Target Space Modeling - The End of 3D Widgets

Kai Berger
TU Braunschweig
Germany
berger@cg.tu-bs.de

Christian Linz
TU Braunschweig
Germany
linz@cg.tu-bs.de

Christian Lipski
TU Braunschweig
Germany
lipski@cg.tu-bs.de

Tobi Vaudrey
University of Auckland
New Zealand
t.vaudrey@auckland.ac.nz

Reinhard Klette
University of Auckland
New Zealand
r.klette@auckland.ac.nz

Marcus Magnor
TU Braunschweig
Germany
magnor@cg.tu-bs.de

ABSTRACT

In today's modeling tools, the graphical user interfaces are required to be accurate and intuitive to use. Most tools therefor rely on additional 3D-widgets (e.g., arrows or circles) that enable the user to operate towards a desired modeling result. In this paper we present, for the first time, a method that makes these widgets obsolete. We propose to use simple geometric primitives such as planes or spheres as low-dimensional subspaces, so called *target spaces* for the interaction. Instead of operating towards a modeling result, the user then directly steers the result. The target spaces suffice to be indicated to the user just as additional visual information. We verify by means of a user study that with our method it is now possible to develop accurate single-view GUIs without 3D-widgets that are highly intuitive to use.

Keywords: Graphical user interface, human computer interaction, interactivity, computer geometries, 3D interaction, graphics applications.

1 INTRODUCTION

Today's modeling applications are in general used for the task to position or deform complex objects or parts thereof. Common examples are the modeling of human characters or complex moving objects (e.g., parts of a car engine). In both examples objects are defined by a transformation hierarchy. For example, the pose of a human right hand depends on the pose of the right arm, whose pose is dependent of the current pose of the torso. The term *pose* defines the current position and rotation of an object. A pose is defined by six degrees of freedom (DOF).

The object's transformation hierarchy is determined by a kinematic chain (i.e., the assembly of several kinematic pairs connecting rigid body elements). Often this combination of kinematic pairs, with 6 DOF each, leaves the user even for simple models with a parameter space of high dimensionality. This high-dimensional parameter space poses a challenge for the user to interact with the model in an intuitive way.

Some applications enable the user to specify the translation or rotation of a rigid body numerically. They may be most accurate but modeling tasks become very time-consuming and unintuitive. Others present an interface to manipulate the pose of each rigid body in a kinematic chain separately. Mostly authors show projections of an object on the three coordinate planes and a fourth viewport, where the user can freely choose the camera position. In general the free-viewpoint viewport contributes only little to the accuracy of the modeling results. Altogether, user interactions are commonly performed by using projections in some coordinate planes.

In this paper we present a new method that allows for intuitive modeling operations within a single viewport without any additional 3D widgets. By constraining the interaction space for a given operation in a 3D scene to subspaces defined by geometric primitives, the user succeeds in transforming the object or its subparts accurately and in a highly intuitive way.

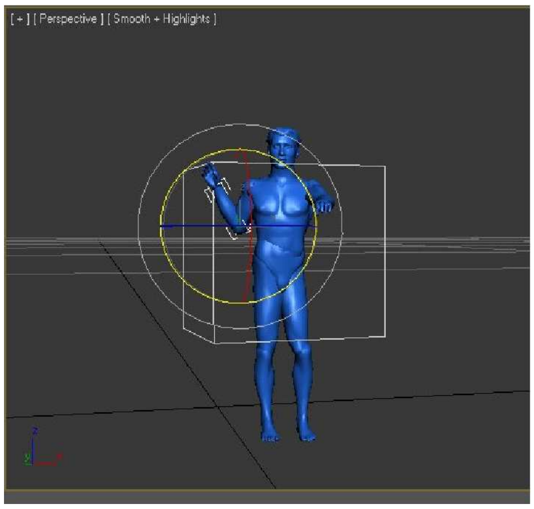
The paper is organized as follows: After giving an overview of related work in the area of interactivity methods for object modeling in Section 2, we define the core idea of subspace interactivity in Section 3 and contrast it to state-of-the-art modeling software. Afterwards in Section 4, a set of geometric primitives used for subspace interactivity is introduced. The usability of our method is evaluated in a case study with a motion capturing software in Section 5, before we conclude in Section 6.

2 RELATED WORK

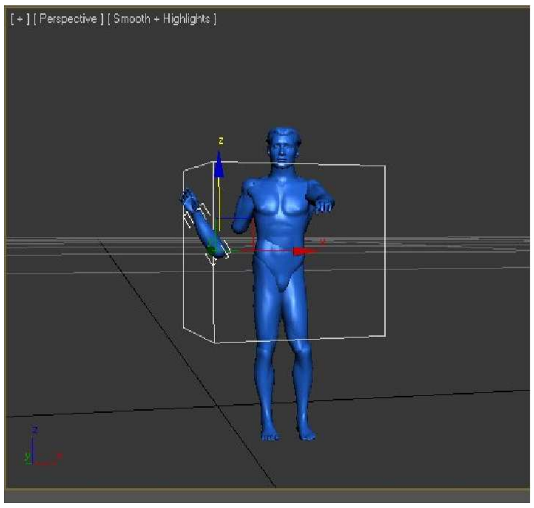
For a brief overview on research in graphical user interfaces for object manipulation, see Myers et al. [MHC$^+$96]. The interaction process from a user's point of view is described by Wright [WFH00]. Our method is related to the following work, where each technique is focusing on certain aspects of interactivity:

Widgets and Input Mapping

Wu et al. [WATB03] present a toolkit which exhaustively uses *picking*, the method to determine the corresponding 3D-object for a selected 2D-pixel. A focus on 3D-widgets for transforming complex geometrical objects is provided by Conner et al. [CSH$^+$92], who give an introduction to state definitions for typical widget operations (e.g., 3D-rotation). 3D-widgets became famous in the OpenInventor framework [SWH$^+$05]. Another paper by Dollner et al. examines 3D-widgets as deformation handlers for geometrical objects [DH98]. 3D-sliders as a type of 3D-widgets and enhancement of 2D-sliders are described by Beckenridge et al. [BHMO01] and Stotts [Sto02].



(a) Rotation



(b) Transposition

Figure 1: A typical object transformation based on 3D widgets. A rotation (a) around an certain axis is performed by dragging the corresponding circle; a transposition (b) is performed by dragging an axis-aligned arrow. Screenshots reproduced from [Aut09].

Surface modeling

Schmidt et al. [SKKS09] present a surface modeling system based on enhanced 2D views. A complex surface is depicted by equidistant lines. The lines are colored in shades of gray according to their distance from the viewpoint. Thus the user can easily select and change surface parts within a single viewport by adjusting particular lines. Another system, ILoveSketch [BBS08], allows for 3D spline sketching in a single viewport by exploiting visual cues (e.g., vanishing points).

Human motion modeling

Buttussi et al. [BCN06] developed a tool to adjust the pose of a humanoid model. They use a single viewport to select joints and drag them fronto-parallely, but they depend on a set of sliders in a second window to provide for accurate positioning. Popular modeling tools, such as Blender [Fou09] or 3D Studio Max [Aut09] employ 3D-widgets to transform nodes of the body's kinematic chain.

In 3D Studio Max, for example, an axis aligned transposition of a right hand is performed by dragging one of three displayed arrows towards the desired direction, Fig. 1 (b).

Our method, instead, introduces 1D- and 2D-subspaces for object transformation operations (e.g., rotation and transposition) and is thus independent of additional widgets, such as arrows, to be drawn in the scene. The deformation of body surfaces, however, is done by adjusting 3D sliders aligned to local coordinate axes. In the following we will mainly contrast the concepts of 3D Studio Max [Aut09], representing state-of-the-art modeling tools, to our method.

3 INPUT HANDLING WITH THE TARGET SPACE

Our main goal at this point is to provide a versatile user interface for modeling tasks which fulfills the following constraints:

1. The user should be able to perform any task in a single viewport.
2. The operation should be performed in a highly intuitive manner.
3. The result of the operation should be accurate compared to the desired task.

In state-of-the-art software, such as 3D Studio Max [Aut09], the user is provided with one or many 3D-widgets to solve the modeling task in one viewport. That means, additional objects, such as arrows or curved lines, have to be drawn around the object to be transformed. The user has to drag these objects in specific directions in order to come closer to the desired result for the object to be transformed. In general it can be stated that the screen is filled with additional objects to be handled. It is very likely that this may confuse a user.

Our approach, instead, is to consider the modeling task as a combination of lower-dimensional operations within the 3D space. An object can be transformed, for example, by a target translation $t \in T$ or rotation $r \in T$ within a lower-dimensional *target space* $T \subset \mathbb{R}^3$. This target space can be either 1D or 2D and is, thus, always embedded in the scene.

When the user triggers a requested operation for a rigid body [see Fig. 2 (a)], the target space T for this operation is optionally indicated to the user [see Fig. 2 (b)], and the backprojection Π of the mouse position p_{screen} to T [see Fig. 2 (c)] determines the desired target position p_T within the target space T.

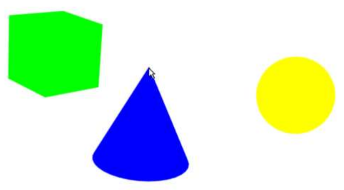
The backprojection Π can be written as

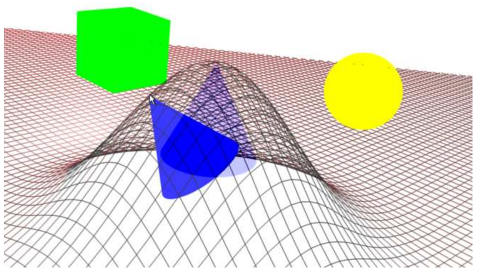
$$\Pi : p_{screen} \in \mathbb{R}^2 \mapsto p_T \in T \subset \mathbb{R}^3 \tag{1}$$

The key difference between the backprojection Π to p_T and a standard backprojection to the current scene is that only the mouse position p_{screen} is backprojected to the target space T. That is, the backprojection is partaken in a second scene which consists only of the geometric primitives which span T. The camera parameters remain constant in both scenes.

The advantage over state-of-the-art software, such as 3D Studio Max [Aut09], is now clearly that the user is independent of additional widgets to achieve desired object transformations. The user only sees the object and optionally the indicated target space. The target space may, for example, be shown as a grid to guide the user. Any mouse movement on the target space is directly interpreted as the new result state for the object. The screen becomes less confusing and more obvious, so that intuitivity increases, while accuracy stays steady.

Any geometric primitive can serve as T, as long as it is a true subset of $\mathbb{R}^3$. This is necessary for the backprojection, because otherwise the result would not be well-defined. In our further considerations we focus on thin lines, surfaces of spheres and planes for modeling tasks, but any Bézier spline or surface is also suitable. Once again we like to emphasize, that the indication of the target space to the user [see Fig. 2 (b)] is *optional*, and that the user can employ the operation on the target space also without any additional visual information. This is the *main difference* to the 3D widgets used in state-of-the-art modeling tools.


(a) User selects object to be transformed


(b) Target space T is optionally indicated to user

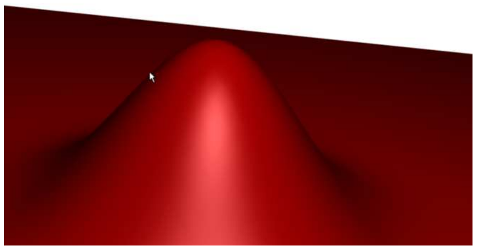
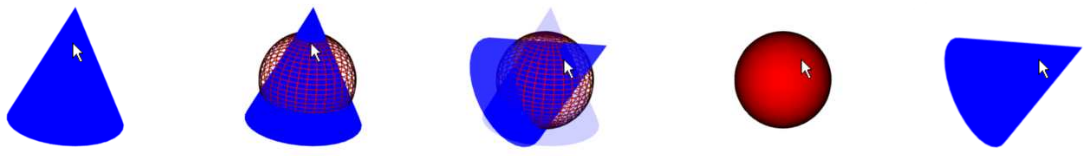

(c) Mouse position is backprojected to T

Figure 2: Illustration of a transformation of a rigid body. The blue cone (a) is considered as an operation in the target space T. Its span is optionally indicated to the user, for example by a grid; see (b). The operation is done by backprojecting the mouse position into the target space T; see the red surface in (c).



(a) User selects object to be rotated

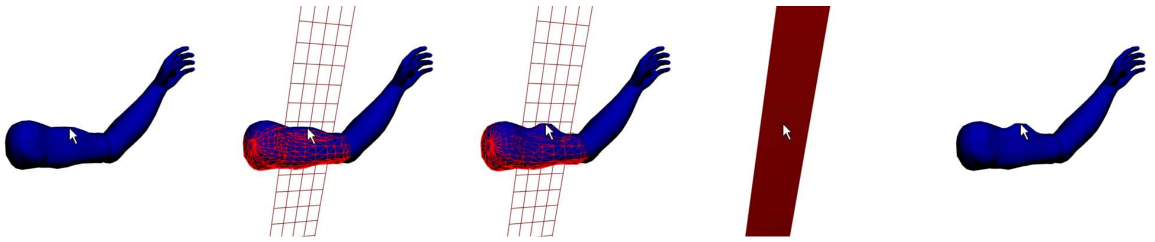
(b) User starts rotation, target space T indicated

(c) User rotates object, target space T indicated

(d) The target space T with backprojected mouse position $\Pi(p_{screen}) = p_T$

(e) User stops rotation

Figure 3: An object rotation with a sphere as geometric primitive for T. The object's center of gravity determines the sphere's center while its distance to the backprojected mouse position determines the radius. However, a larger radius is possible in order to increase usability.



(a) User selects object to be reshaped

(b) User starts deformation, target space T indicated

(c) User deforms object shape, target space T indicated

(d) The target space T (line with arbitrary thickness) with backprojected mouse position $\Pi(p_{screen}) = p_T$

(e) User stops deformation

Figure 4: An object deformation with a line of arbitrary thickness as geometric primitive for T. The line is parallel to the normal of the selected patch and intersects the object at the backprojected mouse position.

4 GEOMETRIC PRIMITIVES IN THE TARGET SPACE

In most modeling applications we face the following basic transformation operations: transposition, rotation and surface deformation. The transposition and rotation operations have 3 DOF each; the DOF of a surface deformation depends on its tessellation. However, combining only few of those operations to a complex task will lead to a parameter space of high dimensionality. Thus we exploit the geometric primitives as target spaces for the elementary transformation tasks:

- A rotation of a rigid object is done on the surface of a sphere of fixed radius r with

$$p_T(\phi,\theta) = \begin{pmatrix} r\cdot\cos(\theta)\cdot\cos(\phi) \\ r\cdot\cos(\theta)\cdot\sin(\phi) \\ r\cdot\sin(\theta) \end{pmatrix} \quad (2)$$

  where ϕ and θ determine the longitude and latitude angles. Thus, the transformation is performed in a 2D target space. The sphere center is mostly located in the object's center of gravity. The radius can be determined by the distance to the selected point on the object; see Fig. 3. However, a larger radius is considerable in order to increase the usability

- A transposition of a rigid object is done on a plane (e.g., axis-aligned on the XY-plane) in the coordinate system of the object's parent in the transformation hierarchy with

$$p_T(u,v) = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix} \quad (3)$$

  where u and v determine the point on the plane. Thus the transformation is performed in a 2D target space, as well.

- A transposition of an object with a fixed distance to a local or global center is done on the surface of a sphere according to Eq. (2). Thus the transformation is also performed in a 2D target space.

- A parametrized surface deformation is done patch by patch. For each parametrized patch, spanned by tangent vectors $\vec{r}_u$, $\vec{r}_v$, the deformation is performed on a line parallel to the patch normal

$$\vec{p_T}(\alpha) = \alpha \cdot \frac{\vec{r}_u \times \vec{r}_v}{|\vec{r}_u \times \vec{r}_v|} \tag{4}$$

  Here, α determines the point on the line. The target space in this case is only 1D, the user interacts with the surface by moving it upwards or downwards that line; see Fig. 4.

5 CASE STUDY: MOTION CAPTURING SYSTEMS

The main purpose of motion capturing applications is to fit humanoid 3D-models to the input data captured from one or many video cameras. Since in most cases the input data show only the actor in front of a well-distinguishable background, the application succeeds in recognizing and modeling the intended pose of the actor.

However, sometimes the system can produce an erroneous guess for the pose due to lighting conditions or ambiguous body constraints. Then it is very useful, if the system provides an intuitive interface for the user to correct the pose. Furthermore it is also useful for the user to define new motion sequences independently from input videos (e.g., to create new motions for an already captured character). Here, an intuitive user interface is inevitable.

In the following case study we apply the presented target space approach to implement a GUI for an existing motion capturing software. Although single joint positions can be adjusted numerically, motion capturing software has not provided an intuitive interface for a user so far. Main requirements for an interface are that the user could literally drag the body entities of the humanoid to desired poses and that he could deform the shape of the body in one single viewport. The user can rotate the camera around the captured scene like a trackball.

From these requirements we deduce the basic transformation operations of the application: While entities like hands or feet can be translated within a sphere around the shoulders or hips, other entities like the torso could be rotated in place. Furthermore the surface mesh of each entity could be deformed (i.e., inflated or deflated).

For the basic transformation operations we identify a suitable target space according to Section 4. Some entities can be transformed in multiple target spaces (e.g., the hand of the humanoid model can be dragged on a sphere around the shoulder or on a plane parallel to the shoulder's coordinate plane).

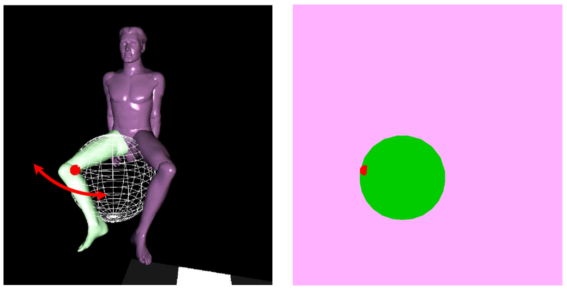

(a) Rotation of the knee, target space T is indicated to user
(b) Mouse position is backprojected to T

Figure 5: A rotation (a) of the model's knee is performed with the target space method (b): In the backbuffer the mouse position (small red sphere) is projected onto a simpler scene consisting of a sphere (green), that is tangential to the knee, the foot and the hip. A plane (pink) parallel to the viewing plane intersects the sphere at its center and thus prohibits marginal errors.

The user can pick an entity of the model by hovering over it with the mouse pointer. If multiple transformation operations are available, the user can choose the operation by pressing different keys.

A deformation of an entity's mesh can be done by moving the mouse parallel to the local surface normal. A mouse motion away from the surface indicates an inflation; a mouse motion closer to the surface indicates a deflation.

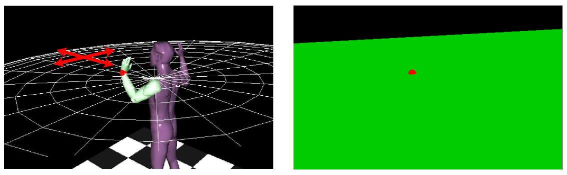
In the following, we briefly describe how the target space approach is implemented for the different operations mentioned above.

5.1 Rotation of humanoid body parts

The rotation of body parts is modeled as a transformation with a sphere surface as target space. An axis in the local coordinate system defines the rotation axis. The magnitude of a rotation angle is determined relatively from the difference of the backprojected mouse position when the mouse button is pressed and the backprojected mouse position when the mouse button is released again. Figure 5 shows the rotation operation and the according target space, the green sphere. The fronto-parallel pink plane intersects the sphere center to avoid backprojection problems, when the mouse pointer leaves the surface area of the sphere. The backprojected point p is then automatically evaluated as the intersection between a ray from the sphere center c to p and the surface of the sphere.

5.2 Translation of humanoid body parts

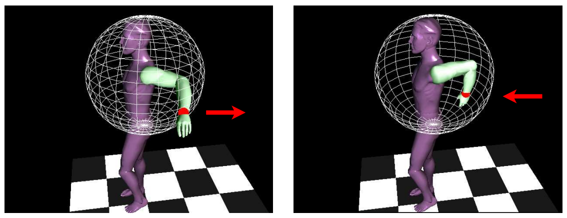
The translation of a body part is modeled both as a transformation with a plane as target space and with a



(a) Translation, target space T is indicated to user

(b) Mouse position is backprojected to T

Figure 6: A translation (a) of the model's arm is performed with the same approach (b): In the backbuffer the mouse position (small red sphere) is projected onto a simpler scene consisting of a plane (green) intersecting the arm's origin.



(a) Translation on the front hemisphere

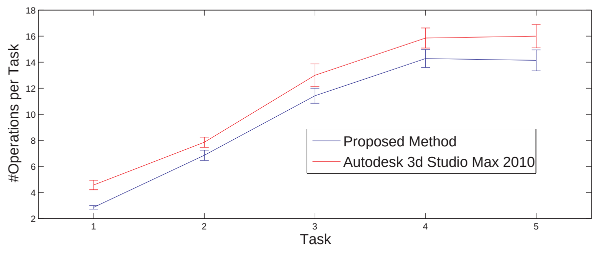
(b) Translation on the back hemisphere

Figure 7: When leaving the sphere's surface with the mouse (a), the system automatically cuts off the half-sphere in front and enables the user to translate the joint to the backside (b), which would not be visible or reachable in normal view.

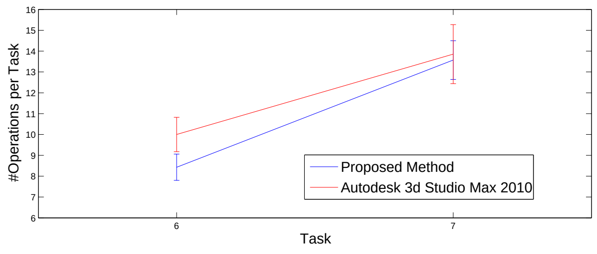
sphere surface as target space. The resulting position of the translated body part is evaluated absolutely by back-projecting the mouse position to the target space. An inverse kinematic chain [ZB94] is then computed for the parent nodes of the body part to account for body constraints (e.g. the flexion-extension range of a knee or an elbow).

Figure 6 shows the translation of the left wrist on a plane parallel to a coordinate plane of the shoulder's coordinate system and the corresponding target space, a green plane. The backprojected mouse position on the green plane determines the required position for the wrist.

Figure 7 shows the same translation of the left wrist with a sphere surface as target space. The sphere center in the target space is again intersected by a fronto-parallel plane. This time the plane is used to discriminate between the visible front side of the sphere and the invisible back side of the sphere. When the mouse pointer leaves the sphere surface and is backprojected on the plane, the system automatically cuts off the front half of the sphere. The user is then able to move along the inner surface of the sphere's back half.



(a) Results for the modeling tasks



(b) Results for the shaping tasks

Figure 8: The results of the user study. The users have been assigned five modeling (a) and two shaping tasks (b). For each task the mean number of necessary operations is plotted with variance. Note, that, on average, the proposed method needs less operations per task than the state-of-the-art tool.

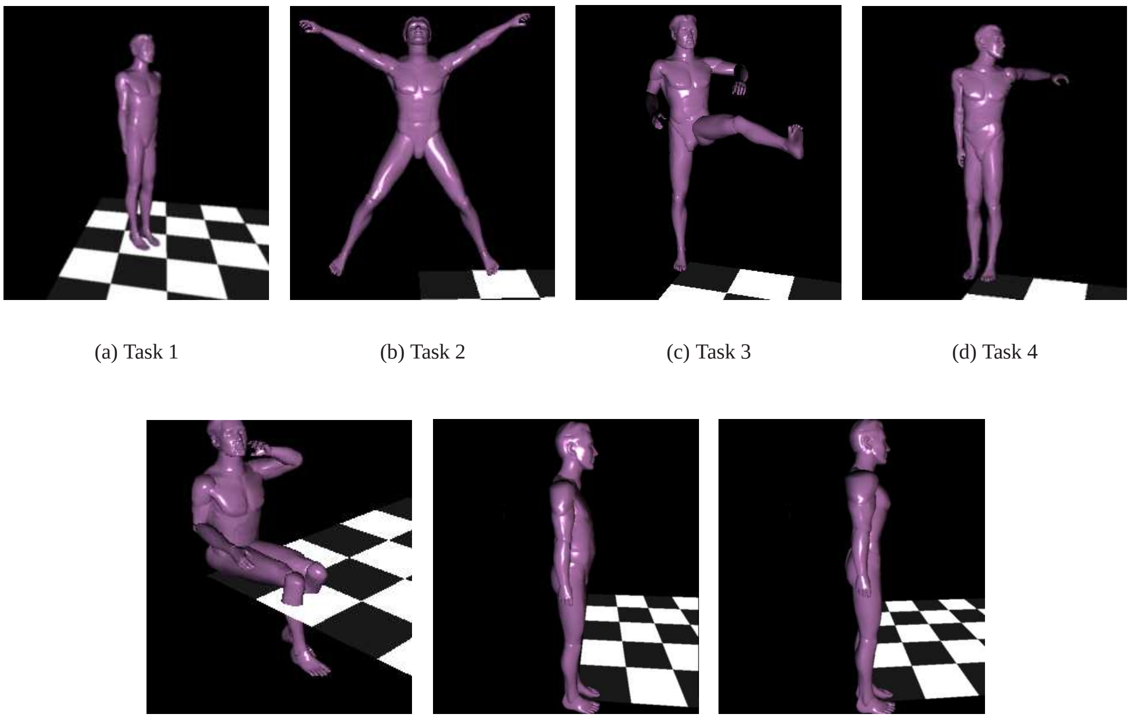
5.3 Deformation of humanoid body parts

Each body part of the model, used in the system, consists of a coherent mesh. The deformation state of the mesh is defined by two fourth order polynomials of its longitudinal axis along for each of the remaining axes. The deformation parameters for each polynomial are 1D. Thus, we employ the target space approach by moving the selected patch in the mesh along a 1D-line, which is parallel to the patch's normal. Since the parameters are 1D, the thickness of the line is arbitrary.

5.4 Results - The User Study

The target space method has been implemented in an existing motion capturing software as graphical user interface. In a comparative user study with seven participants, the applicability of the method has been evaluated. Each user has been assigned five modeling tasks and two shaping tasks. In each task the users have been given an input model and a picture of a target pose (e.g., a man sitting, kicking, or lifting arms) or a target shape (e.g., fat, or musculous).

Further information about these tasks is shown in the Appendix. The users then had to change the pose or shape of the input model so that it resembled the picture of the target pose with as few modeling operations as possible.



(a) Task 1 (b) Task 2 (c) Task 3 (d) Task 4

(e) Task 5 (f) Task 6 (g) Task 7

Figure 9: The seven modeling tasks in the conducted user study. Tasks 1-5 depict the poses the users had to model with both the proposed method and 3D Studio Max 2010 [Aut09]. Tasks 6 and 7 depict the different shapes the users had to give to the humanoid model.

In the first iteration the users modeled the pose with the proposed method. In the second iteration they used the commercial software Autodesk 3D Studio Max 2010. In both iterations, for each task the number of operations that were necessary to model the desired pose have been evaluated.

Figure 8 shows the mean number of necessary operations for each modeling (a) and shaping (b) task. The plots show that the proposed method enables the user to model desired poses and shapes with less necessary operations than with the state-of-the-art modeling software.

The users have also graded both modeling tools for intuitivity, learnability and usability. In terms of intuitivity and learnability, 71 % graded the proposed method as good as, or even better than the commercial tool. In terms of usability, 57 % graded the proposed method as good as, or even better than the commercial tool.

6 CONCLUSIONS

We presented a new approach for graphical user interfaces in modeling applications, which is based on lower-dimensional target spaces. The target spaces are embedded into the 3D scene and thus allow for single viewport interactivity. While standard modeling applications rely on fronto-parallel operations or operation vectors that the user has to drag along the axes of a local coordinate system, our approach enables the user to transform the object in a very intuitive way, still achieving the desired accuracy.

We verified the applicability of the approach by a case study, where a graphical user interface had to be implemented for an existing motion capturing system. In a comparative user study we found that with the new approach users need less necessary operations for a modeling task than with state-of-the-art modeling software.

In the future we want to apply our approach to further modeling applications beyond motion capturing. We also want to exploit the constraints of kinematic chains of arbitrary objects for the target space modeling, because this will enhance the intuitivity of modeling tasks even more.

7 APPENDIX

In Fig. 9 we have listed the seven different modeling tasks of the conducted user study. In Task 1 [see Fig. 9 (a)], the users had to reposition the model and rotate it in an angle of 45°.

In Task 2 [see Fig. 9 (b)], the users had to lift the arms and legs to emulate a jumping pose.

In Task 3 [see Fig. 9 (c)], the users had to lift one leg and angle both arms to emulate a kicking pose.

In Task 4 [see Fig. 9 (d)], the users had to position the left arm and angle it and had to rotate the head towards the far left.

In Task 5 [see Fig. 9 (e)], the users had to model a person sitting and drinking a glass of water. Not only the arms, but also the hands had to be altered.

Tasks 6 and 7 regarded shaping operations. In Task 6 [see Fig. 9 (f)], the users had to give the humanoid model a potbelly. In Task 7 [see Fig. 9 (g)], the users had to shape the model like a body builder. In both tasks, several meshes had to be selected and reshaped in all three dimensions.

REFERENCES

[Aut09] Autodesk. 3D Studio Max 2010. `http://www.autodesk.de/`, 2009.

[BBS08] S.H. Bae, R. Balakrishnan, and K. Singh. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 151–160. ACM New York, NY, USA, 2008.

[BCN06] F. Buttussi, L. Chittaro, and D. Nadalutti. H-animator: a visual tool for modeling, reuse and sharing of X3D humanoid animations. In *Proceedings of the eleventh international conference on 3D web technology*, pages 109–117. ACM New York, NY, USA, 2006.

[BHMO01] A. Breckenridge, B. Hamlet, D. Mehlhorn, and K. Oishi. A Dynamic Design Strategy for Visual and Haptic Development. *Proc. of the PHANTOM Users Group Workshop*, pages 31–37, 2001.

[CSH+92] B. D. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. *Proc. of the symposium on Interactive 3D graphics*, pages 183–188, 1992.

[DH98] J. Döllner and K. Hinrichs. Interactive, Animated 3D Widgets. *Computer Graphics International 1998*, pages 278–286, 1998.

[Fou09] Blender Foundation. Blender. `http://www.blender.org/`, 2009.

[MHC+96] B. Myers, J. Hollan, I. Cruz, et al. Strategic Directions in Human-Computer Interaction. *ACM Computing Surveys*, 28(4):794–809, 1996.

[SKKS09] R. Schmidt, A. Khan, G. Kurtenbach, and K. Singh. On expert performance in 3D curve-drawing tasks. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 133–140. ACM, 2009.

[Sto02] D. Stotts. 3D Sliders: Programming Uses for 3D Object Warping in Collaborative Virtual Environments. Technical report, University of North Carolina at Chapel Hill, 2002.

[SWH+05] D. Stalling, M. Westerhoff, H.C. Hege, et al. Amira: A highly interactive system for visual data analysis. *The Visualization Handbook*, 38:749–67, 2005.

[WATB03] S. T. Wu, M. Abrantes, D. Tost, and HC Batagelo. Picking and snapping for 3D input devices. *Brazilian Symposium on Computer Graphics and Image Processing*, pages 140–147, 2003.

[WFH00] P. C. Wright, R. E. Fields, and M. D. Harrison. Analyzing Human-Computer Interaction as Distributed Cognition: The Resources Model. *Human-Computer Interaction*, 15(1):1–41, 2000.

[ZB94] J. Zhao and N.I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)*, 13(4):313–336, 1994.