Combining Rigid body simulations with oriented particles

Bachelor's thesis

TECHNISCHE UNIVERSITÄT DARMSTADT

GRADUATE SCHOOL
computational engineering

Technische Universität Darmstadt, 2010

Hiermit versichere ich, die vorliegende Arbeit unter der Betreuung von Prof. Dr. X. Xxxxxx und Dxxx.-Ing. X. Xxxxx nur mit den angegebenen Hilfsmitteln selbst"andig angefertigt zu haben.

Darmstadt, den

Abstract

The major objective of this study is to explore the possibility to combine a traditional rigid body simulation with a deformable body simulation. The deformable bodies are implemented according to the relatively new approach of using oriented particles. The combined bodies consist of a rigid inner structure and a soft outer structure. The goal is to improve upon traditional system simulating skeletons. The bones can now be modeled with a surrounding cushion of tissue using the presented method. This extends and improves simpler models using only rigid bodies especially in areas where number of contact points is important to achieve a stable simulation. Using the soft deformable tissue enables bodies to surround other bodies more realistically while increasing the number of contact points significantly. This combined simulation model thus can provide a more stable simulation.

The main contribution of this thesis is the modeling and description of the interface between the deformable oriented particles and rigid bodies. The first interface is inside the body itself between the bones and the tissue, which requires an efficient way to closely couple both subsimulations. The second interface is between these combined bodies and rigid bodies in the simulated world. Here the impulses generated by the tissue onto the rigid bodies has to be defined and calculated.

The results show that the simulation provides realistic results. For the simple scenario of picking up a cylinder only two rigid bodies, modeling the fingers, are needed. A rigid body only simulation would either completely fail at this task or would required multiple bodies to simulate the whole finger. This result promises both more efficient and faster simulations of complete skeleton models.

Contents

List of Figures

List of Tables

Liste der Algorithmen

1 Introduction

Simulating the physically properties of objects with the help of a computer is a very important topic across a wide range of fields. As computing power is both expensive and limit the approach to simulations is distributed across a spectrum of accuracy at the one end and performance on the other. Some fields require the best and most accurate simulations currently available. These are especially predominant in applications where predictions for the real world behavior of the system are required and later applied to constructing and optimizing real world objects. Example fields for this are automotive or aerospace modeling and simulations. Here extremely detailed models are used to predict the properties and behavior of all kinds of different materials, shapes and bodies. The results of these simulations directly influence the construction and design decisions for their real world counter parts. Simulating the objects first inside the computer enables a faster development cycle and thus provides an efficient and cost effective way of constructing new parts and pieces.

Other fields are not reliant on the direct applicability of their results to the real world. The main focus of these simulations is to provide physically plausible and visually believably results in a timely fashion. Probably the two most popular and widely known application for these types of simulations are Games and Animations. Here the goal is to introduce the viewer into a completely virtual universe where everything is controlled by the media creators. In order to achieve the desired immersion of the viewers realistic behavior of the actors and objects in this universe is really important. Obvious incorrect physical behavior is most of the time immediately recognizable by the consumers and can easily break their immersion in the content.

Today games and animations use a huge number of interacting bodies. These bodies are simulated using a variety of different simulation methods and techniques. For animating characters and objects in movies the main focus lies in reducing the workload of animators, who traditionally had to animate and model everything by hand, a long and tedious process. Physical simulations can aid the animators in automatically generating physically plausible animations to start with. These animations can then be later adjusted and modified as necessary but a huge amount of work can be done by the simulation. These techniques and methods are only secondarily focused on the simulation time, while this is still a consideration to enable faster turn around times for the animator playing with different ideas the primary focus of these systems is the controllability of the output. These system should enable the animators to define their desired behavior and letting the system automatically figure out how to simulate everything in between, always trying to have an at least physical plausible believable result.

Physical simulations in games on the other hand have to primarily focus on the real time aspect of their respective techniques and methods. Today games simulate a huge a amount of objects in an ever increasing world. All these simulations have to calculated in an extremely short period of time. The simulated physics have to share the resources with everything else required to simulate the game world including game logic, artificial intelligence and computer graphics. In order to achieve smooth 60 fps all these different subsystems have to be extremely efficient and fast. If the simulations slow down too much the immersion of the player is immediately broken when things start to move unrealistically.

Another interesting subarea of the games are the so called serious games. Here the focus of the simulated world is not to provide fun and entertainment to the player. Instead the goal is the model the real world as realistically as possible to enable training and orientation for people working in the simulated field of expertise. Prominent examples are military and emergency personnel. Simulations can be used to give these persons a chance to experience real world scenarios in a controlled environment so they can train and prepare themselves for their duties. These types of games are also used to help train medial personnel to perform surgeries on completely simulated patients first before applying the knowledge to real world cases. Providing realistic real time feedback to the users is the main focus of these types of simulations.

For all close to real time and controllable simulations rigid body simulations have been a very popular choice. Rigid body simulations enable fast and efficient simulations of a huge number of interacting bodies. This can be achieved with the assumption that all bodies in the simulation a completely rigid and can never deform. This assumption is plausible for a wide variety of applications ranging from simple car models to more complex ragdoll models. Unfortunately not all possible scenarios can be modeled with simplified rigid bodies. Objects in the real world a very often deformable and bend and stretch under the influence of external forces. Incorporating deformable simulations models into games and animations enable a whole range of new interesting scenarios. However the simulation of deformable bodies requires a lot more computing resources to achieve realistic results. Novel ways to reduce the amount of work associated with deformable simulations of the be found.

One such approach is to combine the two different simulation approaches. Here the goal is combine the very fast rigid body simulations with the slower deformable simulations in order to reap the benefits of both simultaneously. Some approaches try to simulate rigid bodies and deformable bodies side by side for each object, while other approaches try to model a single body using both rigid as well as deformable parts. A real world example of the second time is the human body itself. Here basically rigid bodies, the bones, form an underlying rigid skeleton. However in contrast to simple ragdolls the reality this rigid skeleton is surrounded by a soft layer of tissue. This soft tissue can deform when forces are applied. Modeling this seemingly obvious scenario efficiently is non trivial and a perfect match for a combined simulation model.

The focus of this thesis is to explore this type of simulation, where a rigid inner core is simulated with a soft outer layer of deformable material. The goal is have an efficient simulation with is both fast and robust in scenarios where only using a simplified rigid body simulation would give unsatisfactory results. The thesis furthermore focusses on a specific technique to implement the deformable material. This technique uses oriented particles to simulate the deformations of a body. Oriented particle can be simulated efficiently and in realtime while still giving physically plausible results.

The thesis first gives a short background on how rigid body simulations are implemented and how the simulation is used in the proposed method. After that a more in-depth description of the way oriented particles are implemented is presented, beginning with position based dynamics and shape matching and explaining how these approaches enable oriented particles. The final part describes the method and ideas for combining these two different approaches into one system.

2 Problem Anaylsis

This thesis focusses on a concrete example scenario. This helps to better understand the implications and advantages of the proposed method. Also the obtained results are more easily compared against the traditional approach.

A widely applicable scenario for modeling bones and soft tissue are fingers. The scenario being used in this thesis involves two fingers grabbing onto and lifting block of the ground. The fingers are positioned to the either side of the block in the model. While applying a continuous pressure onto the block the fingers are lifting it up from the floor and into the air. This is directly applicable to a number of different scenarios involving a robotic arm with attached fingers, which is supposed to pick up blocks are other objects of the ground.

image

illustration of scenario

image

All this has to be able to be simulated in realtime in order to be used games and animations. The simulations also has to be robust against numerically instabilities. These are either caused by simple numerical or rounding errors or by the fingers being slightly off the target when grabbing onto the block. Forces between the inner bone and outer tissue as well as forces between the tissue and other objects have to be modeled so a realistic result can be achieved. This is especially true for the friction caused by the tissue on the block, as otherwise the block would not be lift up at all.

Using a simple rigid body simulation for this is suboptimal. Both fingers and the block would be modeled as rigid bodies. Resolving all the resulting forces acting on the block can quickly get numerically unstable. The blocks can start to move and wiggle incorrectly until the results are no longer visually pleasing and physically plausible.

The problem is further complicated when instead of a simple block any kind of round object like a cylinder would have to be picked up. This would require the fingers to be touch the cylinder perfectly perpendicular and on the exact opposite points. Otherwise the pressure and the resulting forces would simply let the cylinder slip out either in front or the back. This is basically impossible to achieve numerically.

The way to traditionally solve this problem has been to use multiple rigid bodies to simulate the each finger to get a better grip on the object that is lifted up. This way complicates the simulations quite a bit as joints and and many more blocks have to be simulated. However the problem of slowing increasing numerically instabilities would still occur and might still be very hard to handle efficiently.

Solving the problem with only deformable bodies is theoretically possible, however this would not really capture the real world idea of a rigid bone structure with a surrounding soft layer of tissue. Simulating everything as a deformable body would not be very efficient. This would require a lot more computational resources, especially when considering to apply this to a complete skeleton. Great care has also be taken to ensure that some deformable parts are mostly behaving as rigid structures while still giving a physically plausible result.

Therefore the main requirements for the proposed technique of combining an inner core of rigid bodies with a soft layer of surrounding deformable material can be summarized as:

- 1. realtime simulation
- 2. low number of bodies
- 3. only approximate orientation of the fingers relative to the block
- 4. collision handling/force propagation inside the body between "bone" and "tissue"
- 5. modeling friction between fingers and cylinder

3 Related Approaches

- Some(**concrete reference missing**) focus on simulations which only aim to simulated both rigid bodies and deformable bodies in the same world. But any body is either rigid or deformable. So the inner forces and collisions between bones and tissue are not really modeled and explored.
- Other(**concrete reference missing**) use non realtime methods to simulated deformable bodies and are thus not applicable to the realtime simulation and problems involved when using oriented particles.
- The state of the art thus does not provide realtime simulations of soft tissue directly attached to rigid bones.

4 Fundamentals

This section describes the fundamental buildings blocks of the algorithm described in this thesis. It provides a quick overview and explanations of how the two different simulation techniques work. It does not describe in any way how the two different simulations might interact but instead describes them in total isolation.

4.1 Rigid Body Simulations

Rigid bodies simulations is very similar to simple particle simulations. Simple particles are modeled as infinitesimal point masses, rigid bodies additional are defined by their dimension. This extension in space remains constant through out the whole simulation. A rigid body can therefore be thought of like a set of many particles. If a force is acting on any one of these point masses all the other connected masses in the rigid body are affected as well. Each point mass has a fixed mass m_i and is located at fixed position r_i inside the system. The total mass of the rigid body is then simply the sum of all the point masses combined

$$M = \sum_{i=1}^{N} m_i$$

The position of the particles is defined relative to a local body coordinate system. The origin of coordinate system is the center of mass of the rigid body. The center of mass is calculated as

$$c = \frac{\sum m_i r_i}{M}$$

In order to describe a rigid body simulation two variables are of central importance. First is the position $x(t)$ of the rigid body in space. The second is the orientation $q(t)$in space. Both these variables are defined in relation to the world coordinate system. The quaternion $q(t)$ is used to represent the rotation, as quaternions offer a number of practical advantages over rotation matrices during the simulation. By definition only the affine transformations of rotation and

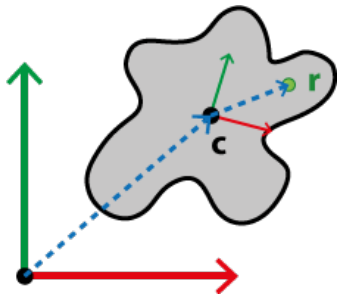

Figure 4.1: Rigid body model illustration

translation can apply to any rigid body. The main task of the rigid body simulation is therefore to calculate these to variables over time t.

The physics behind the rigid body simulation is based on Newton's laws of motion. Most importantly the first law stating that if no force is acting on an object, the velocity of the object will remain constant. Thus follows that the simulation has to model all forces acting on the body. These forces are the only causes for a change in the bodies state and thus its position and orientation in space.

- Two parts of force, Linear and Angular
- Formulas for Linear and Angular forces
- Integrating velocities over time
- Using impulses to directly model velocities
- Overview rigid body minimal state variables
- Simple simulation loop

4.2 Shape Matching

Shape Matching is an geometrically motivated approach to simulating deformable objects. It describes objects as a collection of points and does not need connectivity informations. By trading in physical correctness the technique is able to provide an unconditionally stable simulation.

The main concept of this deformable model is to replace the system of forces and energies between the different particles by simple geometric constraints and distances. All points are moved to goal positions which are obtained by shape matching the original rest state of the object to the currently deformed state of the object. These well-defined goal positions are computationally easy to obtain and provide an unconditionally stable simulation.

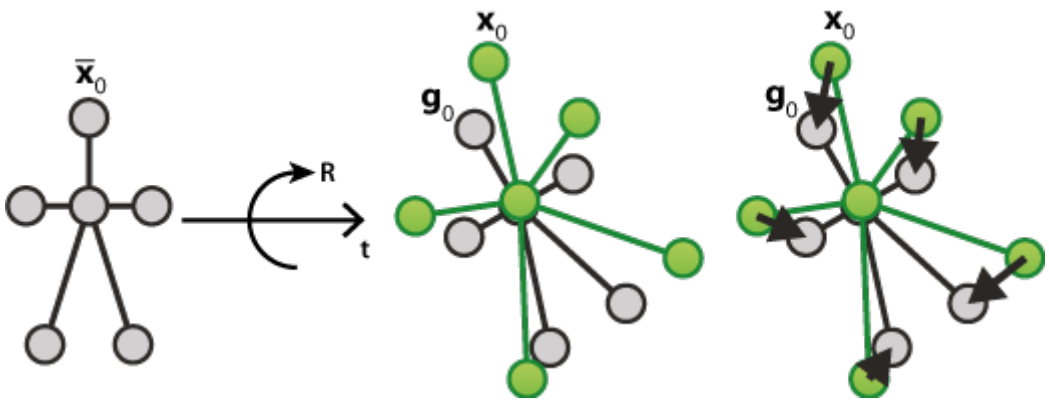

Figure 4.2: Shape matching illustration

The algorithm only takes a set of particles and there initial configuration as inputs. Each particle has four variables

$$\begin{aligned} m_i &, \text{the mass} \\ \bar{x}_i &, \text{the rest position} \\ x_i &, \text{the current position} \\ v_i &, \text{the velocity} \end{aligned}$$

The integration scheme uses the goal positions g_i to move the particles to the desired goal positions. The amount of movement can be directly modeled as a simple stiffness parameter α. A stiffness parameter if 1 thus basically doesn't allow for any deformation as the particles are always moved to their ideal transformed state. By using all these information and the external forces (i.e. gravity) the integration step becomes:

$$v_i(t+h) = v_i(t) + \alpha * \frac{g_i(t) - x_i(t)}{h} + h * f_{ext}(t)/m_i \tag{4.1}$$

$$x_i(t=h) = x_i(t) + h * v_i(t+h) \tag{4.2}$$

The goal position g_i is obtained by shape matching the rest shape to the current deformed shape. The problem shape matching solves is thus defined by finding the rotation matrix R and the translation vector t between the two sets of points $\bar{x}_i$ and x_i. The desired matrix R and vector t minimize the following term:

$$\sum_i m_i (R(\bar{x}_i - \bar{t}) + t - x_i)^2 \tag{4.3}$$

The optimal translation vectors are simply the respective center of masses of the rest shape and the deformed shape.

$$\bar{t} = \bar{c} = \frac{\sum_i m_i \bar{x}_i}{\sum_i m_i}, t = c = \frac{\sum_i m_i x_i}{\sum_i m_i} \tag{4.4}$$

Unfortunately the optimal rotation matrix is not as simple to compute. The equation **??** has to be simplified. The first step is to define relative positions for all points with respect to their center of mass.

$$q_i = \bar{x}_i - \bar{c}, p_i = x_i - c \tag{4.5}$$

$$\sum_i m_i (Rq_i - p_i)^2 \tag{4.6}$$

The next insight is to actually find the optimal linear transformation A instead of the optimal rotation matrix R. Calculating the derivative with respect to all coefficients of A and setting it to zero gives the optimal linear transformation.

$$A = (\sum_i m_i p_i q_i^\top)(\sum_i m_i q_i q_i^\top)^{-1} = A_{pq} A_{qq} \tag{4.7}$$

A_{qq} is a symmetric matrix can therefore contains no rotational part. All the rotation is encapsulated in A_{pp}. The optimal rotation matrix can now be found by calculating the polar decomposition $A = RS$. This allows to finally write down the goal position for each particle as

$$g_i = R(\bar{x}_i - \bar{c}) + c \tag{4.8}$$

4.3 Position Based Dynamics

Position based dynamics is a method which omits the traditional force based approach simulate dynamics systems but instead directly works on the positions to simulate deformable objects. This means it also is not physically correct but instead provides an efficient unconditionally stable simulation. Like shape matching it also tries to satisfy constraints in order to arrive at stable end positions. But instead of one global constraint, the shape, position based dynamics handles many different constraints. These constraints either model the physical properties of the object and are thus fixed throughout the simulation, or are generated on demand in the case of collision constraints used to resolve collisions. Also as the name implies position based dynamics only works with the position of the particles updating them directly. The velocity of the particle is solely determined by the difference in positions overtime. This is best illustrated by the the pseudocode of the simulation loop.

The first key insight into the algorithm occurs in lines 9-11. Here positions for all vertices are estimated using the current velocity and the time step. This step is completely unrestricted and just predicts and estimates a position which is then later redefined. Line 15-17 illustrate this refinement. A Gauss-Seidel type iteration is used to satisfy all constraints defined for the system. These constraints mostly model the inherent structure of object currently simulated but another important aspect handled here are collisions with are just another constraint for the system and generated in each time step after estimating the position. The idea is to iterate over all constraints multiple times so that the particles are projected to valid locations with respect to the given constraint. The final vital piece of the algorithm can be seen in lines 18-21. After all constraints have been satisfied as best as possible the estimated positions alone are used to update the state of all vertices. The velocity of the particle is solely calculated by the difference of the new estimated position and the former position. This integration scheme is unconditionally stable as it doesn't just extrapolate into the future like traditional explicit schemes do. Instead it uses the physically valid positions computed with the constraint solver.

4.4 Oriented Particles

- Oriented particles combine shape matching and position based dynamics approach
- Improves upon Shape Matching especially in areas of low density
- Shape Matching is extend with orientation information for each particle

Algorithmus 4.1 Position based dynamics simuation

1: **for all** vertices i **do**
2: initalize $x_i = \bar{x}_i, v_i = \bar{v}_i, w_i = 1/m_i$
3: **end for**
4: **loop**
5: **for all** vertices i **do**
6: $v_i \leftarrow v_i + \Delta t w_i f_{ext}(x_i)$
7: **end for**
8: dampVelocities$(v_1, ..., v_N)$
9: **for all** vertices i **do**
10: $p_i \leftarrow x_i + \Delta t v_i$
11: **end for**
12: **for all** vertices i **do**
13: generateCollisionConstraints$(x_i \rightarrow p_i)$
14: **end for**
15: **while** $i \leq solverIterations$ **do**
16: projectConstraints$(C_1, ..., C_{M+M_{coll}}, p_1, ..., p_N)$
17: **end while**
18: **for all** vertices i **do**
19: $v_i \leftarrow (p_i - x_i)/\Delta t$
20: $x_i \leftarrow p_i$
21: **end for**
22: velocityUpdate$(v_1, ..., v_N)$
23: **end loop**

- Enables simulation of extremely sparse regions with fewer particles
- Describing moment matrix for body
- Derive moment matrix for combination of multiple particles
- Derive moment matrix for single particle
- Instead of using the goal positions to get velocities, PBD approach of calculating velocities from difference between new position and old position
- Description of the initial prediction step for both position and orientation
- Description of shape matching constraints and implicit shape matching
- Resolving collisions between particle and plane
- Resolving collisions between particle and particle
- Using Gauss Seidel type iterations to satisfy all shape matching constraints
- Updating the current state using integration scheme
- Handling friction friction in PBD

5 Solution Details

This section describes the concept and approach of the solution proposed in this thesis. It first explore the conceptual construct describing the new simulation model. This describes how the different simulations are combined and work together in the proposed system. After that the practical implementation details are described in more depth.

5.1 Conceptual Approach

The simulation tries to combine a traditional rigid body simulation with the simulation of deformable bodies using oriented particles. The idea is to have an object which consists of both a rigid core and a surrounding soft layer. Both these layers will be simulated using their respective dynamics system. Both the interaction of such a combined body with the world as well as the interaction between the two layers is explained using the proposed technique.

5.1.1 Simulation Model

Every object in the simulation is either a combined body or a simple rigid body. The simulation model describes how these different bodies are modeled and constructed in order to enable the simulation and interaction between them. This requires both a model for rigid bodies as their used both alone as well as at the core of the combined body. Oriented particles are used for the deformable parts. Combining these two models into a unified combined body is the main contribution of this thesis.

Rigid Body

Simple rigid bodies are used to simulate objects which do not have or need a soft surrounding layer. Most prominently the object being picked up is for simplicity sake a simple rigid body. Also the ground doesn't need to be deformable and is thus simulated as a rigid body. These usually static rigid bodies are simulating using the traditional rigid body dynamics. They have all the usual properties of rigid bodies. The position, orientation, linear velocity and angular velocities are simulated over time. External forces, like gravity, are acting on the bodies and cause a change in velocity. The velocities are then updated in the collision resolution step using impulses. Finally the position and orientation are integrated and updated.

Oriented Particles

The oriented particles form the deformable tissue of every combined body. They are simulated exactly like the proposed by Mueller et al.. All particles have position and velocity. After updating the velocities shape matching is used to arrive at an optimal configuration of the deformed particles. The position of the particles are than moved towards this configuration. In

the final step new velocities are calculated for each particle based solely on the new position of the particle.

Combined Body

Combined bodies are more complex. These bodies are a combination of a rigid inner core with a soft outer shell. The rigid inner body behave exactly like a traditional rigid body or other non combined bodies in the simulation

- Rigid inner body core behaving exactly like a traditional rigid body
- Position, orientation, linear velocity and angular velocity are simulated over time
- External forces like gravity act directly on rigid body and alter its state
- Surrounding oriented particles are divided into two groups
- First a group of particles with are evenly distributed across the surface of the rigid body
- These particles are called attached particles
- Attached particles provide the means to exert forces on the rigid body if the tissue is moved relative to the inner core
- The other group of oriented particles is arranged in any possible configuration around the attached particles
- These particles are called outer particles
- Outer particles are used to model the shape and form of the surrounding tissue
- The simulation places no limitation on this shape
- All particles, both attached and outer particles, are simulated according to the approach in the oriented particles paper
- The shape match constraints are implicitly defined by the connection between the different particles

5.1.2 Collision Handling

- Two different types of collisions
- 1. Collision with other objects in the world
- 2. Collision between outer soft shell with rigid inner core
- Collisions with other objects is handled the exact same way as other described in the original oriented particle paper.
- Taking advantage of the elliptical shape of the particles they are simply moved to a non penetrating position

- The collision response of rigid bodies colliding with particles is not modeled as all objects are either moving and have therefore a soft tissue around them or are completely static and thus do not react to impact forces
- Collision between two rigid bodies are not directly modeled as all moving objects have soft tissue around them.
- Inner collisions work similarly but are only evaluated for the attached particles on the surface of rigid core
- In addition to penetrating of the inner core by the tissue pulling of the tissue has also be modeled and will be described in the the next section

5.1.3 Force Propagation

- Simple collision correction of the attached particles is not enough for the combined system
- The movement of the surrounding tissue can both create push and pull forces
- After the shape matching iterations only the attached particles are looked at
- For each particle the relative error between its final predicted position and its fixed position on the surface of the rigid inner core is calculated
- The magnitude of this error creates an impulse on the surface of the rigid body
- The impulse is applied to the fixed position of the attached particle on the surface
- After having applied this impulse the attached particles are directly moved to their fixed position.

5.2 Implementation

- Implementation combines rigid body simulation and oriented particles
- Using bullet library for the managing the rigid body state
- Includes forces, integrating velocities, and predicting unconstrained motion
- Bullet is also used for applying impulses generated by the tissue and finally updating the rigid bodies state
- Oriented particles implementation is completely custom

5.2.1 Setup

- Settings up the environment
- In total four bodies are simulated
- ground, 2 fingers, cylinder

- ground is a simple static rigid body and does not move are react to force during the simulation
- fingers and cylinder are all combined bodies
- Setting up a combined body
- Each inner rigid body has a arbitrarily chosen mass
- The underlying geometric structure is used to evenly distribute attached particles across its surface
- The density of the attached particles can be controlled by a configurable $density$ parameter
- For each attached particle on the surface a corresponding outer particle is generated
- The outer particle is positioned along the same vector as its corresponding attached particle relative to the center of mass of the rigid inner core
- The distance between attached and outer particle can be controlled by a configurable $extrude$ parameter
- After all the particles have been added to the surrounding tissue the implicit shape matching groups are generated
- The size of each group is a configurable $groupSize$ parameter
- For each particle the $n = groupSize - 1$ closest particles using simple euclidean distance are added to an implicit group with this particle at the center
- In addition to the closest particles the corresponding attached or outer particle is also added to the group
- Thus the group size for each particle becomes exactly $groupSize$

5.2.2 Loop

- For all active bodies gravity is applied and the velocities updated
- After that an unconstrained motion is predicted
- Both the rigid body is temporarily moved to a new position as well as all the particles belonging to the body
- This predicts both the positions and orientations
- Predicted position of particles are adjusted in case of potential collisions
- Gauss-Seidel type iterations are performed a configurable number of times in order to resolve the shape matching constraints
- After arriving at the final shape matched configuration of predicted position and orientation for the particles, the error between the fixed position of attached particle and their predicted position is evaluated

- For each error an impulse is applied to the predicted transform of the rigid body
- After applying the impulses the predicted position and orientation of the attached particles is forcefully set to their fixed position and orientation relative to position of the rigid body
- In the final step the the position/orientation of the rigid body and all particles is updated to their new values

6 Evaluation

- Describing scenario
- Two fingers try to pick up a cylinder of the floor
- Pressure onto the cylinder results in friction and the ability to pick up the cylinder
- Comparing traditional simple rigid body simulation with new combined body approach
- Simulation using three simple blocks of rigid bodies
- Simulation fails because the cylinder is pushed out of the two fingers
- Simulation using the proposed combined bodies
- Tissue wraps around are cylinder providing a larger contact area
- The friction at contact area enables fingers to pick up cylinder

7 Conclusions