Workflow for real-time simulation of deformable objects

Karolina Lubiszewska*
Supervised by: Anna Tomaszewska

Computer Graphics Group
West Pomeranian University of Technology, Szczecin

Abstract

In this paper we address the problem of content creation for physical simulation of soft body objects. We aim to optimize the process of modeling deformable bodies with complex rigid-body skeletons which can be used to visualize realistic movement and animations. Currently no efficient or standarized asset design and implementation method exists for this type of models. We propose reorganization of the present segregation of duties between designer and developer, through a new specialized data interchange file format and the use of extensible open-source designing environment Blender. Simplification of programming work is achieved without unnecessary workload addition for the content creator. Blender is used for object modeling as well as for physical properties and skeleton specification. In result a crucial part of the design workflow is extracted outside of the game engine's SDK toolkit towards independent 3D modeling tools. To evaluate the proposed method a real-time physically-based soft-body character animation is created using Nvidia PhysX and OpenGL.

Keywords: deformable objects, soft body physics, content creation, Blender, rigid skeletons.

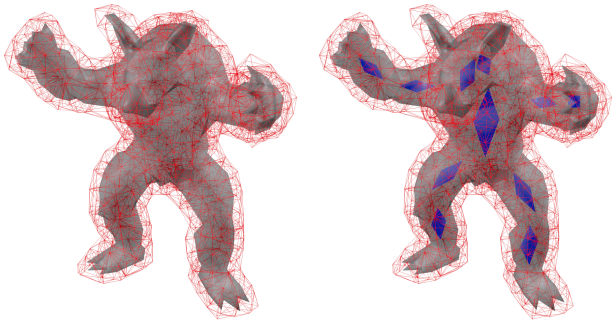
1 Introduction

When rendering real-time computer-generated 3D scenes, the goal is often to create as realistic-looking impression as possible. Under the term of realism we understand 3D models which first of all look life-like, but also behave in a physically-correct manner [4].

Different types of objects behaviour of which can be simulated exist and their simulation requires different approaches. Examples of such objects include:

- rigid bodies - firm, not changing their shape,
- soft bodies - deformable, elastic, fluids, simulating substances like metal, rubber or water.

Depending on the type of given model, for proper simulation of its physical behaviour different kinds of meshes are used. For very simple rigid objects it is enough to provide just the same polygon mesh as the one used for rendering purposes. For more complex objects often a simplified convex hull dedicated to collision detection is supplied. Soft bodies, however, require two different kinds of meshes: one built of planar polygons for rendering purposes, and the other built of tetrahedra (Figure 1a) necessary for the simulation of object's volume.



(a) (b)

Figure 1: Soft body with its tetrahedral volumetric mesh (a) and its counterpart rigged with a skeleton (b).

In the paper we focus on models which consist of both soft and rigid objects (Figure 1b). These are mainly elastic bodies built over rigid-body skeletons. Such complex way of composing and physically simulating objects is not yet popular in video games and real-time virtual reality applications. This situation is mainly caused by the computational cost regarding non-rigid bodies. Every soft object is represented by a number of tetrahedra, each of which has its own position, physical properties and constantly influences the state of every adjacent element. Computations such as the previously described are not required when simulating rigid bodies as their shape is invariable, so is their volume.

Until recently it was impossible to simultaneously simulate and visualize several such objects in real time. Currently thanks to the improving processing power of modern hardware and still improving software tools such as physics and rendering engines, we can finally simulate real soft bodies instead of using keyframe interpolation for smooth animation of deformations [8]. This fact intro-

*klubiszewska@wi.zut.edu.pl

duces completely new possibilities in depicting the reality which surrounds us.

Today even for the simplest models their description is divided into parts. Renderable mesh data and skeleton data are exported to a different file than the tetrahedral mesh necessary for soft body simulation. In addition some information has to be provided manually by the designer in descriptive documents, which are then interpreted by the programmer. A unified, universal and open format is missing.

We propose gathering all the data using an extensible file format to automate and optimize the content creation process and to simplify building simulations which make use of deformable bodies. The goal is to reduce the developer's workload without unnecessarily increasing the number of artist's, designer's and animator's responsibilities by allowing them to work with a single, well-known software suite. Such an optimization will lead to faster, more efficient transferring of created models from design software to end applications. It is most important for video games asset creation, where artist's fantasy is one of the key aspects leading to success.

In the next chapter we provide a short survey of existing approaches to the simulation of soft bodies, or methods which can be used in replacement. The third chapter covers the issues related to the current workflow and proposes our solution to the outlined problems. In the fourth chapter we present implementation-related details important when using our method. We conclude the paper with results gathered from building a test scene containing a soft body with a rigid skeleton and we compare it with a sole deformable body. In the end our future goals are described.

2 Related work

The idea of soft body simulation for computer graphics applications was proposed in the late '80s [15]. The first trials were conducted with non-real-time simulations only as real-time visualization of complex objects was far out of the scope for that time hardware.

There are many different approaches for simulating soft bodies. They were comprehensively surveyed in [5] and [10]. We have chosen the method which bases on using tetrahedral lattice mesh for representing the object's volume. The mesh consists of finite number of elements which thoroughly fill the modeled object's extent [14]. Its vertices form the topology of a mass-spring system used for simulating the body's interaction and self collisions [2]. Cost of this approach can be easily scaled to suit the needs of a particular model, scene and hardware capabilities. It makes the method useful for interactive simulations.

To address the calculation complexity related issues, a keyframe-based simplification can be used. The actual deformations are calculated off-line and only certain keyframes are exported for use in real time. These keyframes are then interpolated to resemble smooth animation [8]. This solution lacks the freedom of interaction as the object's reactions are limited only to a strictly defined set of previously prepared possibilities. To achieve the true real-time simulation we decided to perform online calculations for the whole object's volume instead of using the keyframes.

However, a body which only consists of a deformable volume is hardly controllable and classic methods for e.g. character animation cannot be used. To solve this problem a coupling between the soft body and a rigid skeleton can be used. The skeleton can react to applied forces and movement induced by methods such as reverse kinematics [7]. The use of a skeleton allows us to introduce limits for bone joints which restrict the movement that could be recognized as unnatural depending on the object's characteristics [6, 13].

The field for soft bodies application, including those equipped with a rigid skeleton, is very broad. Apart from the already mentioned character animation, deformable objects are used for modeling destruction [12] and jelly-like entities [3] in video games. Moreover physically-correct simulation of soft tissues is pursued by the numerous virtual reality applications aimed at training medical personnel [1, 9].

Using a soft body in real-time graphics visualization presents many benefits:

- preserves a consistent volume - as its finite elements influence each other,
- smooth deformations can be applied,
- is elastic - retains its previous form,
- tearing can occur - in effect it breaks the former topology and forms two or more objects.

3 Proposed content creation workflow

In order to simulate a soft body together with its rigid skeleton, appropriate input data are needed. These include not only the renderable mesh information, but purely physical properties as well. The following are necessary:

- coordinates of polygonal mesh vertices for rendering purposes,
- coordinates of tetrahedral mesh vertices for volume simulation,
- coordinates of rigid body vertices,
- information on joints and their types for coupling rigid bodies,
- limitation and spring information for joints,

- mass (for both body types),
- friction (for both body types),
- gravity response (for both body types),
- texture coordinates for renderable soft body mesh,
- information necessary for dynamic calculation of normal vectors.

3.1 Current content creation workflow

On the diagram (Figure 2) the problem of current workflow for designing soft body with rigid skeleton is pictured.

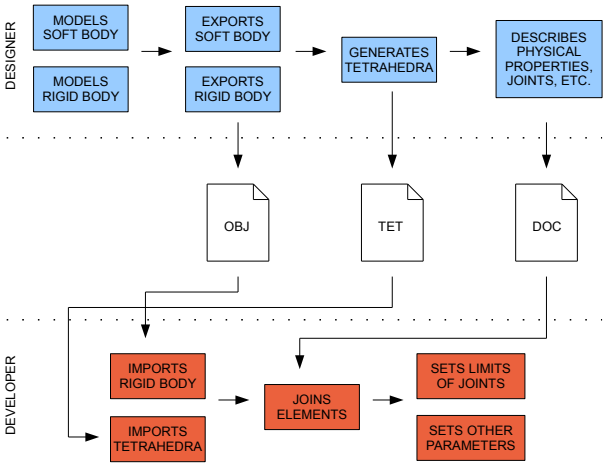


Figure 2: Current content creation workflow for designing soft body with rigid skeleton.

It can be seen that the pipeline is a multi-stage, complex process which can result in numerous misunderstandings between the design and development teams.

Currently the artist who designs the 3D model creates a mesh for both soft body and rigid skeleton. Then the parts are exported to appropriate files. Additionally he is responsible for creation of volumetric tetrahedral mesh in a separate, specialized program in order to allow the deformation of soft body structure. The next step is the description of dependencies between certain objects, the physical properties, joints etc. to achieve the desired object rigging.

For example, an elbow exposes a naturally limited freedom of swinging angle and direction. The joint prevents the bones from moving away from each other (Figure 3). These limitations have to be applied to simulated bones as well. And because of no popular, cross-platform solution, the risk of misunderstandings between cooperating persons is high.

The developer receives a set of files that have to be imported one-by-one in the end application. The additional provided information has to be implemented manually, the joints need to be set up, the physical properties and the

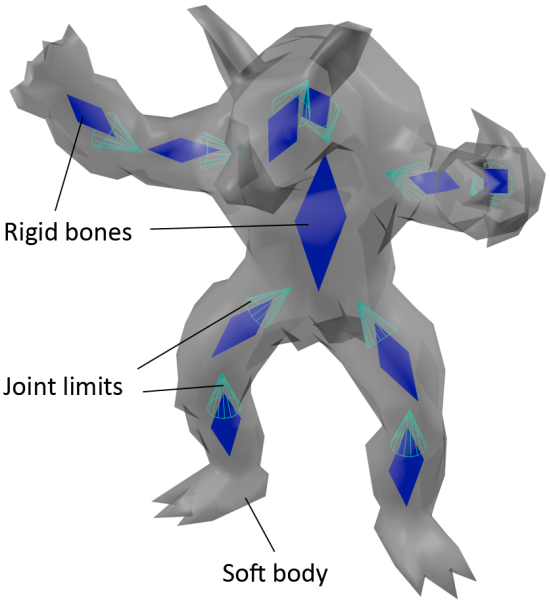


Figure 3: Stanford Armadillo skeleton with joint limits visualized as cones.

joint limitations need to be introduced according to the designer's description.

The current workflow requires the developer to possess knowledge about the model which he imports. He has to perform numerous object-dependent actions manually. The complexity of the workflow, its time-consuming aspect and proneness to mistakes does not allow for efficient use in the creation of deformable assets for real-time computer graphics applications.

3.2 Solution

The diagram (Figure 4) presents our proposed content creation pipeline for designing soft body with rigid skeleton.

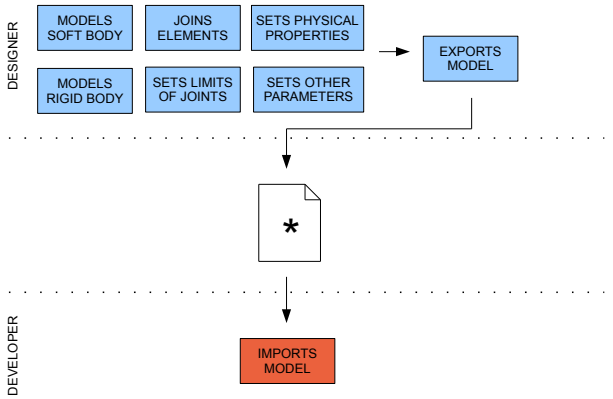


Figure 4: Proposed content creation workflow for designing soft body with rigid skeleton.

The main aim of the improvements is to reduce the programmer's workload. As the diagram pictures, the most

of the stages are now controlled by the designer. However, despite controlling the major part of the workflow, the designer is not overwhelmed with responsibilities. He describes the object's properties within the same software suite he uses for modeling. The programmer's only remaining duty is to import the resulting file to make the object available for simulation and rendering.

During the first stage the 3D artist creates meshes for objects: deformable one related to the body silhouette and rigid ones for the skeleton. Then he is able to create joints between the skeleton parts and set their limits to form the object's rigging. Physical properties are assigned to proper parts of the model and other rendering-related data such as the material and texture data can be applied as in regular objects modeling pipeline. While exporting, the tetrahedral mesh is generated automatically for the deformable volume.

Differences between the proposed and current solution are relatively insignificant from the designer's point of view, but at the same time the programmer is left with significantly less responsibilities. Instead of descriptive documentation, the artist can configure properties already in the design software (Figure 5). Finally a single file is exported for the complete complex model.

The programmer imports received data to the final application using the importer described later in chapter 4. He is not obliged to specify the parameters, join specific objects and set the limits by himself. He does not need to know anything specific about the particular model as during the import procedure all the necessary data are processed automatically. The imported model is ready-to-use.

Our solution helps the content creation automatization and reduces the necessary workload. It also eliminates the chance for misunderstandings resulting from descriptive documents.

4 Technical details

In order to test our solution we created a simple application which generates a real-time animation of Stanford Armadillo model. Our object consists of a soft body rigged with a rigid-body skeleton. Armadillo's arms are attached to invisible blocks floating in space. Bones of the skeleton are coupled using 3-degrees-of-freedom spherical joints with limited ability to swing so that the character cannot hang limply. The model is influenced by gravity and objects which can be thrown at it to visualize the deformations.

For 3D object design purposes we use *Blender 2.61* (Figure 5) software which is freely available and allows us to create our own plug-in extensions. Visualization is performed in real time by a simple *OpenGL 2.1* and *Glut* library-based rendering engine written in *C++*. *NVIDIA PhysX 2.8.1* engine is responsible for physics-related calculations and soft body simulation.

Currently we use *PhysX Viewer* which is a part of the *NVIDIA PhysX SDK* to create a tetrahedral mesh. However in the future the lattice is to be created automatically during exporting procedure of the model from design environment. We plan to use the algorithm described in [14] for this purpose.

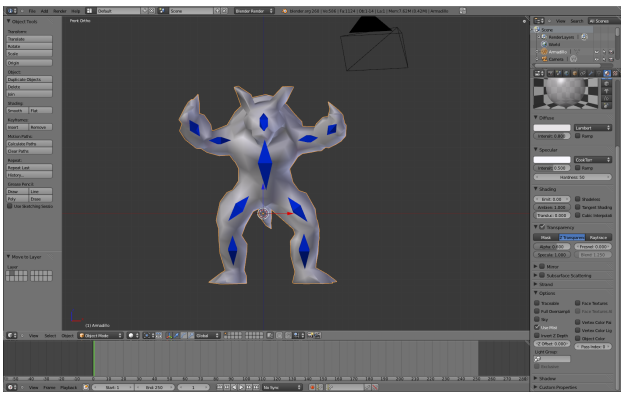


Figure 5: Creating model skeleton of Stanford Armadillo in Blender 2.61.

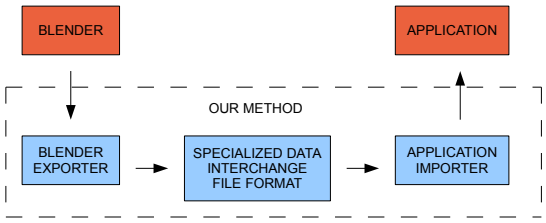


Figure 6: Implementation schema using proposed tools.

Exemplary contents of an *XML* file are shown in Figure 8. They include the contents of both standard *Wavefront OBJ* and native *NVIDIA PhysX TET* files. The file contains skeleton's polygon meshes, tetrahedral volumetric lattice mesh, renderable soft body surface polygon mesh, descriptions of skeleton joints and various physical properties of both rigid and soft parts. We use *XML* mainly to depict the hierarchy of elements which are parts of the proposed format. In the future a binary file format should be considered for final implementation to reduce the file size and improve the processing speed during import.

The whole structure is divided into parts which relate to soft body and rigid bodies.

In the soft-body section there are:

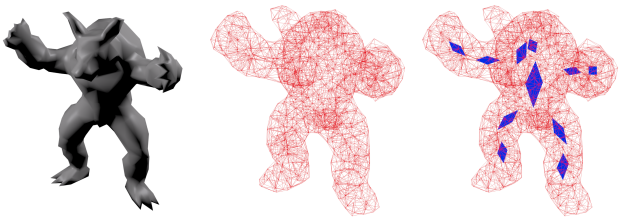
- tetrahedral mesh which describes the volume:
  - vertices
  - configurations of consequent tetrahedra
- polygon mesh for rendering purposes:
  - vertices
  - texture coordinates
  - configurations of consequent polygons

- barycentric coordinates - mapping of renderable mesh vertices into the volume of a specified tetrahedron which includes the vertex in the initial pose
- physical attributes:
  - mass
  - volume stiffness
  - stretching stiffness
  - friction
  - particle radius, solver iterations - scene-dependent values important for simulation stability

The rigid-body-skeleton section consists of:

- convex polygon meshes for collision detection:
  - vertices
  - configurations of consequent polygons
- physical properties:
  - mass
  - friction
- joints between rigid bones:
  - type-dependent values and limitations

When the model is imported, information chunks are routed to appropriate application elements: the rendering engine and physics engine.



(a) (b) (c)

Figure 7: Renderable surface mesh (a), tetrahedral lattice mesh (b), tetrahedral mesh with rigid skeleton (c).

Polygon surface mesh for the soft body (Figure 7a) together with its texture coordinates is passed directly to the rendering engine. The tetrahedral mesh (Figure 7b) is passed to *NVIDIA PhysX*, same as the convex skeleton meshes (Figure 7c). In our case the skeleton is intended for physics simulation only, it is not intended for drawing.

For every rigid bone a separate *actor* is created in *PhysX*. Each of them is coupled with other bones using a joint according to the joints section of model file. The joint can be any of the 10 types offered by *NVIDIA PhysX* [11], e.g.:

- Spherical joint;
- Revolute joint;
- 6-degree-of-freedom joint;
- Distance joint.

Every joint can have different limitations depending on its type. For example, the spherical joint can be restricted to swing or twist only in a specified angle range.

For the soft body no actor is created. Instead, an instance of a specialized *PhysX* soft body class is used. It receives the necessary parameter values from the imported model file, such as mass, stiffness and friction.

To visualize the simulation, during every animation frame the position of each renderable surface mesh vertex is updated according to the simulation results. The update is performed with the help of four-dimensional barycentric coordinates B_{V_i}, which are defined for every vertex V_i of the polygon surface mesh:

$$B_{v_i} = (v_0, v_1, v_2, 1 - (v_0 + v_1 + v_2))$$

Where the k-th element of the vector B_{V_i} we write as $B_{(V_i,k)}$. The coordinates are calculated for the initial pose of renderable mesh vertices in the volume built of tetrahedra. To obtain the current vertex position V_i' necessary to visualize the soft body, we calculate an affine combination using current position P' of vertex T_j of the tetrahedral mesh:

$$V_i' = \sum_{k=1}^{4} B_{(V_i,k)} P'_{(T_j,k)}$$

In order to calculate correct lighting of the soft body, in every frame we use the already updated positions of vertices to calculate current normal vectors N_l for every l-th face. It amounts to applying a cross product between two edges E_1 and E_2 of a given triangle built from vertices V_0, V_1 and V_2.

$$E_1 = V_0 - V_1$$

$$E_2 = V_2 - V_1$$

$$N_l = E_1 \times E_2$$

The order of vertices depends on the order in which the triangles were defined: clockwise or counter-clockwise. To achieve smooth shading, per-vertex normal vectors have to be averaged and normalized.

The soft body can be attached to its skeleton in two ways:

- one-by-one explicitly specified rigid objects are attached to the soft volume,
- every rigid object which collides with the volume is automatically attached.

Advantage of the first approach is that the coupling can be strictly defined. In the second case the attachment procedure has to take place in a controlled space, where no other object can appear by accident. Otherwise unwanted attachment can occur. But when using this way no additional data and activity is necessary for creating the coupling.

5 Results

The resulting visualization of rigid-skeleton-rigged soft body which makes use of joint limits behaves much more naturally than when not using the skeleton (Figures 9, 10). It is worth noticing that the limbs of Stanford Armadillo do not bend under angles which could look unrealistically. Without the skeleton there is no such restriction and the deformation can occur in a completely random manner. The other advantage is the ability to control the soft body with inverse kinematics, what is impossible in a pure-soft-body solution. Also the ability to set different masses to different bones in order to achieve non-uniform mass distribution improves the realism of character's reactions.

In our example the volumetric lattice consisted of approximately 2750 tetrahedra. The renderable surface mesh contained 1036 polygons. The low number of polygons resulted from the necessity to recalculate vertex positions and normal vectors every frame what makes achieving interactive frame rate challenging. While rendering two such models simultaneously on the scene we reach 18 frames per second on an *NVIDIA GeForce GTX 295* and *AMD Phenom II X4 965*.

We observed during the implementation process that it is important to match the soft body volume's particle radius with the simulation conditions like the model's size and density of the lattice mesh. Otherwise skeleton attachment can be problematic as some parts of the bones can slip in-between the particles. It is also important not to forget about setting friction high enough to avoid simulation instability. The importance increases with the particle radii as the particles begin to constantly influence each other.

6 Summary and future work

We presented a file format and workflow improvements which can lead to enhanced work efficiency during design phase of rigid-skeleton-rigged soft body simulations. Behaviour of the model which is simulated with a skeleton proved to be closer to the expected and natural than when using a sole soft body or a rigid body instead. We believe that in the future video games will benefit from using this approach for character animation instead of the currently common unrealistic solutions.

One of our future goals is to create a Blender plug-in for exporting models to our universal format (Figure 8). The exporter should allow easy definition of different physical properties and intuitive coupling of skeleton parts together with setting their attributes and limits using the Blender built-in armature interface. Also the automatic generation of tetrahedral mesh should be introduced to the exporter. Our file format should be extended to cover other important rendering-related features such as materials and textures. Interesting soft-body features such as tearing and heterogeneous materials stay in our field of interest as well.

References

[1] H. Delingette. Toward realistic soft-tissue modeling in medical simulation. *Proceedings of the IEEE*, 86(3):512–523, 1998.

[2] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 1–8, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[3] Epic Games, Inc. Unreal Engine 3 UDK: PhysX soft bodies. http://udn.epicgames.com/Three, 2010.

[4] J. Ferwerda. Three varieties of realism in computer graphics. In *SPIE Human Vision and Electronic Imaging 03*, pages 290–297, 2003.

[5] N. Galoppo. *Animation, simulation, and control of soft characters using layered representations and simplified physics-based methods*. PhD thesis, Chapel Hill, NC, USA, 2008. AAI3331034.

[6] J. Georgii, D. Lagler, C. Dick, and R. Westermann. Interactive deformations with multigrid skeletal constraints. In Kenny Erleben, Jan Bender, and Matthias Teschner, editors, *VRIPHYS*, pages 39–47. Eurographics Association, 2010.

[7] J. Kim and N. Pollard. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph.*, 30:121:1–121:19, October 2011.

[8] R. Kondo, T. Kanai, and K. Anjyo. Directable animation of elastic objects. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 127–134, New York, NY, USA, 2005. ACM.

[9] A. Maciel, T. Halic, Z. Lu, L. Nedel, and S. De. Using the physx engine for physics-based virtual surgery with force feedback. *The international journal of medical robotics computer assisted surgery MRCAS*, 5(3):341–353, 2009.

[10] A. Nealen, M. Mller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models

in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[11] NVIDIA. PhysX SDK 2.8 introduction. http://developer.nvidia.com, 2008.

[12] E. Parker and J. O'Brien. Real-time deformation and fracture in a game environment. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 165–175, New York, NY, USA, 2009. ACM.

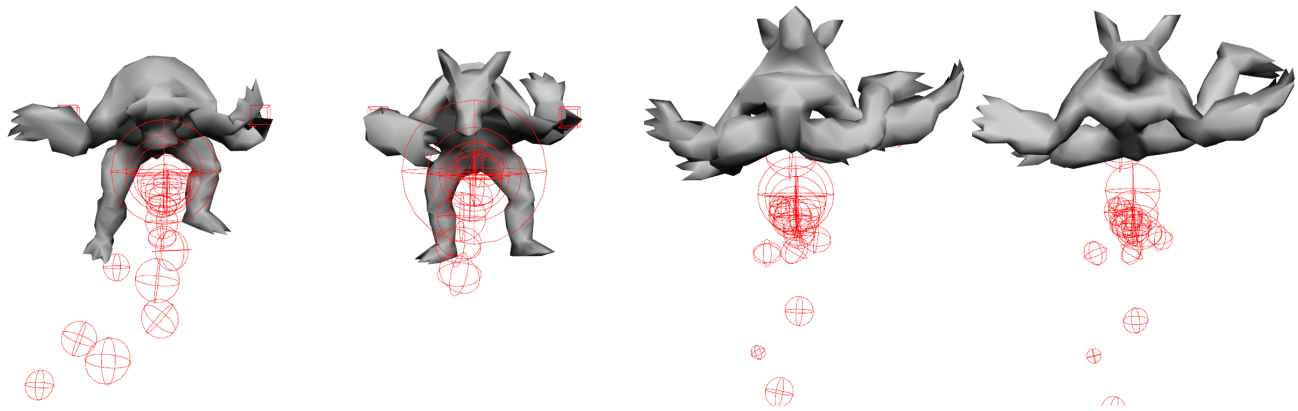
[13] T. Shinar, C. Schroeder, and R. Fedkiw. Two-way coupling of rigid and deformable bodies. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 95–103, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[14] J. Spillmann, M. Wagner, and M. Teschner. Robust tetrahedral meshing of triangle soups. In *Vision, Modeling, Visualization (VMV)*, pages 9–16, 2006.

[15] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 205–214, New York, NY, USA, 1987. ACM.

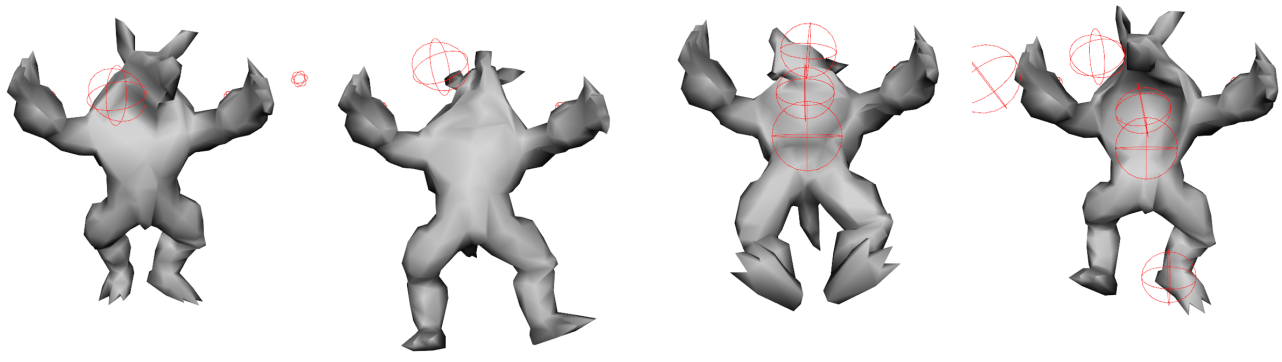
```
<?xml version="1.0" encoding="utf-8"?>
<model>
  <soft_body>
    <tetrahedron_mesh target="volume">
      <vertices>
        <vertex id="1" x="8.4" y="2.3" z="9.5"/>
        <!-- ... -->
      </vertices>
      <tetrahedra>
        <tetrahedron>
          <node v="1"/>
          <!-- ... -->
        </tetrahedron>
      </tetrahedra>
    </tetrahedron_mesh>
    <polygon_mesh target="rendering">
      <vertices>
        <vertex id="1" x="8.4" y="2.3" z="9.5"/>
        <!-- ... -->
      </vertices>
      <texture_coords>
        <coord id="1" u="8.4" v="2.3"/>
        <!-- ... -->
      </texture_coords>
      <normals>
        <vector id="1" x="8.4" y="2.3" z="9.5"/>
        <!-- ... -->
      </normals>
      <faces>
        <face>
          <node v="1" n="44" t="45"/>
          <!-- ... -->
        </face>
      </faces>
    </polygon_mesh>
    <barycentric>
      <coord render_id="7" tetr_id="3" v0="0.2"
       v1="0.2" v2="0.1"/>
    </barycentric>
    <attributes>
      <mass>1.0</mass>
      <volume_stiffness>0.5</volume_stiffness>
      <stretching_stiffness>0.9</stretching_stiffness>
      <friction>0.9</friction>
      <particle_radius>0.4</particle_radius>
      <solver_iterations>10</solver_iterations>
    </attributes>
  </soft_body>
  <rigid_bodies>
    <rigid_body id="1">
      <polygon_mesh target="collision_detection">
        <vertices>
          <vertex id="1" x="8.4" y="2.3" z="9.5"/>
          <!-- ... -->
        </vertices>
        <faces>
          <face>
            <node v="1"/>
            <!-- ... -->
          </face>
        </faces>
      </polygon_mesh>
      <attributes>
        <mass>1.0</mass>
        <friction>0.9</friction>
      </attributes>
    </rigid_body>
    <!-- ... -->
    <joints>
      <spherical_joint r1="1" r2="2">
        <limits>
          <limit target="swing" angle="45"/>
        </limits>
      </spherical_joint>
      <!-- ... -->
    </joints>
  </rigid_bodies>
</model>
```

Figure 8: Exemplary model file contents for a soft-body object with a rigid skeleton.



(a) (b) (c) (d)

Figure 9: Frames from the animation depicting throwing balls at a pure skeleton-free soft body. Worth noticing is the unnatural bending of limbs in images (b), (c), (d) and the overall inertia resulting from uniform mass distribution.



(a) (b) (c) (d)

Figure 10: Frames from the animation of a soft body rigged with a rigid skeleton. The shape is retained much more firmly and the character does not expose unnatural behaviour.