Simulating rolling and colliding balls on freeform surfaces with friction

Arne Lakså, Børre Bang
Narvik University College

Abstract

We introduce an interactive REAL-TIME simulation of rolling balls on freeform parametric surfaces. The simulation includes solid and hollow balls with different size, moment of inertia, slip, rolling friction, collision detection and handling. We consider examples with several hundred balls of different types rolling and colliding on a freeform surface limited by a closed boundary set of walls. We introduce a time discretization method which is a combination of a global nonuniform -computation time dependent- time stepping and a local nonuniform time discretization based on time-sorted event handling. Finally, we discuss some computational aspects. The examples are done on a standard workstation. We use C++, OpenGL and GM_lib, an in-house geometric and graphic modeling library developed at Narvik University College.

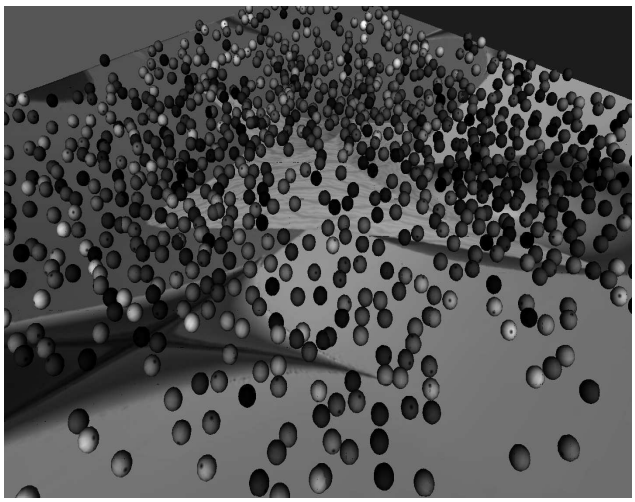

Figure 1: A system of 1000 rolling, sliding and flying balls on a freeform parametric surface bounded by 4 vertical planes. If the available space for the balls are small, and the friction close to zero, as in' this example, the system looks like it is "boiling".

1 Introduction

This paper deals with simulations and visualization of n-body dynamical systems where n is large (typical, in the range $1 \leq n \leq 1000$). Our main goal is not "high precision" simulation (minimizing errors) but "realistic" interactive real-time simulation and visualization of an unstable scenario typically containing a large number of singular events (a holonomic dynamical system approximating non-holonomic chaos) as in figure 1. This simulation address several problems, the mathematical model of motion, including holonomic models of gliding and rolling friction, holonomic restriction of a "hilly" surface given in parametric form, real-time numerical implementation of the model,real-time interactive 3D visualization, handling in a consistent way huge numbers of singular events, etc. The singularities can be either smooth (such as a ball laying still) or non-smooth (such as an elastic collision between two balls or the elastic bouncing of a ball off the surface or its boundary walls).

We use the following definitions, notations and restrictions:

- The surface $S \subset \mathbb{R}^3$ is assumed to be regular. To simplify the model we have restricted any principal curvature on S not to be bigger than the curvature of the biggest ball. The simplification implies that there is only a single contact point between the ball and the surface. However tests have shown that in our model a big ball will move apparently well on a surface by "waddling". The parametrization of the surface is defined by the map $\mathbf{s} : U \subset \mathbb{R}^2 \rightarrow S$.

- The trace of the contact point between a ball and the surface is $\alpha : \left[0, \widehat{t}\right] \rightarrow S$, where $\alpha(0)$ is the start position of the ball, and $\alpha(\widehat{t})$ is the position at time $\widehat{t}$. We also define the track curve in the parameter space: $\gamma : \left[0, \widehat{t}\right] \rightarrow U$, where $\alpha(t) = \mathbf{s} \circ$

$\gamma(t)$. It follows that the respective derivatives are:

$$\begin{aligned} \alpha'(t) &= d\mathbf{s}_{\gamma(t)}(\gamma'(t)), \\ \gamma'(t) &= (d\mathbf{s}_{\gamma(t)}^{\mathsf{T}} d\mathbf{s}_{\gamma(t)})^{-1} d\mathbf{s}_{\gamma(t)}^{\mathsf{T}}(\alpha'(t)), \end{aligned} \quad (1)$$

where $d\mathbf{s}_{\gamma(t)} \in \mathbb{R}^{3\times 2}$, and the usual notation for matrix inversion and transposition is used.

- The trace of the center of the ball is $\beta : [0,\widehat{t}] \rightarrow \mathbb{R}^3$, and is defined by:

$$\beta(t) = \alpha(t) + r\mathbf{n}_{\alpha(t)}, \quad (2)$$

where r is the radius of the ball, and $\mathbf{n}_{\alpha(t)}$ is the unit normal to the surface S at the point $\alpha(t)$. The curve $\beta(t)$ will by definition lie on an offset surface to S. This offset surface is regular because of the restrictions on the curvature of S. The two surfaces are diffeomorphic. Two related points on the respective surfaces have parallel tangent planes. Resulting in a one-to-one map between β and α, and thus, γ. The velocity of a ball at a given time t is $\mathbf{v} = \beta'(t)$ where:

$$\beta'(t) = \alpha'(t) - rd\mathbf{s}_{\gamma(t)}(d\mathbf{n}_{\alpha(t)}(\gamma'(t))), \quad (3)$$

and where $\beta'(t)$ is zero only if $\alpha'(t)$, thus, $\gamma'(t)$ is zero. (The differential geometry notation and the choice of sign of curvature are as in [3].)

- Although we later in this article discuss elasto-plastic deformation in connection with rolling balls and modeling of forces, the influence on the tracks will be so small that we in the computations only use the original surface to find the contact point, tangent plane and normal on the surface. We do not intend to make an advanced reliable friction model (see [8]), but be able to do experiments with controlled loss of energy. In this article we assume a consistent set of units, and our implementation use the SI system.

In the following we first describe our mathematical model for the movement of a ball on a surface with friction, then the numerical and algorithmic interpretation of the same problem. There are some simplifications in the mathematical model, like elasticity of collisions, rigidity of balls, neglecting influence of air and using slip and rolling frictions models based on friction coefficients. The numerical implementation of the model includes finding the closest point on a surface to a ball, a friction model based on Gaussian curvature of the difference surface between the ball and the surface. The algorithm is highly depending on a consistent time discretization including sorting of "singular" events in time.

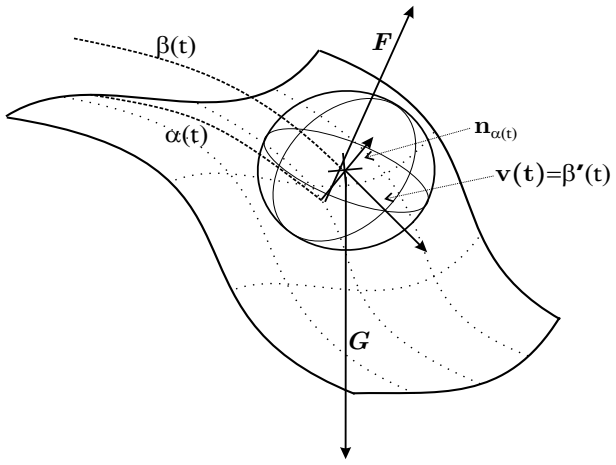

Figure 2: A rolling ball on a surface. The four vectors (arrows) in the figure are the velocity $\mathbf{v}(t) = \beta'(t)$, the surface normal $\mathbf{n}_{\alpha(t)}$, the force of gravity $\mathbf{G}$, and the force $\mathbf{F}$ from the ground on the ball. The track of the contact point $\alpha(t)$, and the track of the center of the ball $\beta(t)$ are also shown in the figure. Remark that $\mathbf{F}$ and $\mathbf{n}_{\alpha(t)}$ start on the surface, but slightly apart from each other. They usually do not span a plane. $\mathbf{G}$ and $\mathbf{v}(t) = \beta'(t)$ start at the center of the ball.

2 A ball moving on a surface

Suppose a ball is rolling on a surface, driven only by the force of gravity $\mathbf{G}$. If we neglect air resistance, the only other force affecting the ball is the force $\mathbf{F}$ generated by the contact with the surface. Simplifying the system we think of the two field of forces as two single vectors applied at each point, $\mathbf{G}$ at the center of the ball, and $\mathbf{F}$ at a point on the surface of the ball slightly in the velocity direction of the "attacking" point of the surface normal going through the center of the ball (see Figure 2). The reason for this choice of contact point is that the surface normal is derived from an un-deformed surface, while the real contact point is slightly ahead of this normal due to the deformation connected with the rolling friction. At a given time, t, the velocity of the ball $\mathbf{v}(t)$ is either a zero vector or orthogonal to the surface unit normal $\mathbf{n}_{\alpha(t)}$ at the present contact point $\alpha(t)$ between the surface and the ball. We therefore define the moving "direction"

$$\widehat{\mathbf{v}}(t) = \begin{cases} \frac{\mathbf{v}(t)}{|\mathbf{v}(t)|}, & |\mathbf{v}(t)| > 0 \\ \mathbf{v}(t), & |\mathbf{v}(t)| = 0 \end{cases} \quad (4)$$

We first decompose $\mathbf{G}$ as a vector $\mathbf{G}_t$, tangential to the surface, and a vector $\mathbf{G}_n$, normal to the surface (see Figure 3). Similarilly we decompose $\mathbf{F}$ into a vector $\mathbf{R}$

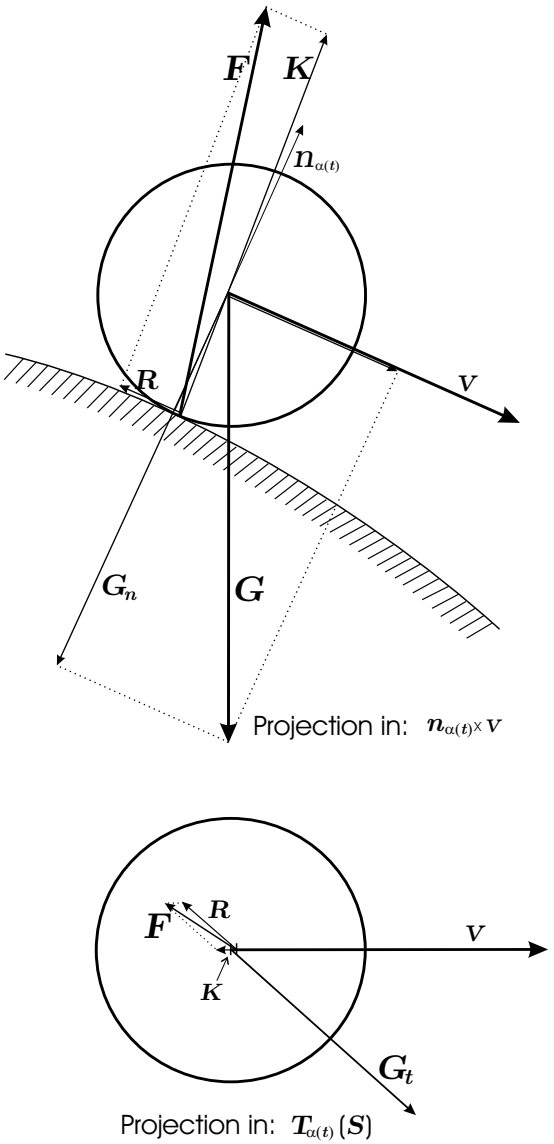

Figure 3: The picture shows two projections of a ball rolling on a surface. At the top, the ball and force/velocity vectors are projected in a plane spanned by the normal vector $\mathbf{n}_{\alpha(t)}$ at the point $\alpha(t)$ on the surface, and the velocity vector $\mathbf{v}$ of the ball. At the bottom the ball and the vectors are projected in the tangent plane $T_{\alpha(t)}(S)$ at the point $\alpha(t)$ on the surface S. The force of gravity $\mathbf{G}$, and the force $\mathbf{F}$ from the ground on the ball, and the velocity of the ball $\mathbf{v}$, are shown as vectors in the figure. Note that these vectors are not in the same plane. $\alpha(t)$, $\mathbf{n}_{\alpha(t)}$ and $T_{\alpha(t)}(S)$ are the contact point, the normal, and the tangent plane on the un-deformed surface S respectively. The real contact point is, slightly in the velocity direction from $\alpha(t)$ due to the elasto-plastic deformation caused by contact, and the rolling friction. Remark that if the ball is sliding on the surface, the direction of $\mathbf{R}$ and thus the projection of $\mathbf{F}$ in $T_{\alpha(t)}(S)$ must be parallel to $\mathbf{v}$.

parallel to the tangent plane, and a vector $\mathbf{K}$ through the center of the ball. The attacking point of $\mathbf{K}$ can now be moved to the center of the ball. We then decompose the vector $\mathbf{K}$ into vector $\mathbf{K}_t$ parallel to the tangent plane, and vector $\mathbf{K}_n$ normal to the surface (see Figure 3). The acceleration vector, $\mathbf{a}(t)$, of the ball at time t is defined by

$$\mathbf{a}(t) = \frac{1}{m}(\mathbf{G}+\mathbf{F}) = \frac{1}{m}(\mathbf{K}_t+\mathbf{G}_t+\mathbf{R}) + \frac{1}{m}(\mathbf{G}_n+\mathbf{K}_n), \tag{5}$$

where m is the mass of the ball. In the r.h.s. of (5) the total forces are partitioned into a tangential and a normal part. The "normal" acceleration is given by

$$\mathbf{a}_n(t) = \frac{1}{m}(\mathbf{G}_n+\mathbf{K}_n), \tag{6}$$

where $\mathbf{G}_n = \langle \mathbf{G}, \mathbf{n}_{\alpha(t)} \rangle \mathbf{n}_{\alpha(t)}$ and $\mathbf{K}_n$ is determined by $\mathbf{G}_n$ and the normal curvature of the surface in the velocity direction, and can be expressed using the second fundamental form [3]. The final equation must reflect the two situations; the ball is "flying", $\mathbf{K}_n$ is zero, or the ball is on the surface, $\mathbf{K}_n$ is nonzero. Then

$$\mathbf{a}_n(t) = \eta(t)\mathbf{n}_{\alpha(t)}, \tag{7}$$

where

$$\eta(t) = \max\left(\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \rangle,\ \langle d\mathbf{n}_{\alpha(t)}(\gamma'(t)), \gamma'(t) \rangle\right), \tag{8}$$

where the gravitation $\mathbf{g} = \frac{\mathbf{G}}{m}$, $(0,0,-9.81)$, and the connection between $\gamma'(t)$ and $\mathbf{v}(t)$ is described in (1) and (3). Later in the article we will show that an algorithmic interpretation of $\mathbf{a}_n(t)$ can be done using a "closest point" algorithm.

The "tangential" acceleration is given by

$$\mathbf{a}_t(t) = \frac{1}{m}(\mathbf{G}_t+\mathbf{K}_t+\mathbf{R}), \tag{9}$$

where $\mathbf{G}_t$ and $\mathbf{K}_t$ are working through the center of the ball, and $\mathbf{R}$ on the surface of the ball. For $\mathbf{R}$ there are two possibilities; the ball has only rolling motion and the contact point between the ball and the surface is not moving, then $\mathbf{R}$ is a counter reaction to $\mathbf{G}_t+\mathbf{K}_t$ and has opposite direction, or the ball is gliding, in which case $\mathbf{R}$ is a counter reaction to the velocity direction of the contact point between the ball and the surface.

The rolling friction is the result of elasto-plastic deformation of both the surface and the ball. Using a simplified model and assuming that the ball is symmetric about the center with a friction coefficient ν_b. Further assume that the surface has a non-uniform structure and a friction coefficient $\nu_{\alpha(t)}$, then the maximum

rolling force at a given point $\alpha(t)$ on the surface is given by

$$\mathbf{K}_t = -\xi(t) m\, \widehat{\mathbf{v}}(t), \tag{10}$$

where

$$\xi(t) = (\nu_b + \nu_{\alpha(\mathbf{t})})\kappa_{\alpha(t)} \frac{|\mathbf{K}_n|}{m}, \tag{11}$$

where $\kappa_{\alpha(t)} = (k_1 - r)(k_2 - r)$ where k_1 and k_2 are the principal curvatures at the point $\alpha(t)$ on the surface and r is the radius of the ball. ($\kappa_{\alpha(t)}$ is the Gauss-curvature of the difference surface.) It follows that $\mathbf{K}_t$ is zero if the ball is "flying". The tangential part of the force of gravity is given by

$$\mathbf{G}_t = m\left(\mathbf{g} - \left\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \right\rangle \mathbf{n}_{\alpha(t)}\right) \tag{12}$$

We now define a pre-description for the "tangential" acceleration by

$$\widehat{\mathbf{a}}_t(t) = \mathbf{g} - \left\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \right\rangle \mathbf{n}_{\alpha(t)} - \xi(t)\widehat{\mathbf{v}}(t). \tag{13}$$

The rotation of the ball is described by the rotation axis $\mathbf{w}(t)$ where $|\mathbf{w}(t)|$ is the rotation speed. If the ball is rolling (the contact point is not moving), we get the following relationship

$$\mathbf{v}(t) = r\left(\mathbf{n}_{\alpha(t)} \wedge \mathbf{w}(t)\right). \tag{14}$$

If however the ball is sliding this becomes

$$\widetilde{\mathbf{v}}(t) = \mathbf{v}(t) - r\left(\mathbf{n}_{\alpha(t)} \wedge \mathbf{w}(t)\right). \tag{15}$$

The sliding friction is the result of the microstructure of the surface and the ball. Using a simplified linear model, and assuming that the ball has a uniform microstructure, with a friction coefficient μ_b and assuming that the surface has a non uniform structure and a friction coefficient $\mu_{\alpha(t)}$, depending on the position on the surface, the maximum friction force at any point $\alpha(t)$ on the surface is given by

$$|\mathbf{R}|_{max} = \mu_b \mu_{\alpha(t)} \frac{r^2}{\kappa_{\alpha(t)}} |\mathbf{K}_n|. \tag{16}$$

When rolling the force $\mathbf{R}$ is dependent on the moment of inertia. For a hollow ball the moment of inertia around an axis is

$$I = 2\pi\rho \int_{r=r_0}^{r_1} \int_{\alpha=0}^{\pi} r^4 \sin^3\alpha \, d\alpha \, dr = \frac{8}{15}\pi\rho(r_1^5 - r_0^5), \tag{17}$$

where ρ is the density of the ball, r_1 is the outer diameter of the ball and r_0 is the inner diameter. Then

$$|\mathbf{R}|_{roll} = \frac{mI}{mr^2 + I} |\widehat{\mathbf{a}}_t(t)|. \tag{18}$$

There are four states for a ball:

1. the ball is flying, $\mathbf{w}(t)$ is constant,
2. the ball is rolling, $|\mathbf{v}(t)| > 0$ and $|\widetilde{\mathbf{v}}(t)| = 0$,
3. the ball is sliding, $|\mathbf{v}(t)| > 0$ and $|\widetilde{\mathbf{v}}(t)| > 0$,
4. the ball is lying still, $|\widetilde{\mathbf{v}}(t)| = |\mathbf{v}(t)| = 0$.

The equations for state 1 are not treated further in this article. In state 2 usually $|\mathbf{R}|_{roll} \leq |\mathbf{R}|_{\max}$, and the change from state 2 to 3 (from rolling to sliding) is triggered when $|\mathbf{R}|_{roll} > |\mathbf{R}|_{\max}$. In state 3 usually $|\mathbf{R}|_{roll} > |\mathbf{R}|_{\max}$, but in a short period $|\mathbf{R}|_{roll} \leq |\mathbf{R}|_{\max}$ (can be treated as an own state). This acceleration is going on until $\langle \widetilde{\mathbf{v}}(t), \mathbf{v}(t) \rangle \leq 0$. All state changes has to be treated as "singularities" (like collisions, see section 6). Thus

$$\mathbf{R} = \begin{cases} -|\mathbf{R}|_{roll} \frac{\widehat{\mathbf{a}}_t(t)}{|\widehat{\mathbf{a}}_t(t)|}, & \text{state is 2 (rolling)} \\ -|\mathbf{R}|_{\max} \frac{\widetilde{\mathbf{v}}(t)}{|\widetilde{\mathbf{v}}(t)|}, & \text{state is 3 (sliding)} \end{cases} \tag{19}$$

The "tangential" acceleration can be expressed as

$$\mathbf{a}_t(t) = \widehat{\mathbf{a}}_t(t) + \frac{\mathbf{R}}{m} \tag{20}$$

and similarilly the total acceleration is given by

$$\begin{aligned} \mathbf{a}(t) &= \mathbf{a}_n(t) + \mathbf{a}_t(t), \\ &= \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t) + \tfrac{\mathbf{R}}{m} + \left(\eta(t) - \left\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \right\rangle\right) \mathbf{n}_{\alpha(t)}. \end{aligned} \tag{21}$$

The general equation for the velocity of a ball is

$$\mathbf{v}(t) = \mathbf{v}_0 + \int_{\widehat{t}=0}^{t} \mathbf{a}(\widehat{t})\, d\widehat{t} \tag{22}$$

and the equation for the angular velocity is

$$\mathbf{w}(t) = \begin{cases} r\left(\mathbf{v}(t) \wedge \mathbf{n}_{\alpha(t)}\right), & \text{state 2} \\ \mathbf{w}_0 + \int_{\widehat{t}=0}^{t} \mu_b \mu_{\alpha(t)} \frac{r^2}{\kappa_{\alpha(\widehat{t})}} \frac{|\mathbf{K}_n|}{|\widetilde{\mathbf{v}}(\widehat{t})|} \widetilde{\mathbf{v}}(\widehat{t}) d\widehat{t}, & \text{state 3} \end{cases} \tag{23}$$

where $\mathbf{w}_0$ is the angular velocity when the ball entered the state. The moving distance is then given by

$$\begin{aligned} \mathbf{d}(t) &= \int_{\widetilde{t}=0}^{t} \mathbf{v}(t) d\widetilde{t}, \\ &= \int_{\widetilde{t}=0}^{t} \left(\mathbf{v}_0 + \int_{\widehat{t}=0}^{\widetilde{t}} \mathbf{a}(\widehat{t})\, d\widehat{t}\right) d\widetilde{t}. \end{aligned} \tag{24}$$

Equations (22 - 24) are the basis of the stepping algorithm.

In the algorithm all computations are local, and assumes constant state during the step. Using state 2, rolling, as an example the moving distance will be:

$$\begin{aligned} \mathbf{d}(t) = \int_{\widetilde{t}=t_0}^{t_1} \Big(& \mathbf{v}_0 + \int_{\widehat{t}=t_0}^{\widetilde{t}} (\tau\, \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t) + \\ & \left(\eta(t) - \tau \left\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \right\rangle\right) \mathbf{n}_{\alpha(t)}) \, d\widehat{t} \Big) \, d\widetilde{t}. \end{aligned} \tag{25}$$

where

$$\tau = 1 - \frac{I}{mr^2 + I}, \tag{26}$$

$\mathbf{g}$ is acceleration of gravity, the friction $\xi(t)$ is given in (11), the velocity direction $\widehat{\mathbf{v}}(t)$ is defined in (4), $\eta(t)$ is defined in (8) and $\mathbf{n}_{\alpha(t)}$ is the surface unit normal at the touch point.

3 Closest Point on a surface to a ball

One part of the movement equation is in directions normal to the surface. An algorithmic interpretation is to use a closest point algorithm. Global and reliable closest point algorithms usually have several disadvantages. They are computationally intensive, complex and slow and might find more than one solution (see [7]). If a sufficiently good initial guess to the solution is available, and the surface is C^2, we can use a local iterative method to find a closest point.

Definition 1 *Given a point $\mathbf{p} \in \mathbb{R}^3$, and a surface $S \subset \mathbb{R}^3$ with a map $\mathbf{s} : \Omega \subset \mathbb{R}^2 \to S$. We define:*

$$C_S(\mathbf{p}) : \mathbb{R}^3 \to S, \tag{27}$$

to be a closest point on a surface S from a point $\mathbf{p}$. By closest point we mean that the distance:

$$d(\mathbf{p}, C_S(\mathbf{p})) < d(\mathbf{p}, \mathbf{q}), \quad \mathbf{q} \in O(C_S(\mathbf{p})) \tag{28}$$

where $O(\mathbf{p}_S)$ is a neighborhood on the surface S around the point $\mathbf{p}_S \in S$.

Suppose $\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix} \in \Omega$ and that $\mathbf{s}(\mathbf{u}) \in O(C_S(\mathbf{p}))$. Then we can use an iterative method to find the closest point on the surface S to a point $\mathbf{p}$ "sufficient close" to $\mathbf{s}(\mathbf{u})$. This involves finding $\Delta = \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix}$ such that $\mathbf{s}(\mathbf{u}+\Delta) \approx C_S(\mathbf{p})$. We first approximate the position and partial derivatives using Taylor expansion (we use the notation $\mathbf{s} = \mathbf{s}(\mathbf{u})$, $\mathbf{s}_u = \frac{\partial}{\partial u}\mathbf{s}(\mathbf{u})$ etc.):

$$\begin{array}{rcl} \mathbf{s}(\mathbf{u}+\Delta) & \approx & \mathbf{s} + \Delta u\, \mathbf{s}_u + \Delta v\, \mathbf{s}_v, \\ \frac{\partial}{\partial u}\mathbf{s}(\mathbf{u}+\Delta) & \approx & \mathbf{s}_u + \Delta u\, \mathbf{s}_{uu} + \Delta v\, \mathbf{s}_{uv}, \\ \frac{\partial}{\partial v}\mathbf{s}(\mathbf{u}+\Delta) & \approx & \mathbf{s}_v + \Delta u\, \mathbf{s}_{uv} + \Delta v\, \mathbf{s}_{vv}. \end{array} \tag{29}$$

Then we require that the distance vector, $\mathbf{s}(\mathbf{u}+\Delta) - \mathbf{p}$, is orthogonal to the plane tangent to the surface at $\mathbf{s}(\mathbf{u}+\Delta)$; i.e.

$$\begin{array}{l} \langle \mathbf{s}(\mathbf{u}+\Delta) - \mathbf{p}, \frac{\partial}{\partial u}\mathbf{s}(\mathbf{u}+\Delta) \rangle = 0, \\ \langle \mathbf{s}(\mathbf{u}+\Delta) - \mathbf{p}, \frac{\partial}{\partial v}\mathbf{s}(\mathbf{u}+\Delta) \rangle = 0, \end{array} \tag{30}$$

and finally we remove the second order terms. Then we arrive at the following linear equation which we solve for each step in the iteration

$$\mathbf{A}\Delta = \mathbf{b} \tag{31}$$

where

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix}, \tag{32}$$

.

$$\begin{array}{l} a_{11} = \langle \mathbf{s} - \mathbf{p}, \mathbf{s}_{uu} \rangle + \langle \mathbf{s}_u, \mathbf{s}_u \rangle, \\ a_{12} = \langle \mathbf{s} - \mathbf{p}, \mathbf{s}_{uv} \rangle + \langle \mathbf{s}_u, \mathbf{s}_v \rangle, \\ a_{22} = \langle \mathbf{s} - \mathbf{p}, \mathbf{s}_{vv} \rangle + \langle \mathbf{s}_v, \mathbf{s}_v \rangle, \end{array} \tag{33}$$

and

$$\mathbf{b} = \begin{pmatrix} -\langle \mathbf{s} - \mathbf{p}, \mathbf{s}_u \rangle \\ -\langle \mathbf{s} - \mathbf{p}, \mathbf{s}_v \rangle \end{pmatrix}. \tag{34}$$

This is, in general, a Newton type iteration, usually converging quadratically to $C_S(\mathbf{p})$ (see [2]), and in our implementation requires in average 3.2 iterations to achieve more than 6 digits precision.

4 Moving algorithm and control of energy

Solving equations (22 - 24) is the task of the moving algorithm. One fundamental restriction is that the state of every ball must be kept constant during the entire step. That is why change of state must be determined as precisely as possible, it is one of the criteria for deciding the time step size. The basic challenge of the moving algorithm is to integrate the acceleration term, $\mathbf{a}(t)$. In the state of "flying" the acceleration is constant, $\mathbf{g}$, and therefore easy to solve. In the state of "rolling" the acceleration [formulated in (25)] is

$$\mathbf{a}(t) = \tau\, \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t) + \left(\eta(t) - \tau \left\langle \mathbf{g}, \mathbf{n}_{\alpha(t)} \right\rangle\right) \mathbf{n}_{\alpha(t)}. \tag{35}$$

The last part of this expression is always in the normal direction to the surface. In the algorithm this part is solved using closest point. In the rest, $\tau\, \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t)$, where $\tau\mathbf{g}$ is constant, and, $\xi(t)\widehat{\mathbf{v}}(t)$ is the rolling friction (a very small part of the total acceleration), we can use simplified integration. The algorithm simplifies to first finding a preliminary step,

$$\Delta\overline{\mathbf{d}} = \Delta t\, \mathbf{v}(t) + \frac{1}{2}\Delta t^2 \left(\tau\, \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t)\right), \tag{36}$$

then moving the ball back to the surface, using closest point, to get the final moving step,

$$\Delta\mathbf{d} = C_S(\mathbf{d}(t) + \Delta\overline{\mathbf{d}}) + r\, \mathbf{n}_{C_S(\mathbf{d}(t)+\Delta\overline{\mathbf{d}})} - \mathbf{d}(t). \tag{37}$$

In order to find the change of velocity, compute

$$\Delta\bar{\mathbf{v}} = \Delta t\,(\tau\,\mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t))\,, \tag{38}$$

then restrict the velocity to the plane tangent to the surface at the new position

$$\Delta\mathbf{v} = \Delta\bar{\mathbf{v}} - \left\langle v(t) + \Delta\bar{\mathbf{v}}, \mathbf{n}_{C_S(\mathbf{d}(t)+\Delta\bar{\mathbf{d}})} \right\rangle \mathbf{n}_{C_S(\mathbf{d}(t)+\Delta\bar{\mathbf{d}})}. \tag{39}$$

Then the change of rotation is as in (23). For the other states we have:

- In state 4 (lying still) nothing is changing.
- In state 1 (flying) the equations are simpler, and the rotation is constant.
- For state 3 (sliding) the algorithm is based on the same principle as for rolling.

Even though the algorithm is very precise we introduce a correction step. Preserving and controlling the energy is very important to keep it consistent. The kinetic energy depends on velocity and rotation such that

$$\begin{aligned} E_k(t) &= \tfrac{1}{2}m\,\langle \mathbf{v}(t), \mathbf{v}(t)\rangle + \tfrac{1}{2}I\,\langle \mathbf{w}(t), \mathbf{w}(t)\rangle \\ &= \tfrac{1}{2}\left(m + \tfrac{I}{r^2}\right)\langle \mathbf{v}(t), \mathbf{v}(t)\rangle\,. \end{aligned} \tag{40}$$

Potential energy varies as

$$E_{\Delta d} = m\,\langle \mathbf{g}, \Delta d\rangle\,, \tag{41}$$

and loss of energy by friction is given by

$$E_F = m\,\xi(t)\,\langle \widehat{\mathbf{v}}(t), \Delta d\rangle\,. \tag{42}$$

To ensure the preservation of energy we apply

$$|v(t+\Delta t)| = \sqrt{\frac{2m}{m + \frac{I}{r^2}}\,\langle \mathbf{g} - \xi(t)\widehat{\mathbf{v}}(t), \Delta d\rangle + \langle \mathbf{v}(t), \mathbf{v}(t)\rangle} \tag{43}$$

thus correcting the speed of the ball and the rotation speed. The velocity direction is not changed.

5 Algorithm main part

One of the most important parts of the algorithm is consistent and efficient time-step handling. In [9] several methods are discussed; Retroactive detection (RT) is a method using small steps and checking for discontinuities. If there is a discontinuity RT steps back to that time and "position". Conservative advancement (CA), is based on "never pass a discontinuity" philosophy, and tries to step to the discontinuities. Discrete event simulation (DES), is often applied to very large models. An improved version of DES is the "Time warp algorithm" introducing local virtual time (LVT) for each process, and global virtual time (GVT), where GVT is the minimum of all LVT's.

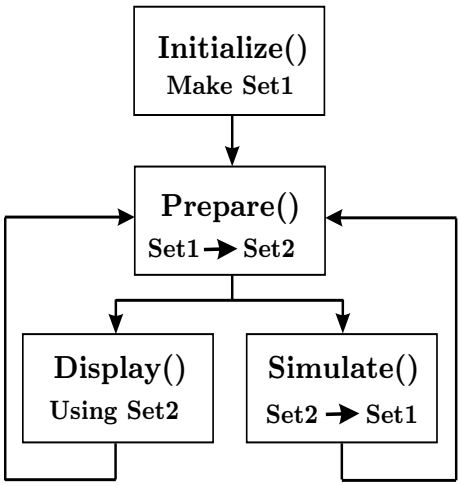

Figure 4: Diagram of the main algorithm. There are two set of positions, orientations and simulation data processed by the system, set1 and set2. Prepare() updates set2 using set1, display() uses set2, and simulate() updates set1 using set2.

In our simulation we use a method similar to the "time warp algorithm", using a global time stepping deciding the frame rate of the visualization and the precision of the simulation, and a local time discretization based on what we call "singular" events. If the simulation is supposed to be real-time there must be a global time-discretization based on the computer clock. There are two main methods used; using a uniform time-step so big that one can guarantee that the simulation and display always can be done during this time, and then using a wait statement for the rest of the time, or using a non uniform time-step, depending on the computation-time used in the previous step. The last method is not necessarily real-time because the time used in one simulation step determines the next time-step, however it has a lot of advantages. In most simulations the computation-times are generally the same for every step. This means that the method will, almost always, work as a real-time method, and at the same time be consistent if a case with more computations occur. Other advantages is that the method optimizes utilization of computation resources, and that the time-step is decided automatically. In our main algorithm the non uniform global time-step method is used:

Algorithm 1 *The main algorithm,*
initialize();

prepare();
while(not finished)
In parallel *display();*
and *simulate(time-step);*
prepare();

In the algorithm the time-step is the time used since the previous call to simulate(). The purpose of the simulation algorithm is to update the position, orientation, velocity and rotation data of each dynamic object (ball) based on the changes during the last time-step. Position and orientation is stored in the homogenous-matrix stack of the graphic system, and velocity and rotation are properties of each ball.
The algorithm is possible to parallelize, as there are two sets of position, orientation, velocity and rotation data for each object. The first set is in the local coordinate-system, and the second is in global coordinates. The simulate() function updates set 1, and prepare() updates set 2 using set 1. Both simulate() and display() use set 1 (in global coordinates) in their computations (see figure 4). Therefore simulate() and display() are able to run in parallel, while prepare() must be run separately. Thus the algorithm is well suited for hyper-threading and dual-core technology.

6 "Singular" events

"Singular" events are the fundamental elements for the algorithm of the refined local time-discretization. We define an object describing a singular event ζ_i, as

Definition 2 *ζ_i is an object describing singular events, which can be indexed and contains:*

- $x_i \in \langle 0,1]$, *time as a part of a time step,*
- *references to the balls/plane involved in the event,*
- *a state telling the type of event.*

The different type of states for a singular event is:

1. *a collision between two balls,*
2. *a collision between a ball and a plane,*
3. *a flying ball colliding with the surface,*
4. *a ball changing state (see section2).*

The relations between two elements ζ_i and ζ_j of singular event object are:

- $\zeta_i < \zeta_j$, *if* $x_i < x_j$.
- $\zeta_i = \zeta_j$, *if* *ζ_i and ζ_j have reference to at least one common dynamic object (ball).*

Definition 3 *A singular event set $\mathfrak{S}$ is a finite set containing elements of a singular event. The set has two properties:*

- *It can be ordered,*

Given a set of singular event elements. The set is ordered if the sequence of the elements are sorted based on when the event occured, and that the first element in the set represent the event occurring first.

- *It can be made unique,*

A set is unique if no pair containing two elements of the set are equal. To make a set unique the first element occurring must be kept, all the latter equal elements must be removed.

Building an ordered and unique set $\mathfrak{S}$ is only the first step in a dynamic process. To make a set empty by removal and then performing the removed event may result in inserting new elements into the set. A description of the process is:

1. remove and process the first element ζ_0,
2. insert all new events occurring
   as a result of performing ζ_0,
3. Make $\mathfrak{S}$ ordered and unique
4. If $\mathfrak{S}$ is not empty go back to 1.

This process builds a vector $\mathbf{y} = \{y_0, y_1, \ldots, y_n\}$, $0 < y_i \leq 1, \quad i = 0, \ldots, n$, which define the local time discretization by splitting the global time-step in non-uniform parts based on "singular" events.

7 Collision between two balls

For each ball, b_i, assume a step vector $\mathbf{d}_i$ describing the trace of the center of the ball during the remainder of the present time-step, is available. In a collision between two balls we start with two balls and their step vectors and we want to find a $x_i \in \langle y_j, 1]$ (the part of the time-step used when collision occur), where y_j is the "time" position of the previous "singular" event. We start with the positions of the balls, $\mathbf{p}_1$ and $\mathbf{p}_2$, the step vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, and $r = r_1 + r_2$, the sum of the radii of the two balls (see figure 6). We first compute the distance vector function,

$$\begin{aligned} \mathbf{d}(x_i) &= \mathbf{p}_1 + x_i\mathbf{d}_1 - (\mathbf{p}_2 + x_i\mathbf{d}_2) \\ &= \mathbf{a} - x_i\mathbf{b}, \end{aligned} \tag{44}$$

where

$$\begin{aligned} \mathbf{a} &= \mathbf{p}_1 - \mathbf{p}_2, \\ \mathbf{b} &= \mathbf{d}_2 - \mathbf{d}_1. \end{aligned} \tag{45}$$

Then

$$\langle \mathbf{d}(x_i), \mathbf{d}(x_i) \rangle = r^2, \tag{46}$$

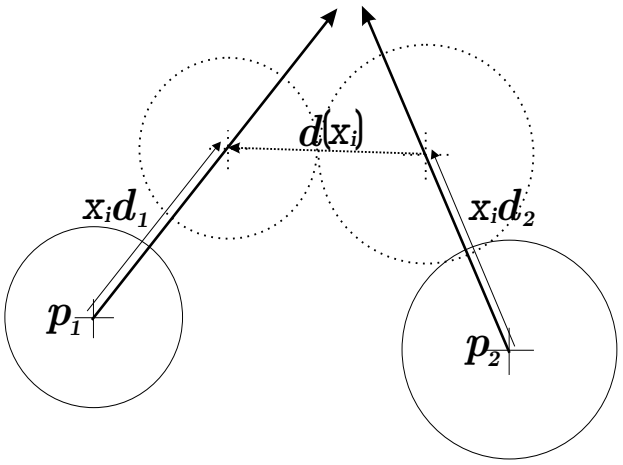

Figure 5: The figure shows two balls with center points $\mathbf{p}_1$ and $\mathbf{p}_2$, and their respective step vectors $\mathbf{d}_1$ and $\mathbf{d}_2$. Note that $\mathbf{d}_1$ and $\mathbf{d}_2$ are not necessarily in a common plane. Assuming constant speed along the step vectors, the figure shows the position of the collision between the two balls, where the distance vector $|\mathbf{d}(x_i)| = r_1 + r_2$, where r_1 and r_2 are the radius of the two balls. Note that there are two "solutions", when the balls collide, and when they have been intersecting and are leaving each other.

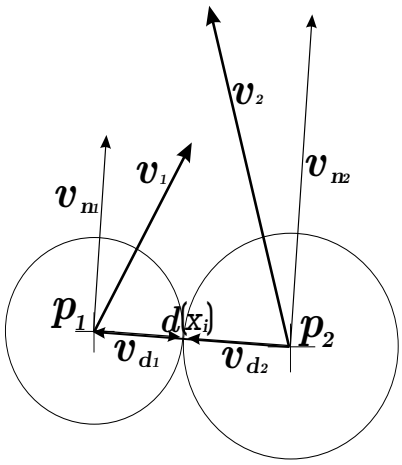

Figure 6: The figure shows two balls with center points $\mathbf{p}_1$ and $\mathbf{p}_2$, and their respective velocity vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Remark that $\mathbf{v}_1$ and $\mathbf{v}_2$ are not necessarily in a common plane. The balls are in contact and the length of distance vector $|\mathbf{d}(x_i)| = r_1 + r_2$, where r_1 and r_2 are the radii of the two balls. The velocity vectors are decomposed into a vector parallel to, and the other orthogonal to $\mathbf{d}(x_i)$.

gives the following solution

$$x_i = \frac{\langle \mathbf{a},\mathbf{b}\rangle - \sqrt{\langle \mathbf{a},\mathbf{b}\rangle^2 - \langle \mathbf{b},\mathbf{b}\rangle \left(\langle \mathbf{a},\mathbf{a}\rangle - r^2\right)}}{\langle \mathbf{b},\mathbf{b}\rangle}. \quad (47)$$

Note that we use the minimum solution to find when the collisions occurs, and not the time when the balls are leaving each other after intersecting (see figure 6). There are some problems connected to equation (47):

- if $\langle \mathbf{a},\mathbf{b}\rangle^2 - \langle \mathbf{b},\mathbf{b}\rangle \left(\langle \mathbf{a},\mathbf{a}\rangle - r^2\right) < 0$,
  no solution, the shortest distance is to long.

- if $\langle \mathbf{b},\mathbf{b}\rangle < \delta$, δ - a tolerance depending on the precision ($\approx 10^{-5}$ for single precision),
  no solution, implies that the two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ are almost equal (parallel and equal length). If additionally $\langle \mathbf{a},\mathbf{a}\rangle - r^2 < \delta$, the two balls are moving in parallel touching each other. In this situation experience has shown that the algorithm will fail. In order to prevent this, one solution is to set the two step vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ equal: $\mathbf{d}_2 = \mathbf{d}_1$.

- if $x_i \in \langle y_j, 1]$ there is a solution, where y_j is the last accepted singular event.

8 Energy exchange in ball/ball collision

A complex impact model is based on a friction model using the product of the frictions coefficient for both balls, $\mu_{bi}\mu_{bj}$. Vi first define the distance vector $\mathbf{d}$ from the center of ball 1 to the center of ball 2. If we decompose the velocity vector of a ball into a part parallel to $\mathbf{d}$, $\mathbf{v}_d$, and one orthogonal part $\mathbf{v}_n$ (see figure 6). We also get the relative velocity vector of the contact point between the two balls:

$$\bar{\mathbf{v}}_n = \mathbf{v}_{n1} - \mathbf{v}_{n2} + \frac{r_1}{|\mathbf{d}|}(\mathbf{w}_1 \wedge \mathbf{d}) - \frac{r_2}{|\mathbf{d}|}(\mathbf{w}_2 \wedge \mathbf{d}), \quad (48)$$

where $\bar{\mathbf{v}}_n$ lies in the plane tangent to the contact point of the balls. The equations are similar to, but more complex than the equations in section 2, with 2 states, sliding or not sliding, depending on the impulse in direction $\mathbf{d}$.
Because of the computational complexity (resulting in reduced framerate), we are, for the moment, using frictionless impact (see [1]). From the equation (40) we get:

$$E_k(t) = \varpi \langle \mathbf{v}(t), \mathbf{v}(t)\rangle . \quad (49)$$

where:

$$\varpi = \frac{1}{2}\left(m + \frac{I}{r^2}\right). \quad (50)$$

Using the decomposed velocities $\mathbf{v}_d$ and $\mathbf{v}_n$, we can divide the energy of this in two orthogonal directions (Pythagoras). In the impact we assume that $\mathbf{v}_n$ is unaffected. Then we look at $\mathbf{v}_d$ of both balls and treat this impact as a central impact. From equation (49) we get

$$\varpi_1(v_{d11}^2 - v_{d10}^2) = \varpi_2(v_{d20}^2 - v_{d21}^2), \qquad (51)$$

and

$$\varpi_1(v_{d11} - v_{d10}) = \varpi_2(v_{d20} - v_{d21}) \qquad (52)$$

where v_{d10} is the velocity, as a scalar, of ball 1 in direction $\mathbf{d}$ before the collision, v_{d11} is the velocity, as a scalar, of ball 1 in direction $\mathbf{d}$ after the collision, v_{d20} and v_{d21} is the before and after "scalar" velocity for ball 2. Index 1 and 2 for ϖ indicate the ball it belongs to. The resuling equations are

$$v_{d11} = \frac{\varpi_1 - \varpi_2}{\varpi_1 + \varpi_2} v_{d10} + \frac{2\varpi_2}{\varpi_1 + \varpi_2} v_{d20}, \qquad (53)$$

and

$$v_{d21} = \frac{\varpi_2 - \varpi_1}{\varpi_1 + \varpi_2} v_{d10} + \frac{2\varpi_1}{\varpi_1 + \varpi_2} v_{d20}, \qquad (54)$$

and the velocity for ball 1 after impact is given by

$$\mathbf{v}_{11} = \mathbf{v}_{n1} + \frac{v_{d11}}{|\mathbf{d}|}\mathbf{d} \qquad (55)$$

and similarilly for ball 2 after impact

$$\mathbf{v}_{21} = \mathbf{v}_{n2} - \frac{v_{d21}}{|\mathbf{d}|}\mathbf{d}. \qquad (56)$$

9 The local simulation algorithm

The next important subject of the algorithm is the refinement of the time-discretization based on singular events. This refinement is done in the simulation() function. The algorithm is based on an iterative process finding the next local time step, and based on the process described in section 6. The ♠ mark in the following algorithm indicates where the next local time step is fixed. Note that when this happens the rest of the set of "singular" events might change.

Algorithm 2 *The local simulation algorithm,*

◇ *for each ball*
  compute preliminary:
    change of position, velocity, rotation
    and record singular events found,(type 3,4)
  compute ball/plane collisions:
    and record singular events found (type 2)
◇ *for all unique combination of two balls*
  compute ball/ball collision:
    and record singular events found (type 1)
◇ *while(set of singular events is not empty)*
  make the set of singular events ordered
  make the set of singular events unique
  treat the first singular event:
    remove it from the set ♠
    make the step and update the data
    for the affected balls:
      compute new preliminary:
        change of position, velocity, rotation
        and record singular events found, 3,4
      compute affected ball/plane collisions:
        and record singular events found, 2
      for all unique combination of two balls
      where one is an affected ball
        compute ball/ball collision:
          and record singular events found, 1
◇ *for each ball*
  make the final step and update
  position, velocity and rotation.

The algorithm consists of four parts, each marked with a ◇. The order of the first and the fourth part is $O(n)$, while the order of the second part is $O(n^2)$. The order of the second part can be reduced by using some kind of sorting, but the computational cost of doing this might easily exceed the cost of the core collision detection. The third part is depending of the dynamics of the system, the number of change of states and collision detected (and thus depending of the total energy and the available space in the system). Tests done by our program has shown that even for a "boiling" system the cost of the third part is small compared to the other parts.

10 Conclusion

The strong point of our method/application is the algorithmic structure, and how it makes it possible and easy, to increase the complexity without changing the structure. It is an efficient and "complete" algorithm, with the possibility of using a large number of balls in the simulation. Together with the structure of the algorithm there are some pre-evaluations boosting efficiency. Some of the problems solved in the application, are not treated in this article, like the importance of an efficient evaluator for the surface [4], including pre-evaluation of basis functions (see [6]). The ball-plane collisions problems are mostly the same as for ball-ball collisions, so one can use similar methods. In a line-surface intersection, used in connection with a ball bouncing on the surface, one can use an algorithm

like the closest point algorithm, but on the offset surface.
There are obvious applications in 3D game engines and visualization, and as computational cost decreases it may be possible to extend this method to particle simulations and perhaps other areas such as for instance wave phenomena.
There is still a potential for improvements of the algorithm, especially regarding performance and realism. Developing an improved friction model for impacts and parallelizing the collision treatment are some of the most important issues. Using Graphic Processor Units (GPUs) for the simulation algorithm [5], with its streaming technology is also a topic for possible improvements.

References

[1] Beer, F. P., E. R. Johnston Jr., W. E. Clausen, *Vector Mechanics for Engineers, Dynamics*, McGraw-Hill, Singapore, 2004.

[2] Conte, S. D. and C. de Boor, *Elementary Numerical Analysis*, McGraw-Hill, Singapore, 1983.

[3] Do Carmo, P., *Differential Geometry of Curves and Surfaces*, Prentice Hall, New Jersey, 1976.

[4] Farin, G., *Curves and Surfaces for CAGD, A practical Guide*, Morgan Kaufmann, San Francisco, 2002.

[5] Hjelmervik, J. M. and T. R. Hagen, GPU-based screen space tessellation. In Mathematical Methods for Curves and Surfaces, Tromsø 2004, M.Dæhlen, K.Mørken, L. Schumaker (eds.), Nashboro Press, 2005

[6] Lakså, A., B.Bang, L.T.Dechevsky, Exploring expo-rational B-splines for curves and surfaces. In: Mathematical Methods for Curves and Surfaces, Tromsø 2004, M. Dæhlen, K. Mørken, L. Schumaker (eds.), Nashboro press, 2005.

[7] Lakså, A., et al, *SISL The SINTEF Spline Library Reference Manual (version 4.1)*, SINTEF Applied Mathematics, Oslo, 1997.

[8] Lebon, F., Contact problems with friction: models and simulations, in Simulation Modelling Practice and Theory 11 (2003), pages 449–463. Elsevier B. V., 2003.

[9] Mirtich, B., Timewarp Rigid Body Simulation, in SIGGRAPH 2000 Conference Proceedings, Annual Conference Series, pages 195–200. ACM SIGGRAPH, Addisen Wesly, 2000.

[10] Rudin, W., *Principles of Mathematical Analysis*, McGraw-Hill, Singapore, 1976.