algorithm refinement and testing. The algorithm output will be small sets of active nodes which will be process for deformation. The algorithm strategy would be:

1. Activate nodes that are near the area where concentrated loads are applied.
2. Simulate spring physics for every active node.
3. Deactivate nodes that reach its equilibrium state.
4. Inactive node with active node neighbor will act as spring constraint between the two.

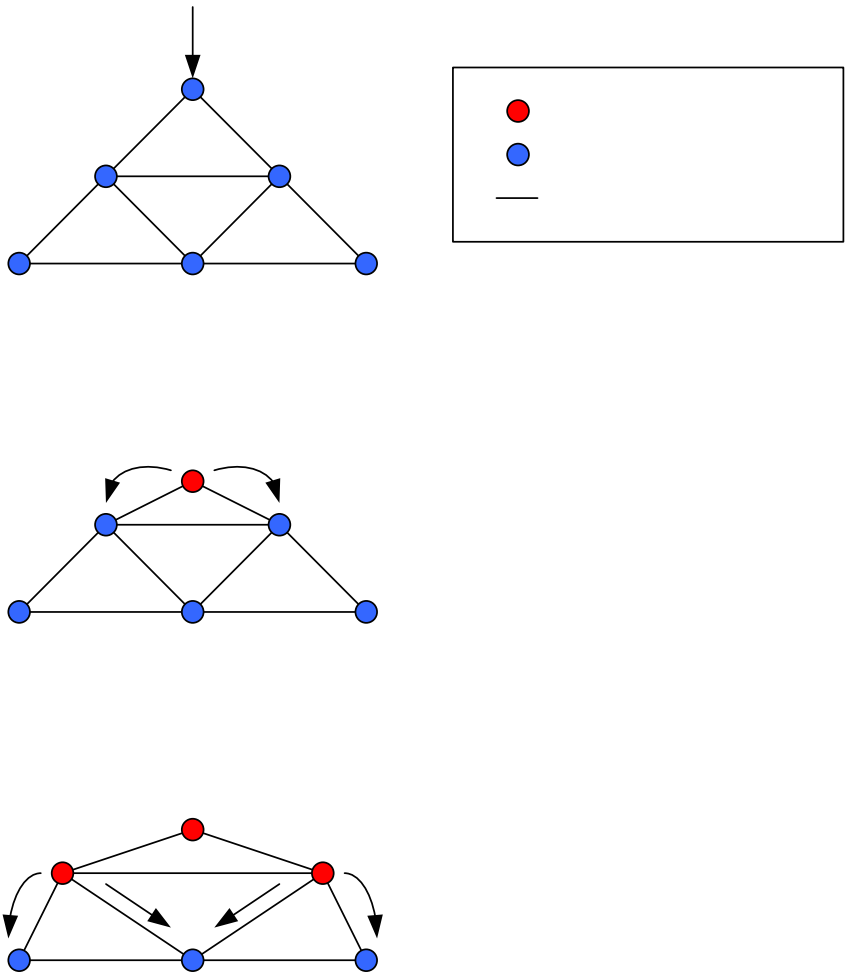


Figure 4.6 Example of activation systems in 2D. (1) Concentrated loads are applied to a node. (2) When the node reaches its non-equilibrium state, it will activate its neighbor. (3) The activation process continues until the node reaches its equilibrium state. Inactive nodes will act as constraint. Active nodes reaching equilibrium state will be deactivated.

(1)

4.7 Defining non-equilibrium state

An object is in equilibrium if the resultant of the system of forces acting on it has zero magnitude. In other words, the object is at rest. In mass-spring systems, the equilibrium state is a state where total accumulated forces on node have zero magnitude. If total accumulated forces on node have non-zero magnitude, the node is considered in non-equilibrium state. Total accumulated forces are the sum of all forces acting on the node. Forces acting on node can be either internal forces or external forces. Internal forces are forces from springs acting on the node. External forces are forces derived from collision response or manually applied forces. Other forces can be additional forces for preserving surface and volume. It is safe to say that nodes will be displaced when its total accumulated forces are non-zero ($\mathbf{F} \neq 0$). Based on equilibrium state, the updated algorithm would be:

1. Deactivate nodes with zero total accumulated forces
2. Activate collided and selected nodes
3. Activate active node's neighbor with non-zero total accumulated forces
4. Start simulation

Testing accumulated forces against certain threshold would be better as the node would be easier to deactivate, harder to activate and threshold provides a means of simulation scaling. The algorithm with accumulated forces threshold would be:

1. Deactivate nodes if total accumulated forces are less than threshold
2. Activate collided and selected nodes
3. Activate active node's neighbor if total accumulated forces are less than threshold
4. Start simulation

Based on Figure 4.5, optimization algorithm takes input from collision response. There are three types of collision response. Constraint based collision response modifies the position of the interpenetrated object directly by giving new projected position. Impulse based collision response use instantaneous impulses or

changes in velocity to prevent objects from interpenetrating by modifying the first derivative of the positions (i.e. velocities). Penalty based collision response uses spring to pull the object out of collision state by modifying the second derivative of the positions (i.e. accelerations). In other words, constraint based collision response provides a new position, **s**, impulse based method provides a new velocity, **v** and penalty based collision response provides a new acceleration, **a**. Instead of computing total accumulated forces for collided object to query the equilibrium status, it would be better to use values from collision response. But first, the relation between forces and collision response output must be defined. In other words, are there any physical quantities that can reflect changes to **F**?

From physics text book,

$$\mathbf{F = ma}$$

Where **F** is force, **m** is mass and **a** is acceleration. Above equation proved that a change in **a** will change **F** as long as **m** not changing. To find velocity relationship,

$$\mathbf{F = m(dv/dt)}$$

$$\mathbf{dv = v_1 - v_0}$$

$$\mathbf{v = ds/dt}$$

Where **dv** is delta velocity, **dt** is delta time and **ds** is delta position. A change in velocity, **v** will change **F**. For position **s**,

$$\mathbf{ds = s_1 - s_0}$$

Thus, equilibrium state can be defined from value provided by collision response. In other words, equilibrium state can be defined as changes in position, velocity or acceleration.

The easiest way to define a non equilibrium state is by using position, **s**.

4.8 Relative distance from node to its neighbor as equilibrium state

Activation test is performed by finding whether the node is in non equilibrium state. Non equilibrium state is defined as a change in distance between current node and its neighbor. Given $\mathbf{s_r}$ as node position, $\mathbf{s_n}$ as neighbor position and $\mathbf{d_{cache}}$ as previous frame distance between node position and neighbor position. A node will be in non equilibrium state if current distance from node to its neighbor, $|\mathbf{s_r}-\mathbf{s_n}|$, is not equal to previous distance from node to the neighbor, $\mathbf{d_{cache}}$. If the node is in non equilibrium state, activate its neighbor. This activation test is performed for every neighbor. Adding threshold would modify the non equilibrium test to if current distance from node to its neighbor, $|\mathbf{s_r}-\mathbf{s_n}|$, is more than or less than previous distance from node to the neighbor, $\mathbf{d_{cache,}}$ multiplied with threshold.

Deactivation test is performed by determining whether the node is in equilibrium state. Since deactivation does not relate to neighbors (deactivation deactivates nodes, not its neighbors), equilibrium state is define as distance from current node position to previous node position. Given $\mathbf{s_r}$ as current node position, $\mathbf{s_{cache}}$ as previous node position, the test would be, if distance from current node position, $\mathbf{s_r}$, to previous node position, $\mathbf{s_{cache}}$, equals to zero, the node is deactivated. After adding threshold, the test would be, if distance from current node position, $\mathbf{s_r}$, to previous node position, $\mathbf{s_{cache}}$, less than threshold, the node is deactivated.

For activation test, each neighbor requires 1 distance cache from node to neighbor, totaling 3 distance caches for 3 neighbors. For deactivation test, previous node position is cached. This method is evaluated using best case scenario where activation and deactivation occurs in a tetrahedral (one active node with three neighbors). The cost for activation test for a single neighbor is 2 additions, 3 subtractions, 5 multiplications, 2 relational and 1 square roots. For 3 neighbors, the total cost is 6 additions, 9 subtractions, 15 multiplications, 6 relational and 3 square roots. Due to threshold multiplications, square root operation cannot be simplified by using squared value. For node deactivation test, the cost for single node deactivation is 2 additions, 3 subtractions, 3 multiplications, 1 relational and 1 square root. Square

root can be eliminated by using squared value since it is used only in relational operation.

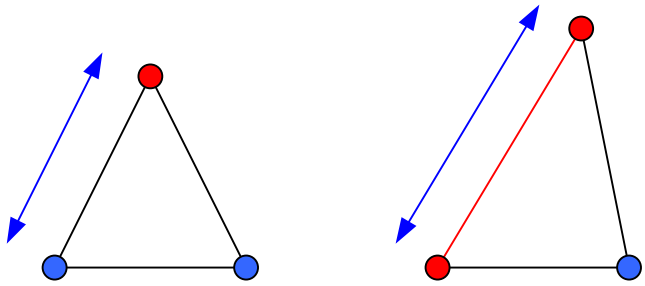


Figure 4.7 Activation test. If ($|\mathbf{s}_{r(t1)}-\mathbf{s}_{n(t1)}| > \mathbf{d}_{cache}$*threshold || $|\mathbf{s}_{r(t1)}-\mathbf{s}_{n(t1)}| <$ $\mathbf{d}_{cache}$*threshold), activates its neighbor, $\mathbf{s}_{n(t1)}$. In other words, if current distance, $\mathbf{d}_{current}$ is more than $\mathbf{d}_{cache}$ multiplied with threshold or current distance, $\mathbf{d}_{current}$ is less than $\mathbf{d}_{cache}$ multiplied with threshold, activate the neighbor, $\mathbf{s}_{n}$.

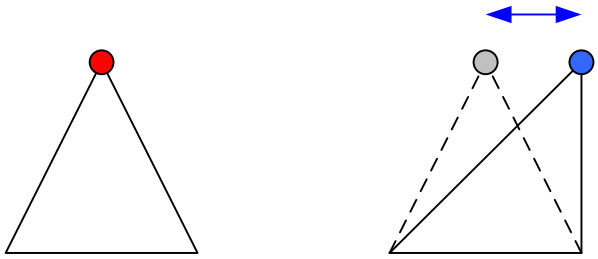


Figure 4.8 Deactivation test. If ($|\mathbf{s}_{r(t1)}-\mathbf{s}_{r(t0)}|$ > threshold), deactivate itself, $\mathbf{s}_{r}$. In other words, if current distance, $\mathbf{d}_{current}$ is less than threshold, deactivate $\mathbf{s}_{r}$.

Table 4.1 The cost for activation and deactivation test for best case scenario (1 node and 3 neighbors).

	Activation	Deactivation	Total
Addition	6	2	8
Subtraction	9	3	12
Multiplication	15	3	18
Relational	6	1	7
Square root	3	0	3
Cache	3	1	4

From the best case scenario, the optimization algorithm is quite fast to be practically used using current hardware. With 3 square roots and 4 data caches being the most expensive, it would be better if both of that can be reduce or eliminated.

4.9 Distance from current node position to previous node position as equilibrium state

Non equilibrium state is defined as the distance from current node position to previous node position is not zero. Given $\mathbf{s}_{\mathbf{r(t1)}}$ as current node position, $\mathbf{s}_{\mathbf{r(t0)}}$ as previous node position, the node is in non equilibrium state if distance from $\mathbf{s}_{\mathbf{r(t0)}}$ to $\mathbf{s}_{\mathbf{r(t1)}}$ is greater than zero. Since the test is performed per node with no relation to its neighbor as compared to previous method, non equilibrium node will activate its entire neighbor. Adding threshold would modify the non equilibrium test to if distance from $\mathbf{s}_{\mathbf{r(t0)}}$ to $\mathbf{s}_{\mathbf{r(t1)}}$ is greater than threshold. Deactivation test is exactly the same from previous method (see Figure 4.8).

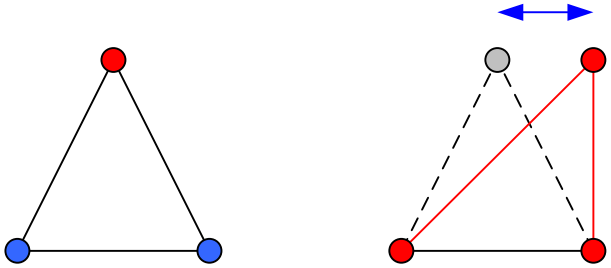


Figure 4.9 Activation test. If ($|\mathbf{s}_{r(t1)}-\mathbf{s}_{r(t0)}|$ > threshold). In other words, if current distance, $\mathbf{d}_{current}$ is greater than threshold, activate all neighbors. Deactivation test is exactly the same from previous method (see Figure 4.8).

Cache of previous node position is used for both activation test and deactivation test. One distance calculation is needed for each activation test and deactivation test. The cost of both activation and deactivation test for best case scenario is 2 additions, 3 subtractions, 3 multiplications, 1 relational and 1 square root. Since square root value is used in relational operations, it is possible to use squared value instead. Thus the total cost for both activation and deactivation test is 4 additions, 6 subtractions, 3 multiplications and 1 relational.

Table 4.2 The cost for activation and deactivation test for best case scenario (1 node and 3 neighbors).

	Activation	Deactivation	Total
Addition	2	2	4
Subtraction	3	3	6
Multiplication	3	3	6
Relational	1	1	2
Square root	0	0	0
Cache	1	1(shared)	1

4.10 Node's linear velocity as equilibrium state

Instead of using positions as basis, the third method defines the non equilibrium state as non-zero linear velocity. Given $\mathbf{s}_{\mathbf{r(t1)}}$ as current node position, $\mathbf{s}_{\mathbf{r(t0)}}$ as previous node position, the node is in non equilibrium state if current node linear velocity is not zero. If current node is in non-equilibrium state, activates its entire neighbor. Adding threshold would modify the non equilibrium test to if current node linear velocity greater than or less than threshold. To simplify this relational operation, magnitude of linear velocities is used instead of vector of linear velocities. This would modify the non equilibrium test to if current node linear velocity magnitude greater than threshold. Computing accurate magnitude requires square root operations. A more simple method would be to measure the magnitude per axis. The problem with this simple method is that the same magnitude does not always pass the non equilibrium test when the node experiences multiple velocities from different axis. An example is shown in Figure 4.10. Common method for finding magnitude requires one square root operation. But, since the magnitude is used in relational operation only, it is safe to use squared value. The deactivation test using linear velocity magnitude would be if current node linear velocity magnitude lesser than threshold, deactivate itself. Apart from the difference in relational operator used, both activation test and deactivation test are perfectly the same.

For best case scenario, activation test and deactivation test requires 2 additions, 3 multiplications and 1 relational operation. Thus, the total cost for both activation and deactivation test is 4 additions, 6 multiplications and 2 relational operations. Successful activation test will result in activation of all current node neighbors. Note that using linear velocity results is no cache and square roots in its testing procedure unlike 2 previous methods.

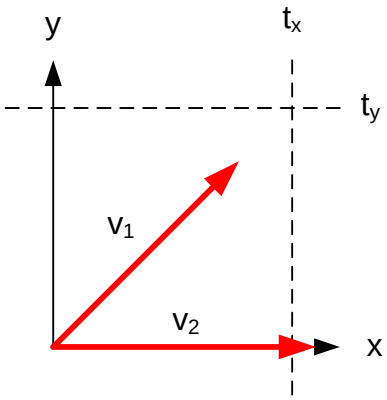


Figure 4.10 Inconsistencies of using simple magnitude measuring by using per axis test. $\mathbf{v_1}$ and $\mathbf{v_2}$ are linear velocities with the same magnitude, $\mathbf{t_x}$ and $\mathbf{t_y}$ are axis threshold and **x** and **y** are axis. $\mathbf{v_2}$ passed the non equilibrium test while $\mathbf{v_1}$ failed the non equilibrium test even when both share the same magnitude.

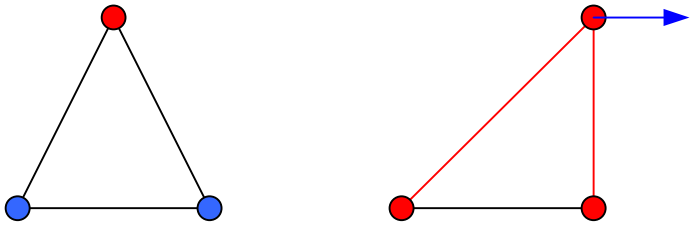


Figure 4.11 Activation test. If ($|\mathbf{s_{v(t1)}}|$ > threshold), activate all its neighbors. In other words, if current node velocity, $\mathbf{s_{v(t1)}}$ is greater than threshold, activate all its neighbors.

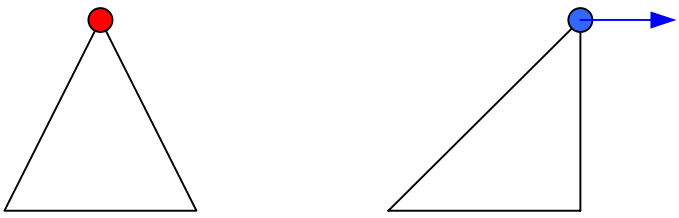


Figure 4.12 Deactivation test. If ($|\mathbf{s_{v(t1)}}|$ < threshold), deactivate itself, $\mathbf{s_r}$. In other words, if current node velocity, $\mathbf{s_{v(t1)}}$ is lesser than threshold, deactivate itself, $\mathbf{s_r}$.

Table 4.3 The cost for activation and deactivation test for best case scenario (1 node and 3 neighbors).

	Activation	Deactivation	Total
Addition	2	2	4
Subtraction	0	0	0
Multiplication	3	3	6
Relational	1	1	2
Square root	0	0	0
Cache	0	0	0

4.11 Algorithm

The final optimization algorithm in simplest form using velocity as non equilibrium definition would be:

```
inline void optimizeDeformation(pointer to softbody, pointer to list
of collided nodes)
{
  //deactivation
  for all activated nodes
  {
    if current time - activation time > active time threshold
    {
      if squared linear velocity magnitude < squared linear velocity
  magnitude threshold
      {
        deactivate node
      }
    }
  }

  //activation from collision
  activate all collided nodes
  activation time = current time

  //neighbor activation
  for all activated nodes
  {
    if current time - neighbor activation time > neighbor activation
  time threshold
    {
      if squared linear velocity magnitude > squared linear velocity
  magnitude threshold
      {
        neighbor activation time = current time
        for all neighbors
        {
          activate nodes
          activation time = current time
        }
      }
    }
  }
}
```

4.12 Data structure

To achieve full performance, it is essential to prepare the required data for run time efficiency. The deformable object does not experience topological changes which make it suitable to prepare static lists of neighbors or other required data in pre processing. Additional node data is linked with node in the physics systems.

The additional node data consist of:

1. List of neighbors
2. Activation status(active, inactive, constraint)
3. Activation time
4. Neighbor activation time

Other data includes

1. List of faces (for vertex normals computation)

A dynamic data structure is needed to keep track of the activated nodes by storing list of pointers to activated nodes. Dynamic data structure is used to minimize the time to traverse active nodes. Other required values are squared linear velocity magnitude threshold, active time threshold and neighbor activation time threshold.

4.13 Conclusion

This chapter provides details on the making of the optimization algorithm and its justifications.

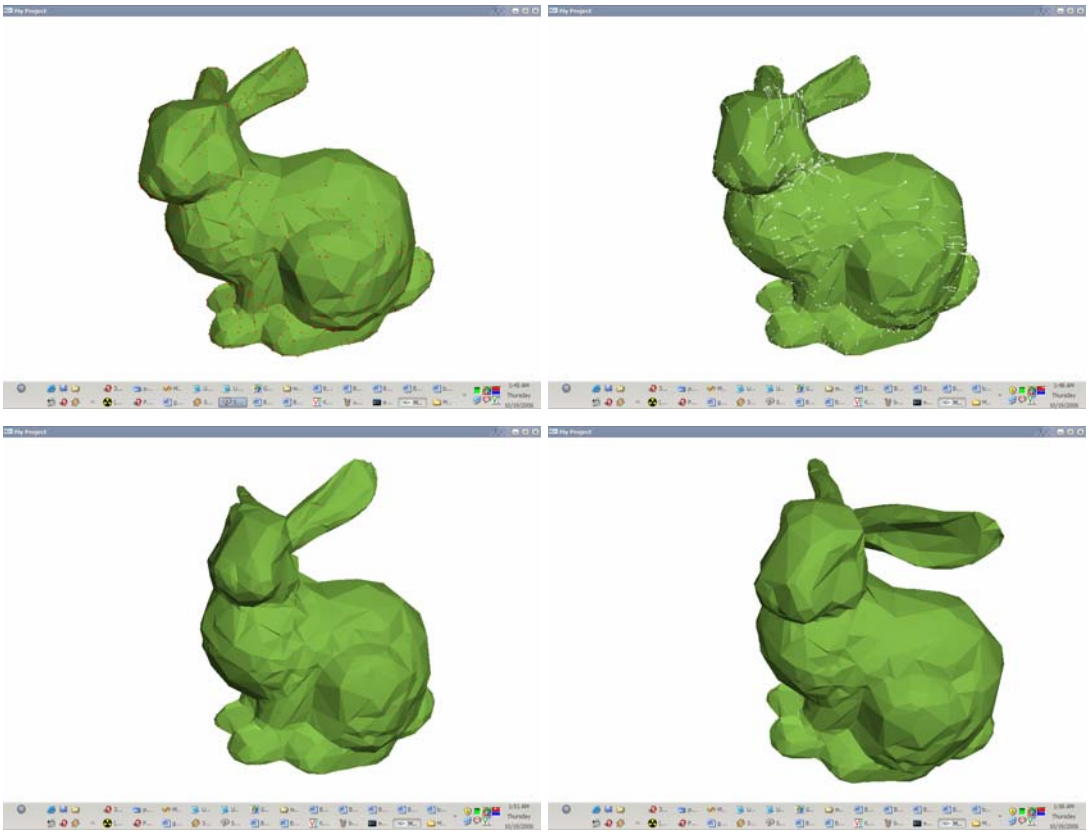


Figure 4.13 Example deformations of Stanford bunny data.(Top left image is the undeform pose)

The next chapter will provide results, analysis and discussion of the optimization algorithm.

CHAPTER V

ANALYSIS

To better understand the optimization algorithm benefits and pitfalls, series of benchmarks are performed.

5.1 Introduction

This chapter performs series of benchmarks to analytically evaluate the performance of optimization algorithms. First, comparison of arithmetic operation for various methods of activation and deactivation are presented. Then, benchmark methodology and discussions followed. After that, discussions continue to various issues concerning the optimization algorithms.

5.2 Evaluations of algorithms

The first optimization algorithm uses distance as its basis and its activation test relies on each neighbor position. This would make the computation cost increase with increasing number of neighbors for each node. Its advantages though are the number of active node is maintained at minimum for each step due to activation test will only activate single node per successful activation test. For best case scenario, the first optimization algorithm requires three square roots and for every node, the total required cache is the total number of neighbors.

The second optimization algorithm requires less computation cost than the first optimization algorithm. It also doesn't require any square roots computations. Since the second optimization algorithm doesn't rely on its neighbors, the total cache per node is only one. The drawback of the second optimization algorithm is that once the activation test succeeded, it will activate all neighbors of current node. Although this doesn't be a problem if the application requires more accurate behavior, it does requires more physics computation compared to the first optimization algorithm. The first and second optimization algorithm operates optimally with constraint based collision response (where it provides new position) coupled with Taylor series based integrator (position based integrator) due to the algorithm reliance of position. Otherwise, for every activation and deactivation testing, data must be converted.

The third optimization algorithm has the lowest computation cost compared to the two previous optimization algorithms. Like the second optimization algorithm, the third optimization algorithm doesn't require any square root. Unlike both previous optimization algorithms, the third optimization algorithm doesn't require any cache. Successful activation test will activate all neighbor of current node allowing better physical behavior at the cost of more node activated. Unlike previous two optimization algorithm, this optimization algorithm will operate optimally for impulse based collision response (where it provides new velocities) couple with Verlet based integrator (velocity based integrator). Current simulation system uses

Physx library which employ an impulse based collision response and Verlet integrator system. Other method for defining the non equilibrium state using acceleration is not tested due to the requirement of data conversion.

Table 5.1 The cost of activation and deactivation test comparison for best case scenario (1 node and 3 neighbors).

	Relative distance from node to its neighbor as equilibrium state	Distance from current node position to previous node position as equilibrium state	Node's linear velocity as equilibrium state
Addition	8	4	4
Subtraction	12	6	0
Multiplication	18	6	6
Relational	7	2	2
Square root	3	0	0
Cache	4	1	0

5.3 Results and benchmarks

To further evaluate the algorithm, a series of benchmark is performed. First, the goals of benchmark are outline. Then, each method of benchmark is detailed. After that, hardware, software and domain specifications are given. The details of input data follow after that.

There are two types of benchmark. The first benchmark compares the optimization algorithm with other algorithm. The second benchmark deals with different optimization algorithm settings.

5.3.1 Goals

The benchmarks are performed based on these goals:

1. To prove that the optimization algorithm can reduce the deformation processing cost especially for high polygonal object.
2. To analyze the effect upon having higher resolution data from low resolution data performance wise.
3. To prove that the optimization overhead cost is very low.
4. To prove that the optimizations algorithm as a highly scalable method.
5. To prove that the optimization algorithm can dynamically scale the deformation area as required.

5.3.2 Benchmarking method

To perform the benchmarking process, series of data is captured during application execution for prefix set of time. Captured data are frames per second, optimization overhead cost, physic library computation cost and total number of active nodes. Multiple input data are used the benchmarking procedure ranging from different resolution of Stanford bunny and perfectly symmetrical icosahedrons sphere. For every object, the same simulation is performed during 30 seconds of real time execution. In the first 10 seconds, no external force is applied to the deformable objects to evaluate the result of best case scenario of the deformable object. The next 20 seconds apply large external force to the deformable object in order to simulated real world usage. The force is constantly applied but the direction is changing for every 1 second. During these 30 seconds, total number of frames rendered per second

is captured in 1 second interval. To eliminate bias and inaccuracy, optimization overhead cost, physic library computation cost and total number of active nodes are captured using other means of timing. Instead of using real time duration, simulation time with duration that is incremented based on fixed time-step per every frame is used. For every one second of simulation time, data is captured for other benchmarking data except fps. Simulation time step is 0.02 seconds (50 time step for 1 second).

The application is customized so that it will not jeopardize the accuracy of captured data. Multi threading feature is disabled, no collision detection is performed, fixed simulation time-step, simple flat surface rendering with hardware normalization and no special optimization method from the physic library is used.

5.3.3 Specifications

The simulation was performed on Intel based PC with 3.0 GHz processor and NVIDIA GeForce FX 5200 3d accelerator with 1gig 400 MHz DDR2 RAM. Display driver version 91.47 is used. The prototype was implemented in C++, uses OpenGL for rendering and uses Ageia PhysX's mass spring system for deformation processing. At current time of writing, the latest available version of Ageia PhysX is 2.5.1.

5.3.4 Input data

There are two main data used in the benchmarking procedure; different resolution of Stanford bunny and perfectly symmetrical icosahedrons sphere. Stanford bunny is chosen not because it's a popular geometric mesh used in computer graphics research, but because it provide various necessary features in a single package. It is non-convex and have uniform distribution of vertices, two long thin ears, complex creased area near front leg and smooth surfaces in the spine area. The Stanford bunny is reconstructed into various different resolutions to measure the performance hit going from low resolution data to high resolution data. The three resolutions Stanford bunnies are named as bunny100, bunny500 and bunny1000. The other input data is a sphere. A sphere provides different properties compared to Stanford bunny. An icosahedrons sphere is nearly perfect symmetry, convex and all edges have nearly similar length. The primary purpose of the sphere is to evaluate the effect of different optimization algorithm settings. The sphere data is named as icosa12.

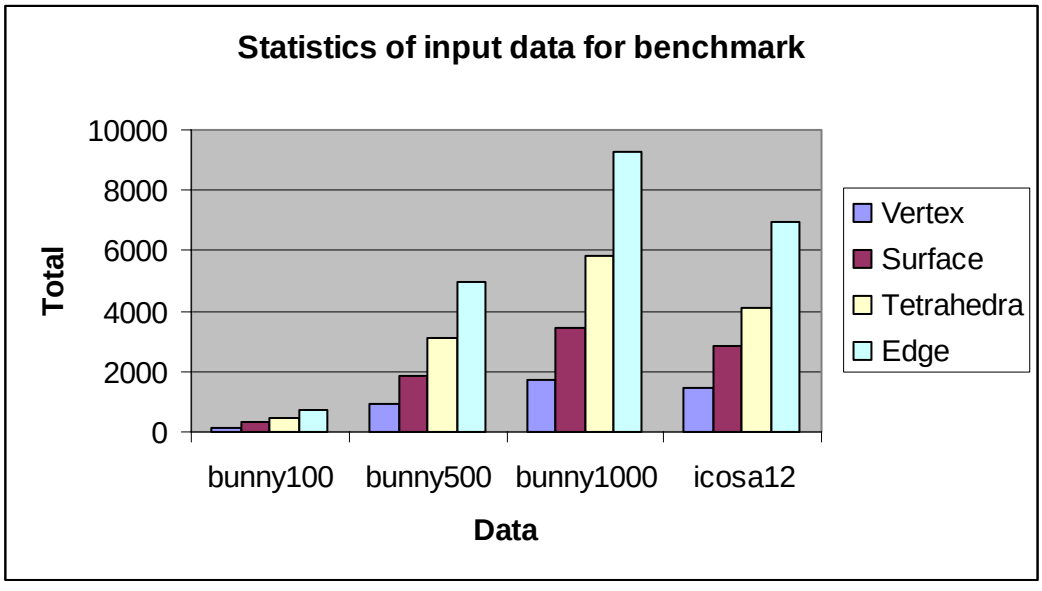


Figure 5.1 The statistic comparison of benchmark input data.

Table 5.2 Details of input data.

	bunny100	bunny500	bunny1000	icosa12
Vertex	159	929	1729	1442
Surface	314	1854	3454	2880
Tetrahedra	441	3100	5818	4098
Edge	756	4955	9273	6979

The algorithm were also tested against a wide range of geometric mesh having properties such as convex, non-convex, thin object, long object, uniform object, non-uniform object and object containing hole. The main purposes of these data are to detect odd behavior, artifacts and bug tracking. Since the evaluation is purely subjective, no discrete result and conclusion are made based on these results.

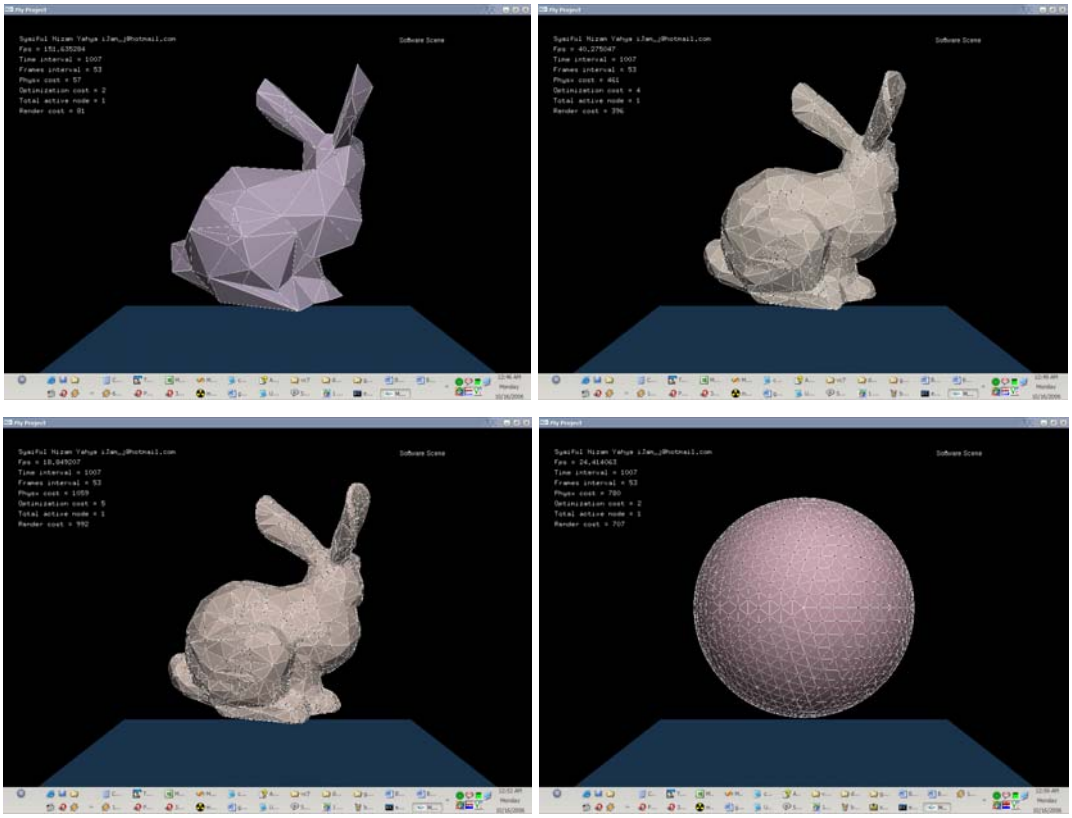


Figure 5.2 Input data for benchmark are bunny100 (top left), bunny500 (top right), bunny1000 (bottom left) and icosa12 (bottom right).

5.3.5 Benchmarks against other method

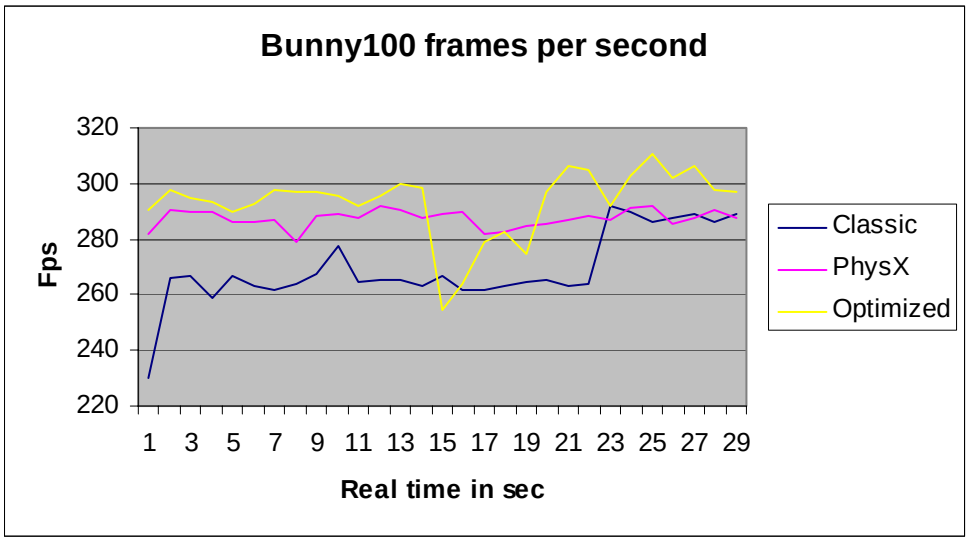
The first series of benchmarks are performed to compare performance and computation cost of three different deformation methods for three different resolutions of input data. The first method is the classical mass spring method. It is the standard no optimization mass spring systems. The second method is Ageia PhysX optimized method. Although Ageia PhysX was originally engineered for rigid body simulation, it provides various features that can be use in deformable simulation in order to gain more performance. Since the underlying method of Ageia PhysX optimization is a trade secret, it will not be discuss further. Only default settings for Ageia PhysX are used. The final method is the optimization method.

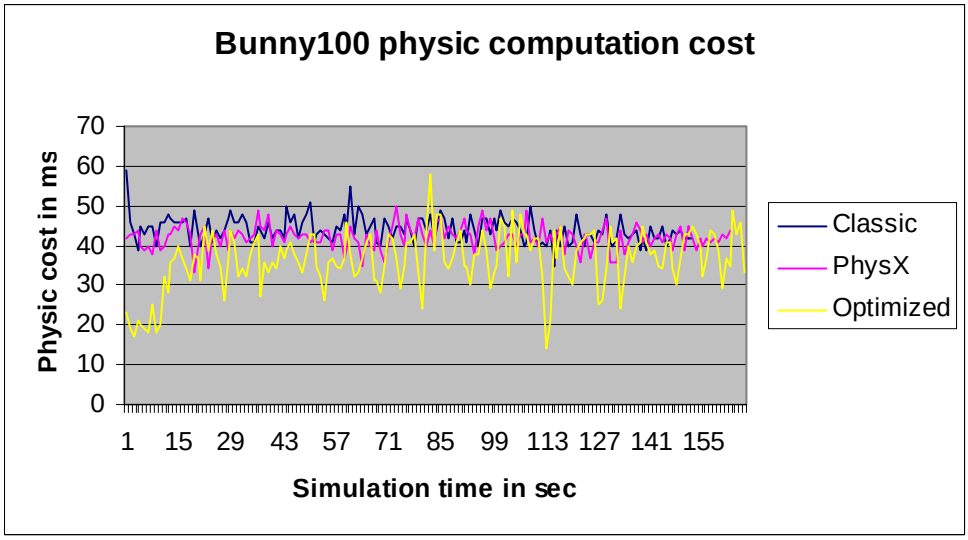
The default optimization algorithm settings are:

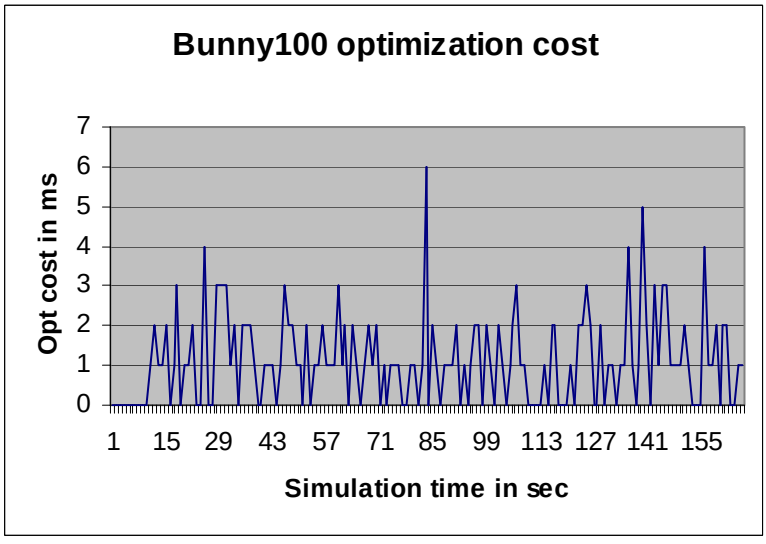
1. Squared linear velocity magnitude threshold = 10 unit
2. Active time threshold = 2000 ms
3. Neighbor activation time threshold = 2000 ms

There’s no real unit in the simulation system. But to give a sense of proportional, deformable object is scaled to fit in a cube with 800*800*800 simulation unit.

For each benchmarking process, 4 types of information are captured; frames per second, physic computation cost, optimization cost (optimization method only) and total number of active nodes (optimization method only). The first benchmark is captured using bunny100 input data.


Bunny100 frames per second
Fps
320
300
280
260
240
220
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
Real time in sec
Classic
PhysX
Optimized


Bunny100 physic computation cost
Physic cost in ms
70
60
50
40
30
20
10
0
1 15 29 43 57 71 85 99 113 127 141 155
Simulation time in sec
Classic
PhysX
Optimized


Bunny100 optimization cost
Opt cost in ms
7
6
5
4
3
2
1
0
1 15 29 43 57 71 85 99 113 127 141 155
Simulation time in sec

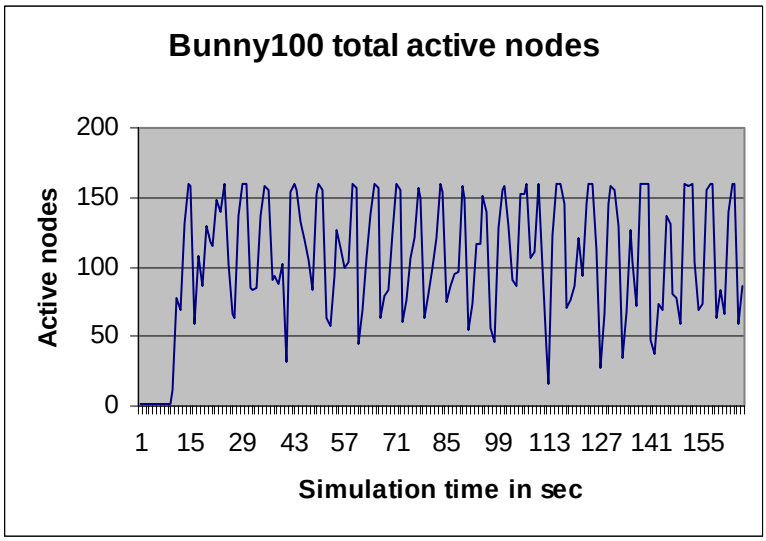
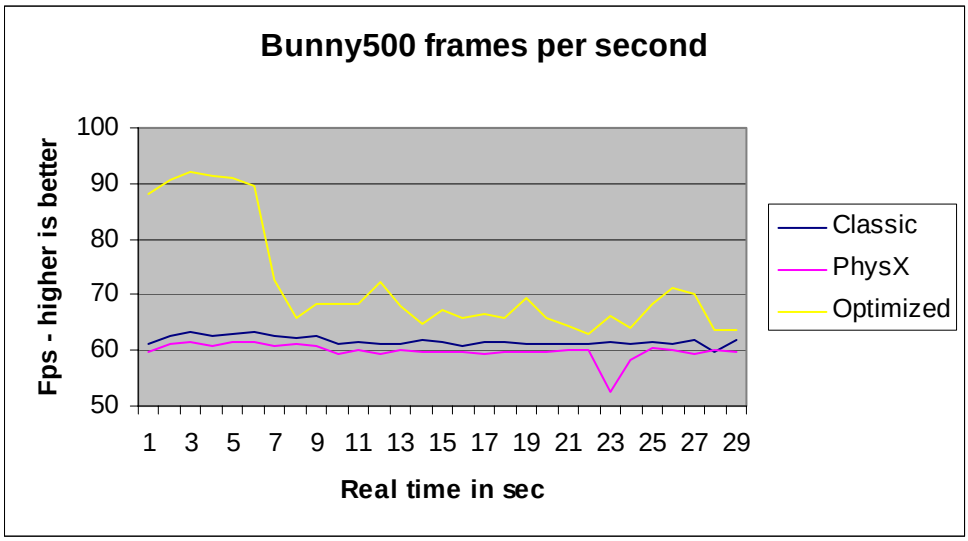
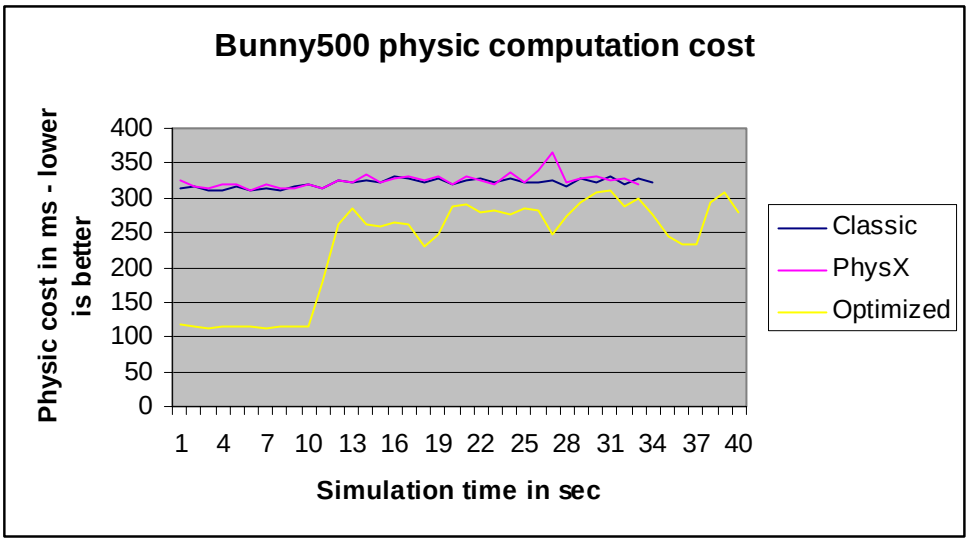
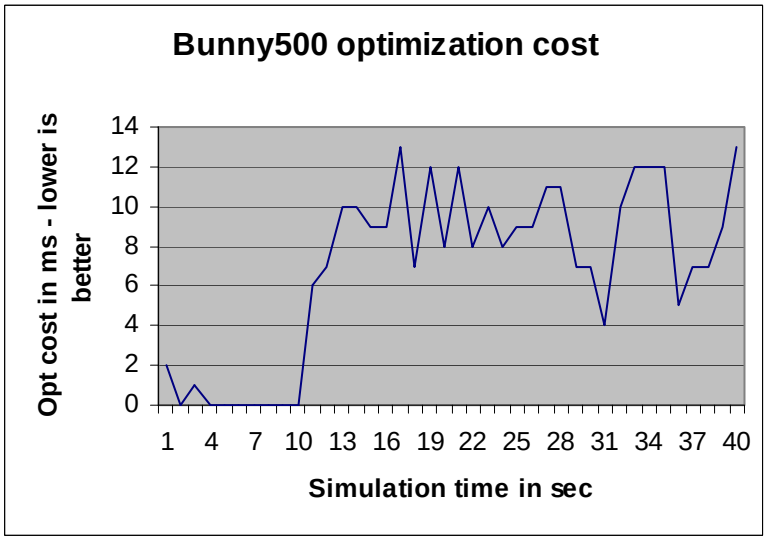


Figure 5.3 Benchmark charts for bunny100.

From Figure 5.3, deformable simulation using optimization method gain more frames per second compared to the other 2 methods. The second chart shows that optimization method does reduce physic computation cost especially during the first 10 second of simulation time where no external force is applied to the deformable object. Since the number of nodes is particularly low, the third chart shows that for around 150 nodes, the worst case scenario (where all nodes are activated), the maximum cost for optimization method is 6 ms. Meanwhile, for best case scenario simulated in the first 10 simulation seconds is less than 1 ms. The final chart shows that almost all nodes are active during benchmarking time. Overall, the bunny100 input data is too forgiving. With 300 frames per seconds averaged, it is unclear where the bottleneck actually is.


Bunny500 frames per second
Fps - higher is better
100
90
80
70
60
50
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
Real time in sec
Classic
PhysX
Optimized


Bunny500 physic computation cost
Physic cost in ms - lower is better
400
350
300
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40
Simulation time in sec
Classic
PhysX
Optimized


Bunny500 optimization cost
Opt cost in ms - lower is better
14
12
10
8
6
4
2
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40
Simulation time in sec

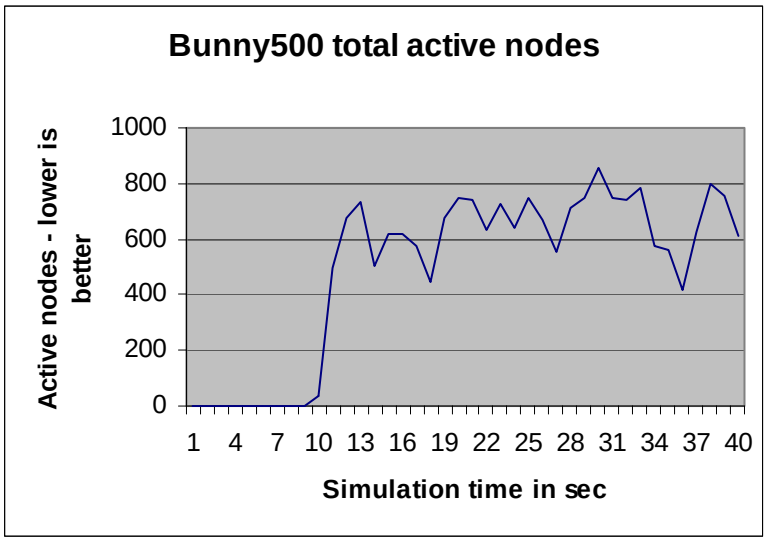
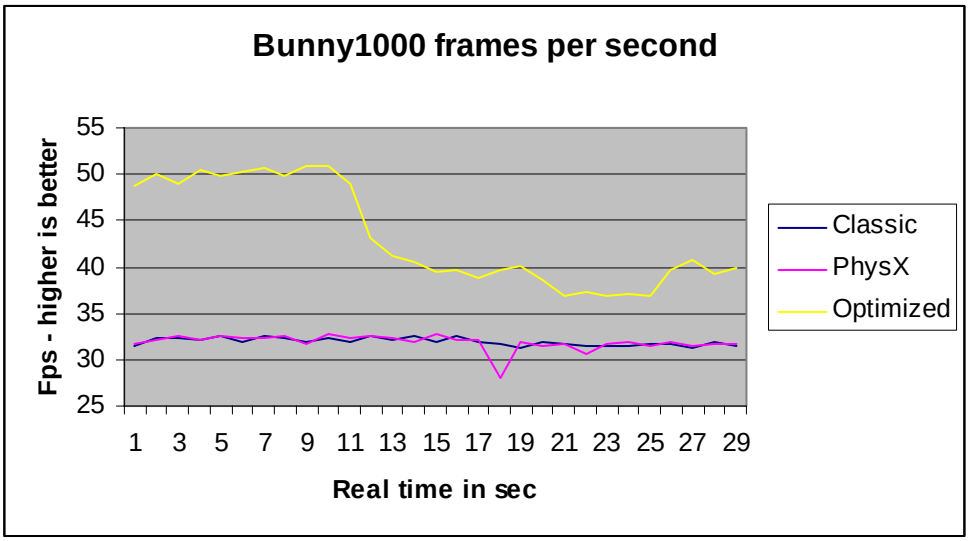
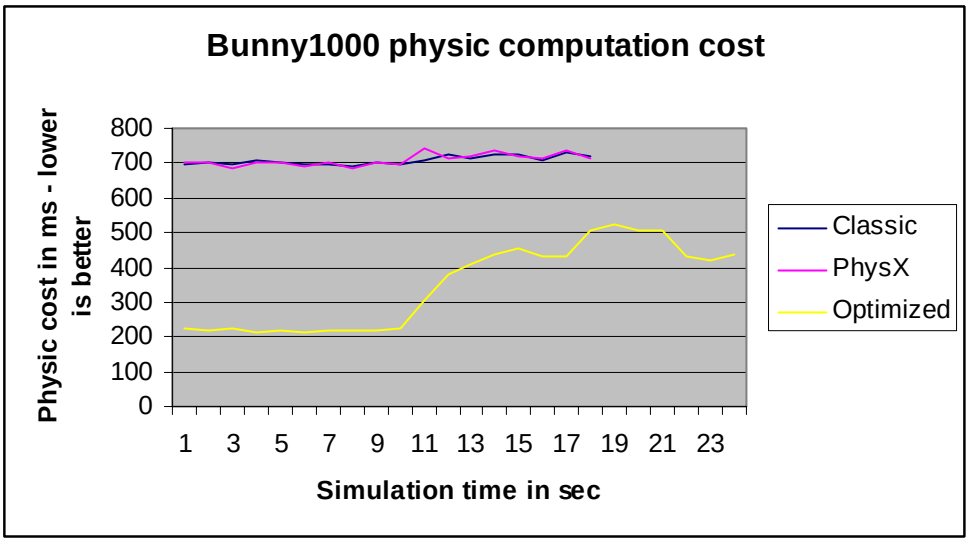
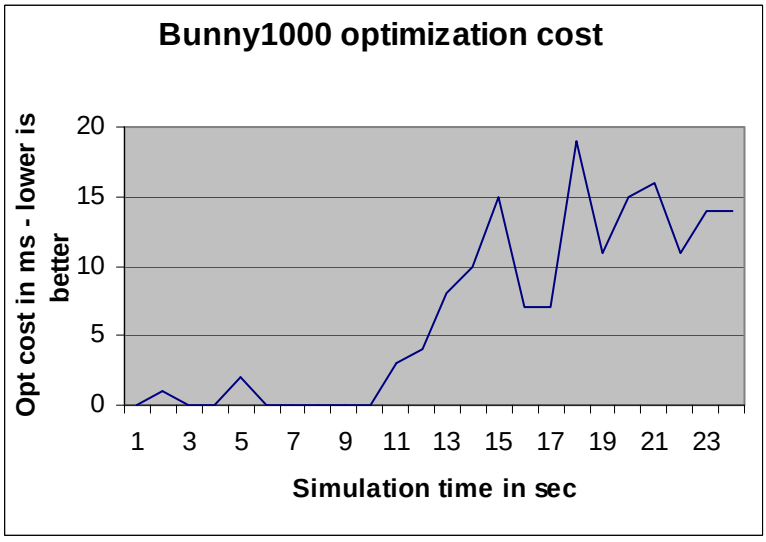


Figure 5.4 Benchmark charts for bunny500.

Figure 5.4 shows the benchmark data captured using medium resolution data, bunny500. Again, the fps is higher by large margin if using optimization method as showed in the first chart. In the second chart, total physic computation time is lower compared to other method of deformation. The maximum optimization method cost is 12 ms, twice higher than bunny100 worst case scenario optimization cost. Considering the number of vertices for bunny500 is almost six times higher than bunny100, the optimization cost is still very low. The final chart shows irregular spikes in total number of actives nodes. This indicates that, with default settings, the worst case scenario is hard to achieve. The optimization algorithm successfully maintains lower nodes for physics computations.


Bunny1000 frames per second
Fps - higher is better
55
50
45
40
35
30
25
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
Real time in sec
Classic
PhysX
Optimized


Bunny1000 physic computation cost
Physic cost in ms - lower is better
800
700
600
500
400
300
200
100
0
1 3 5 7 9 11 13 15 17 19 21 23
Simulation time in sec
Classic
PhysX
Optimized


Bunny1000 optimization cost
Opt cost in ms - lower is better
20
15
10
5
0
1 3 5 7 9 11 13 15 17 19 21 23
Simulation time in sec

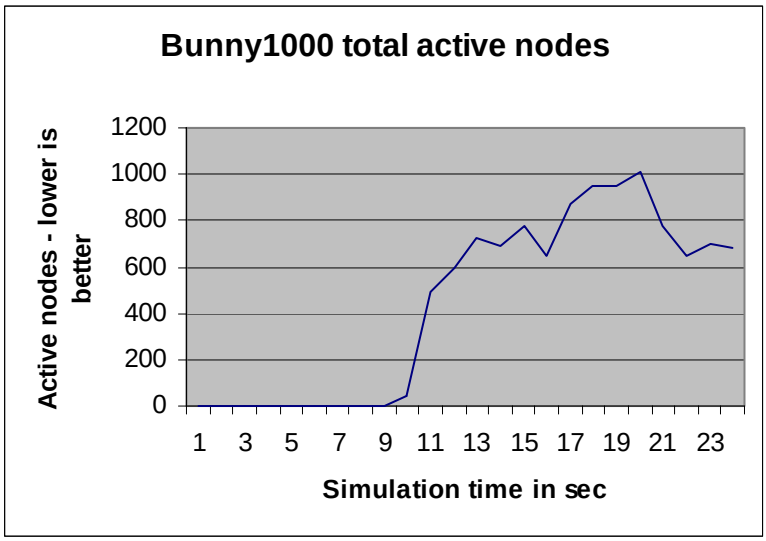


Figure 5.5 Benchmark charts for bunny1000.

The last benchmark shown in Figure 5.5 performed using bunny1000, the highest resolution data used in this benchmark. Like previous input data, the benchmark reported higher fps with optimization method. The physic computation cost also lower than the other two methods. The highest optimization cost reported is 19 ms, which is 7 ms higher compared to bunny500 input data. This is because, the optimization algorithm with current default setting implicitly does not permit node activation more than 1000 nodes as shown in the forth chart.

The results from this benchmark clearly indicated that the optimization algorithm can reduce deformation processing. Based on the benchmark performed on different resolution of input data, the optimization algorithm works efficiently better with high polygonal objects. The optimization overhead cost reported in all benchmark is very low with maximum of 19 ms. The next benchmark focuses of different settings of optimization algorithm.

5.3.6 Benchmarks against various settings

To better see the usage of various optimization algorithm settings, the next benchmark will perform a series of benchmark using icosa12 input data with different optimization algorithm settings. Detail of the setting are given in Table 5.3

Table 5.3 Optimization algorithm settings.

	Default	Setting 1	Setting 2	Setting 3	All low	All high	All medium
Active time threshold, ms	2000	1	2000	2000	1	4000	1000
Squared linear velocity magnitude threshold	10	10	1	10	1	20	10
Neighbor activation time threshold, ms	2000	2000	2000	1	1	4000	1000

There are seven optimization algorithm configurations. The default setting is the medium level, all purpose setting. Setting 1 through 3 are default settings with one of the parameters set to the lowest. The purpose of these configurations is to evaluate the impact of setting very low individual threshold parameter. The other three configurations set all the value to extreme. The All low setting sets all value to extreme low. The All low setting can also be considered as no threshold settings. The All high settings set the value to the highest suitable value. The All medium setting sets the value in between the All low and All high settings.

There are three threshold parameters available. Active time is the duration of how long the activated node remains active before it will be checked for deactivation. Velocity thresh is squared linear velocity magnitude threshold used in activation and deactivation procedure. Active nodes with squared linear velocity magnitude lower than Velocity thresh will be considered for deactivation. Active nodes with squared linear velocity magnitude higher than Velocity thresh will activated all its neighbors. Activate neighbor time is neighbor activation time

threshold. Nodes that have activated all its neighbor will only considered to re-activate all its neighbor again if the period of neighbor activation is more than activate neighbor time.

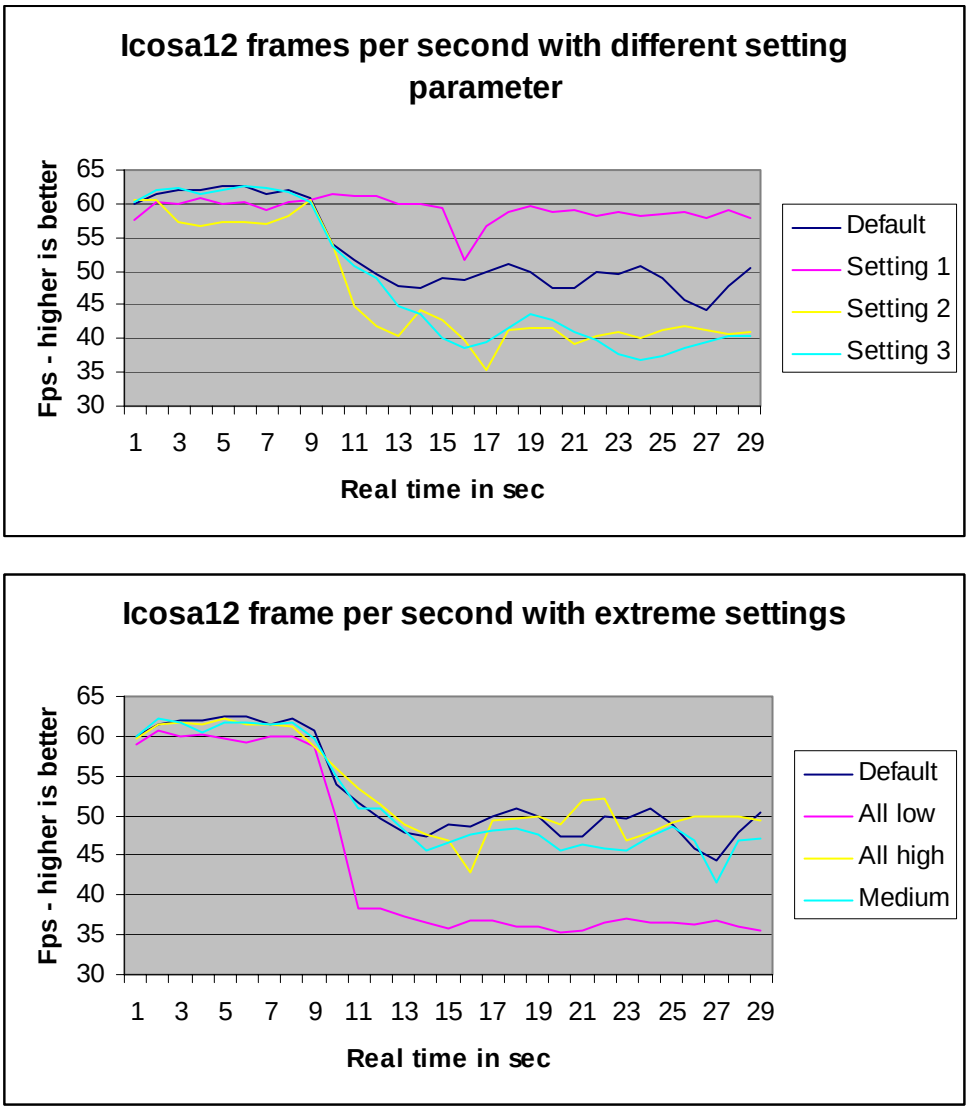


Figure 5.6 Frames per second benchmark result for icosa12.

The first chart from Figure 5.6 shows that, setting the active time very low yield better fps results. Setting the other two parameters to very low values will bring the performance down quite a bit. The second chart shows that setting all parameter to very low will yield very low performance.

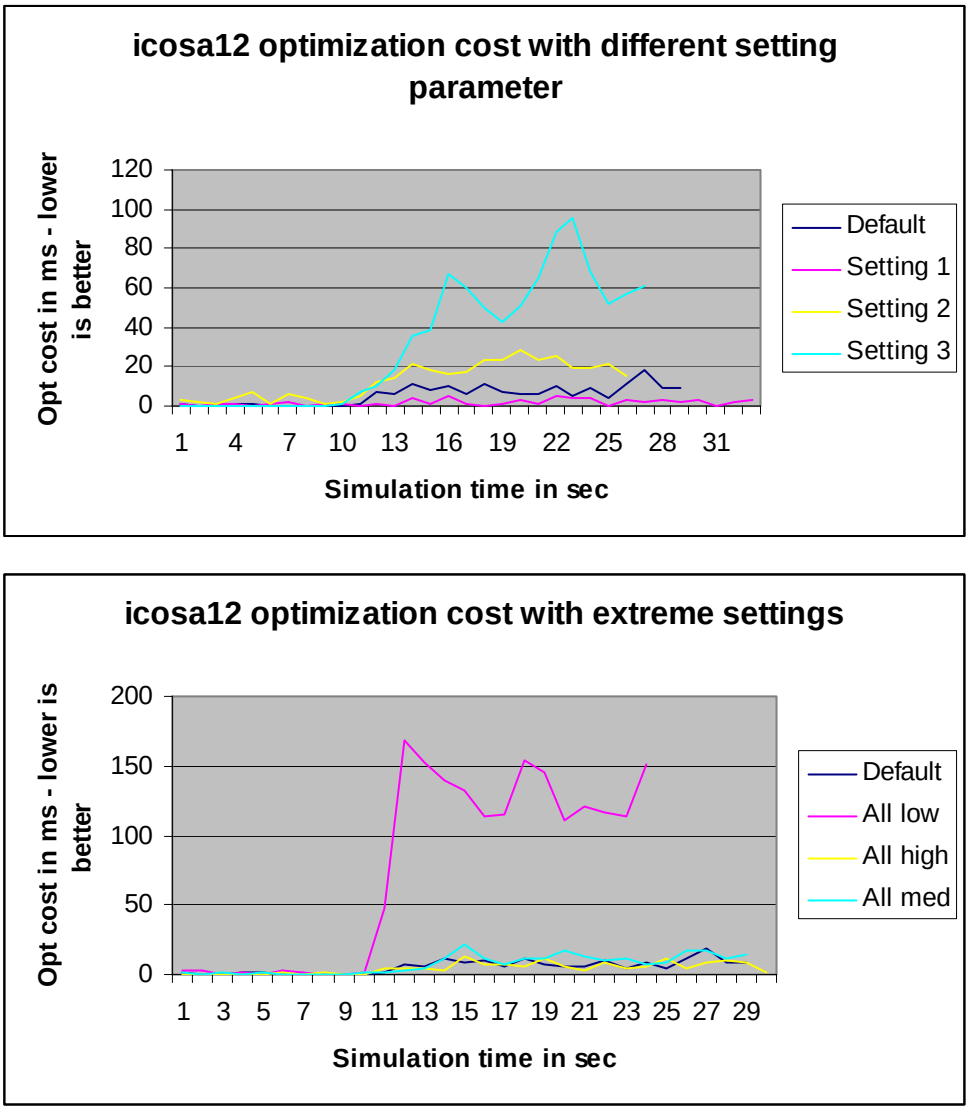


Figure 5.7 Optimization cost benchmark result for icosa12.

In Figure 5.7, the first chart shows that by setting neighbor activation time threshold to a very low value will result in higher optimization cost, which is bad. For best performance, active time threshold must be set to low. The second chart shows that setting all parameter values to low values will result in higher optimization cost. This is due to excessive node activation and deactivation for every time step.

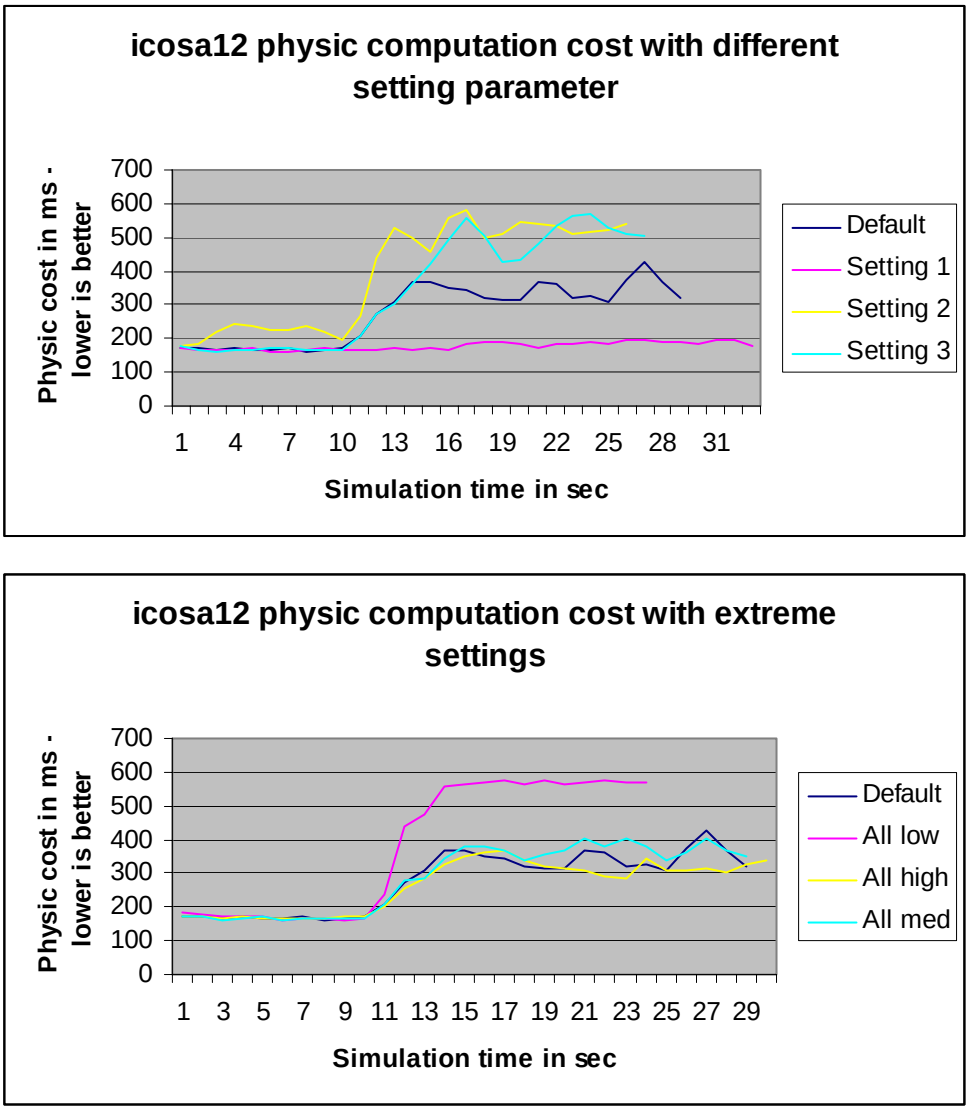


Figure 5.8 Physic computation cost benchmark result for icosa12.

Figure 5.8 show the benchmark result of physics computation cost. For lowest physic computation cost, active time threshold must be set to low. Setting squared linear velocity magnitude threshold to low will result in lower performance, as showed in the first chart. The second chart indicates that setting all values to very low values results in higher physics computation cost. The all high setting yield lowest physic computation cost.

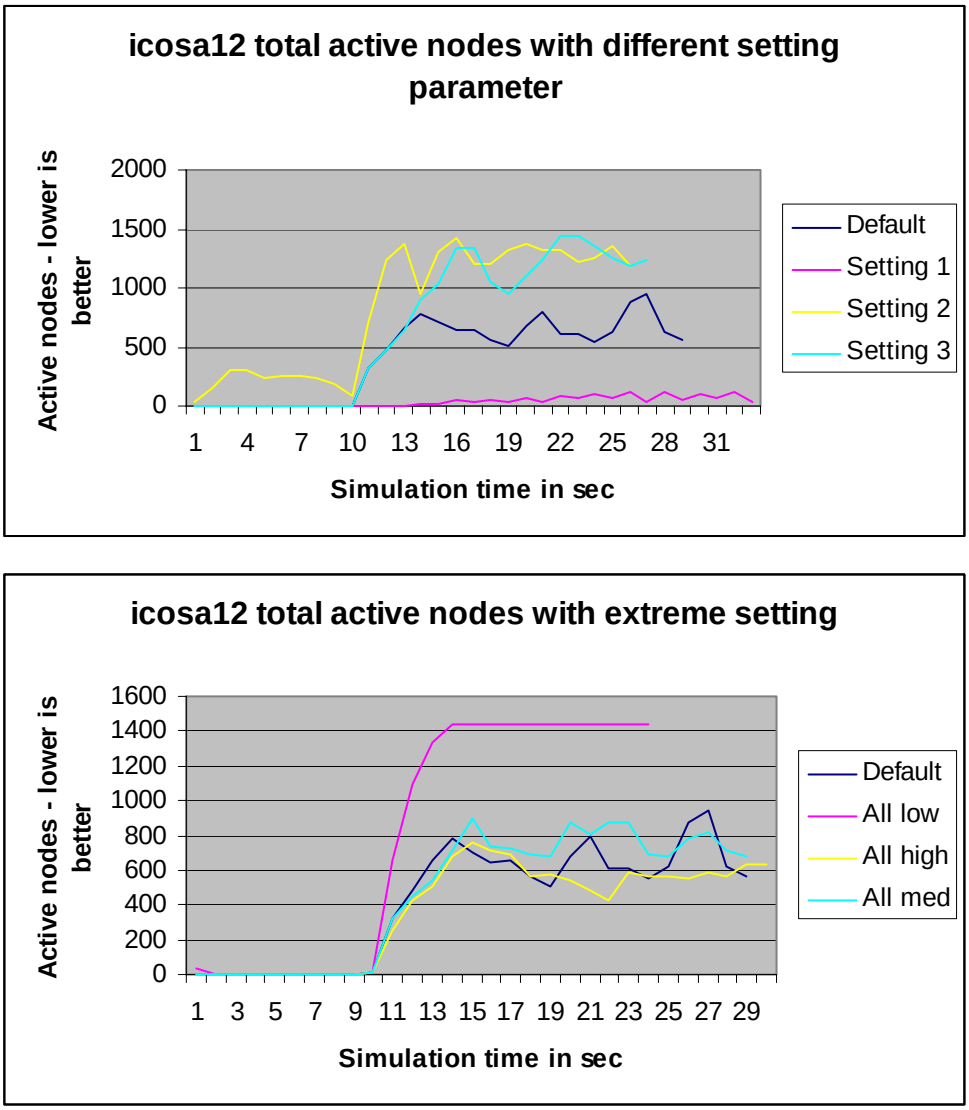


Figure 5.9 Total active nodes benchmark result for icosa12.

In Figure 5.9, the first chart shows that setting active time threshold to a very low values will results in lowest number active nodes. The second chart shows that the All high settings results in lowest number of activated nodes while the All low setting results in higher number of activated nodes.

The second series of benchmark tries to find the effect of different parameter setting. Based on the results, Active time threshold have the biggest performance impact. It must be set to low to achieve better performance. Setting it low will make

activated nodes easily deactivated. The other two parameters are better left at high values for better performance.

High performance comes at a cost of simulation accuracy. Sometimes the inaccuracy is evident visually. Since the visual perception is a subjective topic, it will not be discuss further. Nevertheless, the result from the benchmark should provide a good guidance on finding the best combination and parameter settings for any simulation requirements.

5.4 Other issues

The optimization algorithm contributes extra cost to the simulation systems. For small deformations where the total number of active nodes is small, the cost for optimization algorithm is very small. As the number of active nodes reaches the total number of nodes, running time performance may drop due to extra optimization overhead cost. For worst case scenario where all nodes were activated, the optimization algorithm is inefficient and will render the simulation system slower than classical simulation systems. Thus, the optimization algorithm is best use for small deformation (as shown in Figure 5.12).

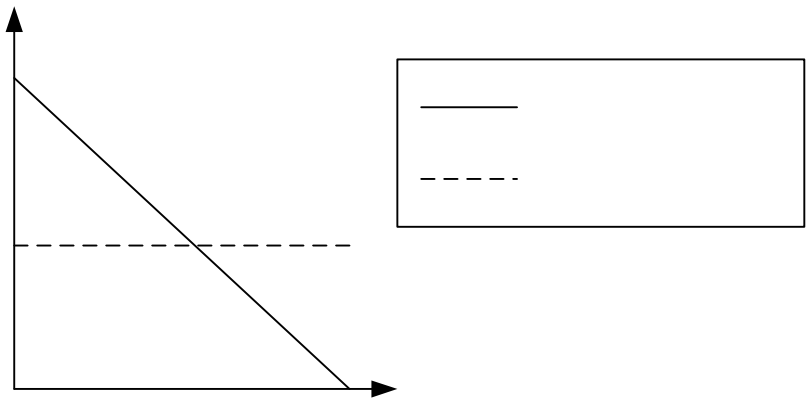


Figure 5.10 The higher the number of active nodes, the lower the performance for simulation systems with optimization algorithm.

The threshold, used as a mean of scaling the simulation system, can produce undesired behavior if used improperly. For best performance, active time threshold must be set to low, squared linear velocity magnitude threshold must be set to high and neighbor activation time must be set to high. This will make the nodes harder to activate and very easy to deactivate. On the contrary, for best behavior all settings must be set to low. This will results in easier nodes activation and faster nodes deactivation. The downfalls of this setting are the overhead cost of the optimization algorithm will rise quickly as more nodes are activated. With this setting, most of the time, large numbers of node are activated. This will defeat the purpose of this optimization algorithm where it would be best if only small portion of nodes active most of the time. Choosing the right threshold would be the matter of whether the simulation accuracy or speed is needed. For optimal solution, the settings must be suited around the distance for the node needs to be displaced 1 pixel in a single time step on the viewing device (see Figure 5.11). If, for example, the node is currently far away or the deformable object is highly complex where multiple nodes shares single pixel in the viewing device, the setting must be, at most, suited around the smallest distance between adjacent nodes (see Figure 5.12). To express the distance in other form such as velocity, the conversion would be how much velocity needed for the node to travel the distance in a single time step. Thus the optimal threshold for a node using distance as non equilibrium state definition would be the distance for the

node to be displaced to the next pixel in the viewing device or if the node shares single pixel in the viewing device with other nodes, the threshold would be the smallest distance of all neighbor distances. Choosing different threshold for every node at runtime using this method is expensive. One way to reduce the computation is to compute the threshold for a single node which has smallest node to neighbor distance. Since the deformation is small, it's easy to track the node which has the smallest node to neighbor distance. In reality, current hardware computation power is not enough for a real time simulation where the nodes is so dense it occupy single pixel with multiple nodes in the viewing device. Thus, the optimal setting method is not implemented.

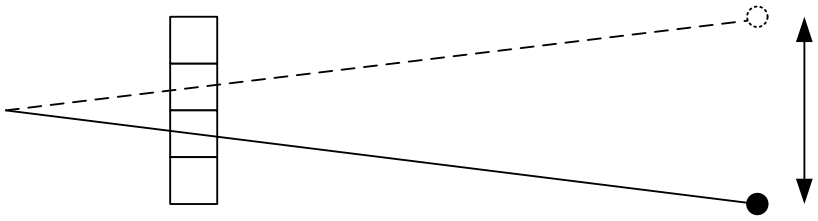


Figure 5.11 The optimal threshold must suited for the node to be displaced to the imaginary position which is the position where the node will be render at adjacent pixel.

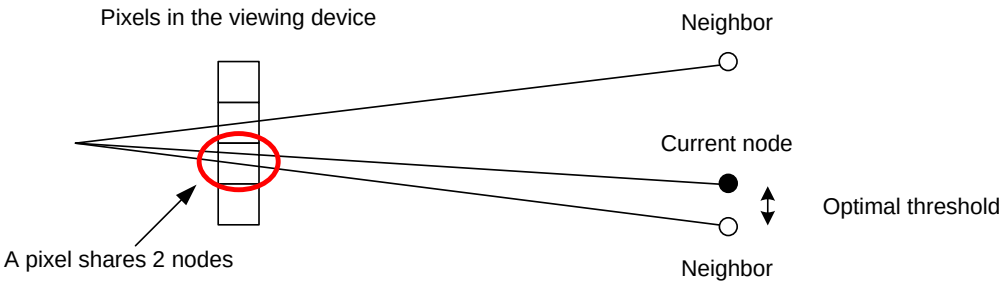


Figure 5.12 When node and its neighbor occupy the same pixel in the viewing device, the optimal threshold must suit for the smallest distance from node to neighbor between all neighbors.

The concentrated loads applied must be very high to ensure node activation. If the concentrated loads applied are very small, while the thresholds are set very high, the deformable object would remain undeform.

For very small concentrated loads, the node tends to deactivate real fast due to deactivation test being executed at every frame. This will result in unintended behavior where the deformable object looks very hard or harder to deform, especially when very small concentrated loads is applied. One way to eliminate this is by only deactivates node that has been active for a pre defined period of time. This will ensure that active node will always remain active after being activated for a period of time. Active nodes have the tendency to activate its neighbor due to its vulnerability to deform. Setting the time too high will result in more nodes active for a longer period, which is expensive.

Rendering time can be reduced by using vertex buffer object instead of immediate mode. This method stores vertex buffers in graphic card's high speed memory instead of system memory to improve rendering performance by minimizing data copying. However, this method is best use with static object. Although it is possible to use it with deformable object (by GPU based physics, etc.), it is not tested.

For deformable object, normals have to be computed every time the object deformed. Since the deformed vertices are active nodes, it is naturally easy to query which vertices or faces that require normals computations. Exploitation is cheap as the systems already have the list of active nodes and its list of neighbors. Flat shading is implemented for that it provides better visual cues on actual surfaces and it is much faster compared to Gouraud shading due to multiple normal averaging per vertex in each frame.

5.5 Conclusions

The main goal of optimization algorithm is to reduce the cost of deformation processing. The benchmark results showed that for deformable objects interacting with only concentrated loads, the optimization algorithm successfully reduce the deformation processing especially for objects with high polygons as shown in the first chart of Figure 5.5.

Charts in Figure 5.3 through Figure 5.5 shows that, the effect having upon large number of nodes will reduce the simulation performance due to large number of nodes to be process. However, the cost of optimization algorithm is still very small with maximum of 19 ms per time step. The second series of benchmark clearly shows that the optimization algorithm can be configured for either performance or best deformation behavior. Various combination of setting and its implication on performance are shown in Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9. Different total frames per second captured from the first series of benchmark shows that the optimization algorithm successfully scales the deformation area as required. This is very evident in Figure 5.5 where when there are no deformation, the fps would go around 50 fps and when there are lots of deformation, the fps would go around 40 fps.

The next chapter will discuss the achievement of the optimization algorithm based on research objectives.

CHAPTER VI

CONCLUSIONS

6.1 Introductions

This chapter reviews the research objectives and research findings. First, the objectives are reviewed along with it's prove of achievement. Next is a list of contributions from the research. Finally, outlines of possible future work.

6.2 Summary

In short, all four objectives as outline in Chapter 1 are achieved. Deformable objects are represented using mass-spring model, which we find most appropriate. Our method called the dynamic selection based method is able to reduce the computation for deforming an object and this is demonstrated in a real-time simulation. The emphasis of this research is on three most critical concerns, which are summarized below.

1. **To reduce deformation processing cost by reducing areas (total number of nodes) for deformation.**
    a. The benchmark result from the first charts in Figure 5.3, Figure 5.4 and Figure 5.5 (first series of benchmarks) shows that overall, the optimization algorithm successfully reduce the processing cost of the simulation.
    b. The second charts from the Figure 5.3, Figure 5.4 and Figure 5.5 shows the deformation processing cost. The optimization algorithm successfully reduces total deformation processing cost throughout the simulation.
    c. The third chart from Figure 5.3, Figure 5.4 and Figure 5.5 shows that the extra cost for optimization algorithm does influence overall performance. However, the cost is extremely low. Simulation with optimization algorithm is still superior compared to other tested method thus making the extra computation cost of optimization algorithm worth it.
2. **To construct a dynamic method that can enlarge or shrink deformation areas.**
    a. The forth charts from Figure 5.3, Figure 5.4 and Figure 5.5 shows total number of nodes that are being processed for deformation for current time step. Total numbers of active nodes vary according to the simulation needs. This proves the optimization algorithm succeeded in enlarging or shrinking the deformation areas as required.
    b. Evidence of the optimization method can enlarge or shrink deformation areas as required can be seen from the first and second charts of Figure 5.3, Figure 5.4 and Figure 5.5. These figures show that during the first 10 seconds of simulation time, when there is no deformation, the simulation performances are at its best. After that, external forces are applied to the deformable object to make it deform. This makes the performance drop quite a bit due to deformation processing.

3. **To develop a deformation system that can be scaled to either higher performance or higher accuracy.**
    a. The second series of benchmark (Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9) evaluates the influence of each optimization algorithm parameter settings. The results indicate that the configuration settings can be tune to provide either best performance or best accuracy.

6.3 Contributions

Listed here are contributions made in this research and its comparison to other similar method.

1. **Reduced area for deformation:** For every frame, deformable object is evaluated for deformation. The result from the evaluation is a small area of deformable object that will be selected for deformation. Similar in nature to ChainMail (Gibson 1997), this will reduce required deformation processing time as only small areas are actually deformed per frame.
2. **Dynamically enlarge or shrink deformation area:** Unlike previous deformation method inspired by force propagation, dynamic selection based method can dynamically enlarge or shrink deformation areas. Previous works usually either resort to static range of areas (Choi et al. 2003) or propagate over the deformable object infinitely (Dusyak and Zhang. 2004). Other method that can dynamically enlarge or shrink deformation area doesn't have physical based justifications in its deformations.
3. **Scalable for performance or accuracy:** In order to tackle broad range of applications, dynamic selection based method allows the user to tinker with the parameter settings. These settings enable the application to be tuned for high

accuracy or high performance. Chapter 5 provides testing result of different parameter settings.

4. **Low cost definition of equilibrium state:** Definition of equilibrium state is a requirement in order to effectively select areas that should be deformed. This research provides an efficient method to define equilibrium state based on physics justifications. Different methods of equilibrium state complete with its comparison analysis of computation cost are provided.
5. **Independent of deformation method:** The algorithm is successfully implemented in existing physical based deformation method. Generally, the algorithm is a selection algorithm. There should be no major problem to implement this algorithm in other physical based deformation method.
6. **Robust algorithm:** There are situations which needs special care in previous works. Different number of neighbours between inside node and surface nodes poses a problem for ChainMail (Gibson 1997). (Choi et al. 2003) and Dragnet (Grimm et al. 2004) experiences problem for multiple contact situation and special care had to be taken. Due to high generality of the proposed algorithm(no neighbour assumptions), this research shows that using the single provided algorithm, no special care is needed to handle above mentioned situations.
7. **Works for both structured and unstructured mesh:** Unlike certain algorithm (Gibson, 1997), this optimization algorithm works for both structured and unstructured mesh. This is because the algorithm does not assume and does not restrict the number of springs per each node.

6.4 Future work

This report presented an optimization technique to existing popular method. The results are better in some areas compared to other similar method in the same domain.

However, there is always a room for future research. Listed here are some suggestions, improvements and open problems based on research findings.

1. The biggest problem with the optimization algorithm is the node activation algorithm. For node activation, the algorithm will activate all neighboring node, even if the node is already activated. This results in wasted resources. The wasted resources are too high as can be seen in the second chart of Figure 5.7 (the All low settings). Although this problem can be counter by applying multiple thresholds, the excessive use of thresholds leads to other problem (bad deformation behavior)
2. Currently, there are two separated loops that read the active node data structure for every time step, one for activation and one for deactivation. If both activation and deactivation can be performed in a single loop, it will surely boost overall simulation performance.
3. Currently, there are no ‘one size fits all’ for optimization algorithm settings. Choosing the right threshold is very tricky, unless the environment is restricted.
4. The nature of the system changes the behavior of the original materials because of the delay from the force propagation.

BIBLIOGRAPHY

Breen D., House D., Wozny M.: “Predicting The Drape Of Woven Cloth Using Interacting Particles.” In Siggraph ’94 1994

Barbic J., James D. L.: “Real-Time Subspace Integration For St.Venant- Kirchhoff Deformable Models.” Acm Transactions On Computer Graphics Acm Siggraph 2005

Morten Bro-Nielsen M., Cotin S.: “Real-Time Volumetric Deformable Models For Surgery Simulation Using Finite Elements And Condensation.” Computer Graphics Forum 1996

Baraff D., Witkin A.: “Large Steps In Cloth Simulation.” In Proceedings Of Siggraph 1998

Kwang-Jin Choi , Hyeong-Seok Ko, “Stable but responsive cloth”, ACM Transactions on Graphics (TOG), v.21 n.3, July 2002

David Baraff , Andrew Witkin , Michael Kass, “Untangling cloth”, ACM Transactions on Graphics (TOG), v.22 n.3, July 2003

Morten Bro-Nielsen: Surgery Simulation Using fast Finite Elements. VBC 1996: 529-534

S. Cotin, H. Delingette, J.M. Clement, V. Tassetti, J. Marescaux, and N. Ayache, Volumetric deformable models for simulation of laparoscopic surgery, Proc. Computer Assisted Radiology (CAR'96), pp. 793-798, 1996

S. Cotin, H. Delingette, J.M. Clément, M. Bro-Nielsen, N. Ayache , J. Marescaux, Geometrical and Physical Representations for a Simulator of Hepatic Surgery, Proc. MMVR-4'96, 1996

Cotin S., Delingette H., Ayache N.: “Real-Time Elastic Deformations Of Soft Tissues For Surgery Simulation.” In Ieee Transactions On Visualization And Computer Graphics, Vol. 5. 1999

Cotin S., Delingette H., Ayache N.: “A Hybrid Elastic Model Allowing Real-Time Cutting, Deformations And Force-Feedback For Surgery Training And Simulation.” The Visual Computer 16, 2000

Chadwick J., Haumann D., Parent R.: “Layered Construction For Deformable Animated Characters.” In Siggraph ’89 1989

Chen Y., Zhu Q., Kaufman A., Muraki S.: “Physically-Based Animation Of Volumetric Objects.” In Ca ’98: Proceedings Of The Computer Animation 1998

Grinspun E., Krysl P., Schröder P.: “Charms: A Simple Framework For Adaptive Simulation.” In Proceedings Of Siggraph 2002 2002

Gibson S. F., Mirtich B.: “A Survey Of Deformable Models In Computer Graphics.” Technical Report Tr-97-19, Merl, Cambridge, Ma, 1997

House D., Breen D.: “Cloth Modeling And Animation.” A. K. Peters, Ltd., 2000

Hutchinson D., Preston M., Hewitt T.: “Adaptive Refinement For Mass/Spring Simulations.” In Proceedings Of The Eurographics Workshop On Computer Animation And Simulation ’96 1996

Hauser K. K., Shen C., O’brien J. F.: “Interactive Deformation Using Modal Analysis With Constraints.” In Graphics Interface ’03 2003

Hunter P.: “Fem/Bem Notes.” University Of Oakland, New Zealand, 2005. Http://Www.Bioeng.Auckland.Ac.Nz/Cmiss/Fembemnotes/Fembemnotes

James D. L., Barbi ˇC J., Twigg C. D.: “Squashing Cubes: Automating Deformable Model Construction For Graphics.” In Proceedings Of The Siggraph 2004 Conference On Sketches & Applications 2004

James D. L., Fatahalian K.: “Precomputing Interactive Dynamic Deformable Scenes.” Acm Transactions On Graphics Proceedings Of Acm Siggraph 2003

James D. L., Pai D. K.: “Artdefo: Accurate Real Time Deformable Objects.” In Siggraph ’99 1999

James D. L., Pai D. K.: “Multiresolution Green’s Function Methods For Interactive Simulation Of Large-Scale Elastostatic Objects.” Acm Transactions On Graphics 22, 1 2003

Müller M., Dorsey J., Mcmillan L., Jagnow R., Cutler B.: “Stable Real-Time Deformations.” In Proceedings Of The 2002 Acm Siggraph/Eurographics Symposium On Computer Animation 2002

Müller M., Gross M.: “Interactive Virtual Materials.” In Gi ’04: Proceedings Of Graphics Interface 2004

Müller M., Keiser R., Nealen A., Pauly M., Gross M., Alexa M.: “Point Based Animation Of Elastic, Plastic And Melting Objects.” In Proceedings Of The 2004 Acm Siggraph/Eurographics Symposium On Computer Animation 2004

O’brien J. F., Bargteil A. W., Hodgins J. K.: “Graphical Modeling And Animation Of Ductile Fracture.” In Proceedings Of Siggraph 2002

Platt S. M., Badler N. I.: “Animating Facial Expressions.” In Siggraph ’81 1981

Point Based Animation: Resource Collection On The World Wide Web. Http://www.pointbasedanimation.org, 2004

Picinbono G., Delingette H., Ayache N.: “Nonlinear And Anisotropic Elastic Soft Tissue Models For Medical Simulation.” In Proceedings Of The Ieee International Conference On Robotics And Automation 2001

Provot X.: “Deformation Constraints In A Mass-Spring Model To Describe Rigid Cloth Behaviour.” In Proc. Graphics Interface 1995

Press W., Teukolsky S., Vetterling W., Flannery B.: “Numerical Recipes In C - The Art Of Scientific Computing, 2nd Ed.” Cambridge University Press, 1992

Reeves W. T.: “Particle Systems – A Technique For Modeling A Class Of Fuzzy Objects.” Acm Trans. Graph. 1983

Szeliski R., Tonnesen D.: “Surface Modeling With Oriented Particle Systems.” Computer Graphics 26, 2 1992

Stam J.: “Stable Fluids.” In Siggraph ’99 1999

Terzopoulos D., Fleischer K.: “Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture.” In Siggraph ’88 1988

Teschner M., Heidelberger B., Müller M., Gross M.: “A Versatile And Robust Model For Geometrically Complex Deformable Solids.” In Proceedings Of Computer Graphics International Cgi Jun 2004

Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Fuhrmann A., Cani M.-P., Faure F., Magnenat Thalmann N., Strasser W., Volino P.: “Collision Detection For Deformable Objects.” Computer Graphics Forum 24, 1 2005

Terzopoulos D., Platt J., Barr A., Fleischer K.: “Elastically Deformable Models.” In Siggraph ’87 1987

Terzopoulos D., Witkin A.: “Physically Based Models With Rigid And Deformable Components.” Ieee Computer Graphics And Applications 8, 6 1988

Terzopoulos D., Platt J., Fleischer K.: “Heating And Melting Deformable Models From Goop To Glop.” In Graphics Interface ’89 1989

Terzopoulos D., Waters K.: “Physically-Based Facial Modeling, Analysis, And Animation.” Journal Of Visualization And Computer Animation 1, 1 1990,

Volino P., Magnenat-Thalmann N.: “Developing Simulation Techniques For An Interactive Clothing System.” In Proceedings Of The 1997 International Conference On Virtual Systems And Multimedia 1997

Volino P., Magnenat-Thalmann N.: “Implementing Fast Cloth Simulation With Collision Response.” Ieee Computer Society 2000

R. Bridson, S. Marino, R. Fedkiw.: “Cloth & deformable bodies: Simulation of clothing with folds and wrinkles” Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation SCA '03, July 2003

Witkin A., Baraff D.: “Physically Based Modeling: Principles And Practice.” Siggraph Course Notes 1995, 1997

Wu X., Downes M. S., Goktekin T., Tendick F.: “Adaptive Nonlinear Finite Elements For Deformable Body Simulation Using Dynamic Progressive Meshes.” Eurographics Sept. 2001.

Matthias Mueller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, Marc Alexa.:”Point Based Animation Of Elastic, Plastic And Melting Objects.” In

Proceedings Of The Acm Siggraph/Eurographics Symposium On Computer Animation 2004

Mark Pauly, Dinesh K. Pai, Leonidas J. Guibas.:"Quasi-Rigid Objects In Contact" In Proceedings Of The Acm Siggraph/Eurographics Symposium On Computer Animation 2004

Aras Pranckevicius.:"Xplodar - A Tiny Fem Simulator Demo." 2004 Dec, Done For University's 'Finite Element Modelling' Course Http://Nesnausk.Org/Nearaz/Projxplodar.Html

Alan Watt, Mark Watt.:"Advanced Animation And Rendering Techniques:Theory And Practice" Acm Press, Addison-Wesley 1992

Christophe Geuzaine And Jean-François Remacle.:"Gmsh: A Three-Dimensional Finite Element Mesh Generator With Built-In Pre- And Post-Processing Facilities" Version 1.60, 14 March 2005 Www.Geuz.Org/Gmsh/Index.Htm

Hang Si.:"Tetgen : A Quality Tetrahedral Mesh Generator And Three-Dimensional Delaunay Triangulator."Research Group Of Numerical Mathematics And Scientific Computing, Weierstrass Institute For Applied Analysis And Stochastics Mohrenstr. Version 1.3.4, June 17, 2005 Http://Tetgen.Berlios.De/

Kyle Owen.:"Area And Volume Calculations" Http://Www.Gamedev.Net/Reference/Articles/Article2247.Asp Kwoxrf@Umr.Edu 5/12/2005

Maciej Matyka And Mark Ollila.:"Pressure Model Of Soft Body Simulation" Sigrad2003, The Annual Sigrad Conference. Special Theme – Real-Time Simulations, November 20–21, 2003, Umeå University, Umeå, Sweden

Thomas Jakobsen.:”Advanced Character Physics” Http://Www.Gamasutra.Com/Resource_Guide/20030121/Jacobson_01.Shtml Gamasutra Weekly article January 21, 2003

John E. Chadwick, David R. Haumann, Richard E. Parent.:”Layered Construction For Deformable Animated Characters” Computer Graphics, Volume 23, Number 3, July 1989

Mollemans, Wouter and Schutyser, Filip and Cleynenbreugel, Johan Van and Suetens, Paul Tetrahedal mass spring model for fast soft tissue deformation IS4TM, 2003 , 145-154

Christensen, J., Marks, J. and Ngo, J. T., \Automatic motion synthesis for 3D mass-spring models", The Visual Computer, 1997, No. 13, pp. 20{28.

Petr Krysl, Eitan Grinspun, Peter Schr¨oder.:”Natural Hierarchical Refinement For Finite Element Methods” International Journal For Numerical Methods In Engineering, Vol 56, Num 8, February 2003

Petros Faloutsos, Michiel Van De Panne, Demetri Terzopoulos.: “Dynamic Free-Form Deformations” Ieee Transactions On Visualization And Computer Graphics, Vol. 3, No. 3, July-September 1997 201

William M Hsu, John F Hughes, Henry Kaufman .:“Direct Manipulation Of Free Form Deformations” Computer Graphics 26, 2, July 1992

Gentaro Hirota, Renee Maheshwari, Ming C. Lin.: “Fast Volume-Preserving Free Form Deformation Using Multi-Level Optimization” Appeared In Acm Solid Modeling '99 Paper Session

Basdogan, C., 2001, "Real-Time Simulation Of Dynamically Deformable Finite Element Models Using Modal Analysis And Spectral Lanczos Decomposition Methods", Proceedings Of The Medicine Meets Virtual Reality Mmvr’2001 Conference, Jan 24-27, Irvine, Ca

Berkley, J., Et Al., 2000, “Creating Fast Finite Element Models From Medical Images”, Proceedings Of Medicine Meets Virtual Reality 2000.

Xunlei Wu, Michael S. Downes, Tolga Goktekin, Frank Tendick.: “Adaptive Nonlinear Finite Elements For Deformable Body Simulation Using Dynamic Progressive Meshes” Eurographics 2001 / A. Chalmers And T.-M. Rhyne Guest Editors Volume 20 2001, Number 3

P.Volino, M. Courchesne, And N. M. Thalmann.: “Versatile And Efficient Technique For Simulating Cloth And Other Deformable Objects”. Proc. Siggraph 1995

Http://Www.Informatik.Umu.Se/~Jwworth/Medpage.Html

Http://Www.Radiologyinfo.Org/Content/Diagnostic/Diagnostic.Htm

Advanced Finite Element Methods Asen 5367 - Spring 2003. Aerospace Engineering Sciences - University Of Colorado At Boulder Online Notes Http://Caswww.Colorado.Edu/Courses.D/Afem.D/Home.Html

Kawabata, S.:“The Standardization And Analysis Of Hand Evaluation.”The Textile Machinery Society Of Japan, Osaka, 1980

Alan H. Barr.: “Global And Local Deformations Of Solid Primitives” Proc. Siggraph 1984

Stephen M. Platt, Norman I. Badler.:“Animating Facial Expressions” August 1981 Acm Siggraph Computer Graphics , Proceedings Of The 8TH Annual Conference On Computer Graphics And Interactive Techniques, Volume 15 Issue 3

Thomas W. Sederberg, Scott R. Parry.:”Free-Form Deformation Of Solid Geometric Models” Siggraph 1986

Sabine Coquillart.:“Extended Free-Form Deformation : A Sculpturing Tool For 3d Geometric Modeling” Siggraph 1990

Yuencheng Lee, Demetri Terzopoulos, Keith Waters .:”Realistic Modeling For Facial Animation” 1995 Computer Graphics

Chen, D., And Zeltzer, D.:“Pump It Up: Computer Animation Of A Biomechanically Based Model Of Muscle Using The Finite Element Method," Proc. Siggraph'92

L. P. Nedel, D. Thalmann.:”Real Time Muscle Deformations Using Mass-Spring Systems” Proceedings Of The Computer Graphics International 1998

Joel Brown, Jean-Claude Latombe, Kevin Montgomery .:“Real-Time Knot-Tying Simulation” 27 April 2004 Springer-Verlag 2004

Andrew M. Ladd, Lydia E. Kavraki.:”Using Motion Planning For Knot Untangling” Department Of Computer Science, Rice University, Houston, Tx 77005, Usa April 8, 2003

Jeff Philips, Andrew Ladd, Lydia E. Kavraki.:“Simulated Knot Tying” Ieee Int Conf On Robotics And Automation 2002

Poh Liong Yong, S. Nagappan.:” Physiscs For Stpm Volume 1” Penerbit Fajar Bakti Sdn Bhd 2003

Michael Hauth.:”Visual Simulation Of Deformable Models” PhD Thesis 2004

A. Fuhrmann, C. Groß, and V. Luckas. Interactive animation of cloth including self collision detection. In Proceedings of Winter School of Computer Graphics WSCG 2003

Pocino, Nick. Writing a Verlet-Based Physics Engine, Game Programming Gems 4, Charles River Media, 2004

Markus A. Schill, Sarah F. Frisken Gibson, Hans-Joachim Bender, Reinhard Männer: Biomechanical Simulation of the Vitreous Humor in the Eye Using and Enhanced ChainMail Algorithm. MICCAI 1998: 679-687

Sarah F. Frisken Gibson: 3D Chainmail: A Fast Algorithm for Deforming Volumetric Objects. SI3D 1997: 149-154, 195

Kup-Sze Choi, Hanqiu Sun, Pheng-Ann Heng, Jun Zou: Deformable simulation using force propagation model with finite element optimization. Computers & Graphics 28(4): 559-568 (2004)

Kup-Sze Choi, Hanqiu Sun, Pheng-Ann Heng, Jack C. Y. Cheng: A scalable force propagation approach for web-based deformable simulation of soft tissues. Web3D 2002: 185-193

Sarah F. Frisken-Gibson, Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining, IEEE Transactions on Visualization and Computer Graphics, v.5 n.4, p.333-348, October 1999

Jinah Park, Sang-Youn Kim, Seung-Woo Son, Dong-Soo Kwon, Shape retaining chain linked model for real-time volume haptic rendering, Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics p. 65 - 72, 2002

Johannes P. W. Grimm, Clemens Wagner, Reinhard Manner, Interactive Real-Time Simulation of the Internal Limiting Membrane, Medical Simulation: International Symposium, ISMS 2004, Cambridge, MA, USA, June 17-18, 2004.

A. Duysak, Jian J. Zhang, "Fast Simulation of Deformable Objects," iv, pp. 422-427, Eighth International Conference on Information Visualisation (IV'04), 2004.

Desbrun M., Cani M.-P.: “Animating Soft Substances With Implicit Surfaces.” In Computer Graphics Proceedings 1995, ACM Siggraph

Mathieu Desbrun, Marie-Paule Gascuel: “Animating Soft Substances with Implicit Surfaces.” Siggraph 1995

Mathieu Desbrun, Nicolas Tsingos, Marie-Paule Gascuel: “Adaptive Sampling Of Implicit Surfaces For Interactive Modelling And Animation.” Comput. Graph. Forum 1996

Desbrun M., Cani M.-P.: “Smoothed Particles: A New Paradigm For Animating Highly Deformable Bodies.” In 6th Eurographics Workshop On Computer Animation And Simulation ’96 1996

Marie-Paule Cani, Mathieu Desbrun: “Animation of Deformable Models Using Implicit Surfaces.” Ieee Trans. Vis. Comput. Graph. 1997

Desbrun M., Cani M.-P.: “Active Implicit Surface For Animation.” In Proceedings Of Graphics Interface 1998

Debunne G., Desbrun M., Barr A., Cani M.-P.: “Interactive Multiresolution Animation Of Deformable Models.” In Eurographics Workshop On Computer Animation And Simulation ’99 1999

Desbrun M., Schröder P., Barr A. H.: “Interactive Animation Of Structured Deformable Objects.” In Graphics Interface ’99 1999

Mathieu Desbrun, Peter Schröder, Alan H. Barr: “Interactive Animation Of Structured Deformable Objects.” Graphics Interface 1999

Desbrun M., Cani M.-P.: “Space-Time Adaptive Simulation Of Highly Deformable Substances.” Tech. Rep., Inria Nr. 3829, 1999

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, Alan H. Barr: “Adaptive Simulation of Soft Bodies In Real-Time.” Ca 2000

Debunne G., Desbrun M., Cani M.-P., Barr A. H.: “Adaptive Simulation Of Soft Bodies In Real-Time.” In Computer Animation ’00 2000

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, Alan H. Barr: “Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling”. Siggraph 2001

Mark Meyer, Gilles Debunne, Mathieu Desbrun, Alan H. Barr: “Interactive Animation of Cloth-Like Objects in Virtual Reality.” Journal of Visualization and Computer Animation 12 2001

Debunne G., Desbrun M., Cani M.-P., Barr A.: “Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling.” In Computer Graphics Proceedings Aug. 2001, Annual Conference Series, Acm Siggraph 2001