Building Up Virtual Environments Using Gestures

Alexander Marinc, Carsten Stocklöw, and Andreas Braun

Fraunhofer Institute for Computer Graphics Research IGD, Darmstadt, Germany
{alexander.marinc,carsten.stockloew,
andreas.braun}@igd.fraunhofer.de

Abstract. When realizing human-machine-interaction in smart environments it is required to create a virtual representation of the environment that encompasses not only location of the different devices supported but may also contain meta-information such as technical and logical communication layers or a description of supported functionalities, e.g. by using semantics. Creating this representation typically requires technical knowledge and manipulation of object representation files. Therefor it is a major challenge to enable this set-up for regular users, by providing an easy way to establish the virtual environment and the respective position and orientation of integrated devices. In this work we present a novel user-centered approach to create these physical parameters in the virtual representation. Based on intuitive gestural interaction we are able to define the boundaries of appliances and select their capabilities. We have evaluated this method with various users, in order to investigate if such a gestural modification of virtual representations provides an easy way for regular users to create their own smart environment.

Keywords: Smart Environments, 3D modeling, distributed computing.

1 Introduction

Considering the increasing amount of technical devices in modern households, new possibilities are desired to enable natural (in terms of intuitive usage) interaction with them. The very base of such modern forms of interaction like pointing-gestures, voice recognition and digital navigation is a virtual representation of the real environment. This model should at least contain information about the physical properties (shape, dimensions, orientation, position) of elements (devices, furniture, walls, …) and their technical properties (network-address, controllable properties / provided services, type of the device).

In the past months we have created a concept to build-up all named properties. The basic idea is here to use a virtual 3D reconstruction of the environment to support the user in configuring all parameters, as shown in **Fig.1**. This virtual reality can support End-User programming, as already has been shown by Kelleher et al. [11]. Big steps in 3D reconstruction with consumer devices for the mass-market has recently been done by Izadi et al.[4] and "KinectFusion". But also other researchers like Du et al. [5] made progress here and allow using such techniques more easily.

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2013, Part III, LNCS 8011, pp. 70–78, 2013.

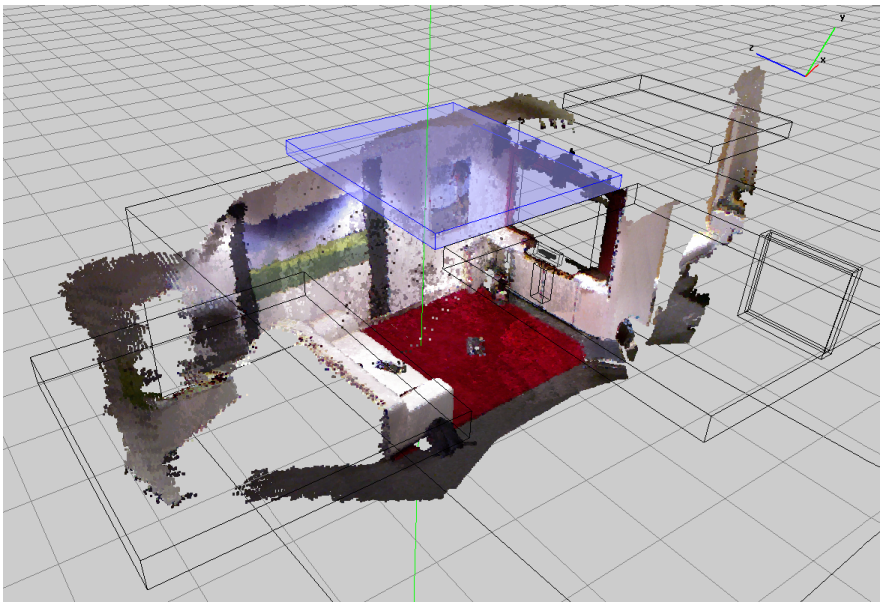

Fig. 1. Concept for modeling whole virtual environment

Here we want to set focus at creation of the physical parameters of devices in the environment. Current approaches like complex CAD applications (e.g. Autodesk AutoCAD[1], Blender[2] or CATIA[3]), more easy-t-use solutions like Trimble Sketch-Up [9] or approaches from science, like presented by Stahl et al. [1] are using a computer mouse as input-device. Consider the needed experience to modeling a 3D-Space using a 2D input-device this is not very well suited for common users.

Therefor we present here a new user-oriented approach to create the physical parameters of a virtual representation of living areas using gestures. Here we currently are focusing on the positioning of bounding boxes, but the general concept can be easily extended to place also other bounding volumes and maybe even create freeform volumes. We can benefit here from the fact that we do not need to create exact matches, but boundaries that are sufficient to get a clear expression about the environments geometry.

2 Related Works

In previous work [2] we have already shown how modeling environments based on OWL ontologies [12] can improve pointing gesture recognition in smart spaces. What we need is a platform supporting the concepts of a service-oriented architecture. A good overview about different possibilities in this domain is given by Papazoglou et al. [3] and Bodhuin et al. [10]. This allows us to associate devices with services

[1] http://usa.autodesk.com/autocad/, 24.02.2013.
[2] http://www.blender.org/, 24.02.2013.
[3] http://www.3ds.com/products/catia/, 24.02.2013.

controlling their properties and therefore to control them by simple "one-click-actions". The OSGi concept [13] is a well-suited approach that is able to realize these activities. The process of extracting semantically connected subparts from point-clouds is known as segmentation or clustering. For example Xiaojuan et al. [6] have already successful extracted architectural elements from unstructured data or Li et al. [7] extracted 3D shapes out of point-clouds. Also it may be sufficient to only extract very basic information like done by Wahl et al. [8]. They are extracting planes from point-clouds that could be used to determine whole bounding boxes.

However, such automatic approaches always have the drawback that they can't guarantee that all necessary elements are found. Additionally the gathered points-clouds given by current approaches are often very noisy and can contain big gaps. Reflecting parts (such as the TV in **Fig.1.**) may be missed completely. Very small elements like a power socket will be very hard to extract. So if manual work is required anyway, why not directly support the process of manual segmentation in living areas by using modern forms of natural interaction?

3 Manual Segmentation Using Gestures

For our concept we assume a user in front of a TV with a device attached that enables basic tracking of the user's skeleton data. Making an initial gesture the user can start the process of adding a new bounding box to the scene. The first challenge is now to determine the initial starting-point for the bounding box. As the easiest approach we could take a standard position in the (virtual) front of the user. The problem in this case is that this is not the best-suited base for the next steps, as it is hard to move the box over larger distances using gestures. Therefore we introduce the concept of defining a starting point in 3D space as shown in **Fig.2.** The idea here is to extract several pointing vectors from the tracked skeleton data and determine an approximating intersection point of all of them.

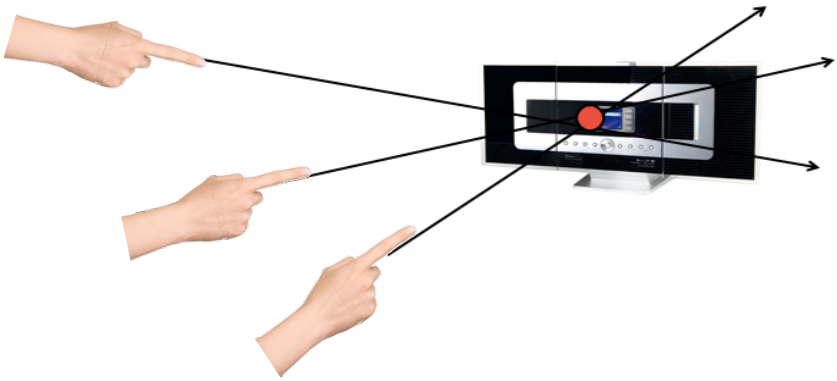

Fig. 2. Determining a spot in 3D by using gestures

Using such a point as the center of a new bounding box we can continue with the next step. Here **Fig.3.** gives a first impression of the basic idea.

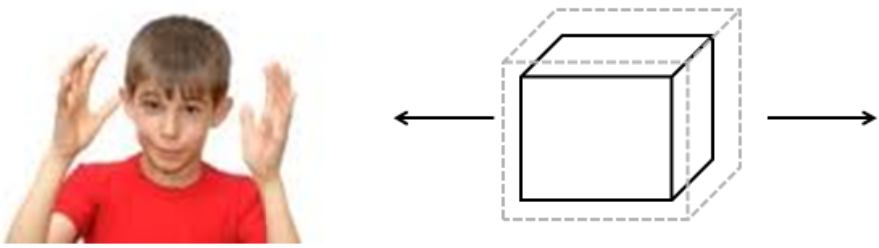

Fig. 3. Setting position, size and orientation of a box by using gestures

The distance between the hands and their orientation will be used as initial values to create more detailed configurations. By changing the distance between both hands the size of the box can be modified; by rotating them in a spherical manner it can be rotated and by moving the hands in some directions in parallel, its position can be translated. Once the user lets both hands fall down to the legs the editing is finished and the new bounding box created. By repeating all these steps a basic representation of the real environment can be created also by end-users that are not familiar with 3D interfaces and modeling.

3.1 Gathering the Initial Position

First we want to have a more detailed view into the details of the process to find an initial point *p* as a center for a new bounding-box. Therefore we extract a pointing-ray out of the tracked skeleton-data of a user. Here we are using the joints at the shoulder and the hand to determine a ray $\vec{v} = p_{shoulder} - r * (p_{hand} - p_{shoulder})$. In a first attempt we take several rays $\vec{v}_1, \ldots, \vec{v}_n$ over time and determine the point with the lowest distance to all of the lines given by $\vec{v}_i$. In practice this turned out to not be well suited, since the process takes a lot of time and does not lead to more exact results if we increase *n* above three. Finally we settled with an approach that is only using two lines to determine a point in 3D space.

Here the idea is to give the user a first pointing ray $\vec{v}$ in the direction of the object or a place where he wants to create a bounding box. After this is fixed he can point along this line to fix a final position. The resulting line we denote as $\vec{u}$. In both cases a time of two seconds (nearly) not moving the arm indicates to proceed to the next step.

Once $\vec{v}$ is fixed and the user pointing to fix $\vec{u}$ we continuously display the currently approximated position of *p*, whereas *p* is determined by the middle of the two points where the distance between the lines given by $\vec{v}$ and $\vec{u}$ is minimal, as shown in **Fig.4**.

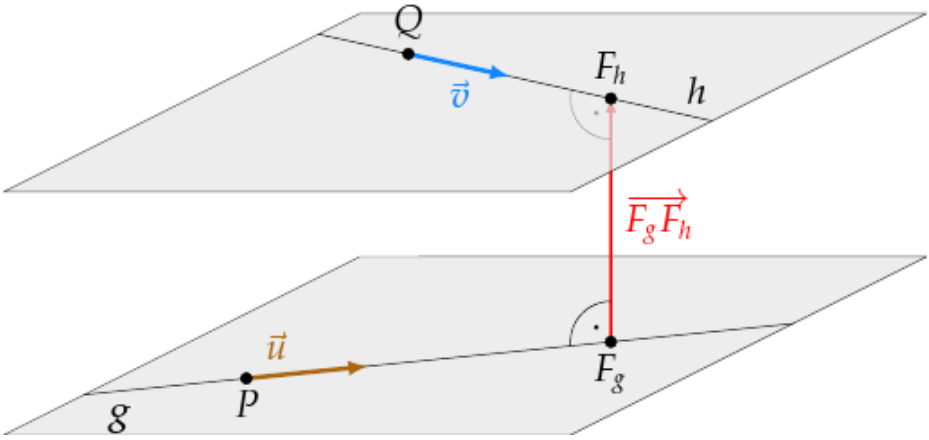

Fig. 4. Compute point of nearest contact from two vectors in $\mathbb{R}^3$ (figure by http://www.ina-de-brabandt.de[4])

In addition to this approach we also tested to get $\vec{v}$ and $\vec{u}$ in parallel by using both arms (one arm for one vector). But in particular for distances that are further away as four meters we received bad results using this approach. This can be attributed to the fact that it is hard to hold both hands fix to a single direction over a longer period of time. In the realized approach we also had troubles with points that are far away., The bigger the distance from *p* to the user's position, the smaller the angle between $\vec{v}$ and $\vec{u}$ and in consequence small changes have a larger effect. But this is simplified by the user only moving the point along a fixed line. In a next step we can consider smoothing those gestures by using also angles between $\vec{v}$ and $\vec{u}$, where *p* is normally behind the user to enable more exact operations in distances that are far away.

3.2 Setting Volume Parameters

The problem of setting (exact) position, size and orientation of a bounding box is in a first way separated from the issues handled in section 3.1. We initialize a bounding box with one size in every dimension (1 meter in real space) and the center in the origin of an affine space in $\mathbb{R}^{3x3}$. To describe all transformations we use a transformation matrix in $\mathbb{R}^{4x4}$ for each box.

Our first attempt was to gather all parameters in parallel. This means to initialize an identity box by using an initial gesture like given in **Fig.3.** Now the distance between the hands influence the size of the box in all dimensions, the angle is given by rotating the hands around an imaginary sphere and the position can be determined by "carrying" the virtual box through space.

It quickly became apparent that this approach is not suited if we assume that there is only one tracking device in front of the user (instead of many around to cover him

[4] `http://www.ina-de-brabandt.de/vektoren/a/abstand-gerade-ws-lot-lfd-punkt.html`, 24.02.1013.

in 360 degree). Here in to many situations parts of the user are covered in a way that not both arms can be tracked. In this case it is not possible to determine all three dimensions concurrently (even if this is needed for most of the objects). Therefore we split the process in several parts. We first ask to give the position by moving hands or the whole body, then to give the size in three steps (width, height, depth) by modifying the distance of the hands and finally the orientation in two steps (first rotation around the y-axis and then around x-axis) by rotating the hands relative to each other. For each step a time of two seconds with constant hand position (we used values from about 5 centimeters for transformation respective 5 degree for rotation) is used to indicate that this value should be fixed.

Even as this approach works quite well there was the need for one additional change. To simplify the interaction the changes are linearly increased. This means the further away the hands are from the initial position the more influence this change has on the transformation of the virtual object. This allows both applying more severe changes in a parameter, but also allows detailed operations.

4 Prototype Application

We combine the single approaches given in sections 3.1 and 3.2 to an overall process. Similar to what we have already done in previous work we used ontological descriptions as data-model to save the geometry to be created [2]. This can be easily created out of files specified in X3D, e.g. created by the same program that realizes user interaction. As input-device to evaluate the gestures we have used the Microsoft's Kinect[5] that is interfaced using the OpenNI[6] SDK.

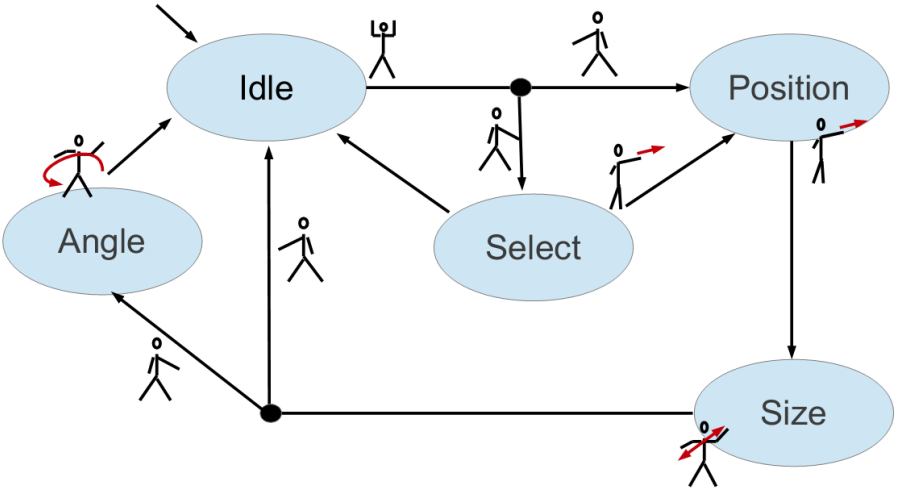

Fig. 5. The whole process of setting bounding boxes

[5] http://www.microsoft.com/en-us/kinectforwindows/, 26.02.2013.
[6] http://www.openni.org/, 26.02.2013.

The whole process of setting one or many bounding boxes into a virtual surrounding is given in **Fig.5**. It is realized as a state-machine that guides a user through the different steps of gathering the required parameters. We are starting at the "idle" state. Here the Psi-pose can be used to indicate the intention to create a new box or changing an existing one. Several users can be in front of the camera, but once a user started the creation process he need to finish or cancel it before another user can interact with the system.

If the user wants to select an existing box he just needs to hold the left arm straight in an angle of about 45° relative to the body. To create a new box he needs to use the right arm in the same way. In first case the positioning process like given in section 3.1 needs to be performed by the user and the box next to the determined point is selected. In both cases the next step is to determine the position of the box. This can either be done using the methods described in section 3.1 or by moving the box like shown in section 3.2. Here a straight arm (indicated by the three joints between shoulder, elbow and hand are in line, having no angle larger than 5° in between) is used as an indicator of what is currently desired by the user.

Now the size and angle can be also given like described in 3.2. Since manipulation and further processing are easier if a bounding box is axis-aligned the setting of the angle is only optional. Here again the arms can be used to indicate if this is desired or not. After this final step we are in the idle-state again.

5 Evaluation

The main focus of the evaluation has been to verify the benefits of the process to build up your environment using gestures vs. using the previous implemented mouse-interface. Therefore we tested both systems with eleven users, measuring the time they needed to create three boxes (Couch, TV and Light like already set in **Fig.1.**), measuring the errors found in the final result and asked qualitative feedback about their experience using the system. As basic information of every user we want to know about his previous experience with positioning elements in virtual 3D spaces (by e.g. CAD applications). Here seven users had a strong experience and four never used such a concept before.

For both groups the result has been less exact then using the mouse interface. This is mainly given by two reasons. First the measurement of the skeleton-data is quite noisy and not suited to determine parameters in centimeters. Second in particular the more experienced users often changed the view during editing to get a more exact feeling of the position of the bounding-box. When using the gestures this is not possible.

The results differed most concerning the time-constraints. Here the experienced group was faster using the mouse. However the other group was quicker using gestures. Accordingly the feedback of the first group regarding the system was very reserved, whereas the second group enjoyed the experience to influence the virtual 3D spaces easily.

In conclusion the results indicate that the system even in its prototypical form is well suited to allow common users to build up a virtual environment. Given the drawbacks above it seems not to be suited for the requirements of experts, or at least will require strong modifcations. Even if this result is not surprising, it shows that with respect to End-User-Programming the general approach seems to be a step in the right direction.

6 Conclusions and Further Work

We introduced an approach that allows common users to build up a virtual space that can be used as a base for modern forms of interaction, like pointing-gestures. Where fully automatic algorithms may easily fail in segmenting specific parts of reconstructed environments, our approach is using the abilities of the user to specify the system according to his needs. Bounding boxes for relevant devices can be placed simply by using different forms of arm gestures. As our evaluation has shown this approach has drawbacks concerning exactness and interaction time for experts, but allows untrained users a much quicker access to the system.

Regarding future work the most important issue is to extend the approach. Currently it is only used to gather the physical parameters, but with respect to the usage in real living areas all parameter(type, device specific inputs, …) need to be given only using gestures or at least a concept is needed to combine them with more traditional inputs. Also more detailed evaluations are needed to get a better impression how quick the learning process of influencing 3D space with a computer mouse is and therefore if there is a discernible advantages for beginners. Also it is desirable to implement alternatives for the used gestures and compare the results with the current approach. It may be possible that an alternative allow also experts to interact quicker with the system than they would using a mouse.

References

1. Stahl, C., et al.: Synchronized realities. Journal of Ambient Intelligence and Smart Environments 1, 13–25 (2011)
2. Marinc, A., Stocklöw, C., Tazari, S.: 3D Interaction in AAL Environments Based on Ontologies. In: Ambient Assisted Living, pp. 289–302 (2012)
3. Papazoglou, M.P., Van Den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16 16(3), 389–415 (2007)
4. Izadi, S., et al.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. ACM (2011)
5. Du, H., et al.: Interactive 3D modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th International Conference on Ubiquitous Computing. ACM (2011)
6. Xiaojuan, N., et al.: Segmentation of Architecture Shape Information from 3D Point Cloud (2009)

7. Li, X., Godil, A., Wagan, A.: 3D part identification based on local shape descriptors. In: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems. ACM (2008)
8. Wahl, R., Guthe, M., Klein, R.: Identifying planes in point-clouds for efficient hybrid rendering. In: The 13th Pacific Conference on Computer Graphics and Applications (2005)
9. Murdock, K.L.: Google SketchUp and SketchUp Pro 7 Bible, vol. 606. Wiley (2009)
10. Bodhuin, T., et al.: Hiding complexity and heterogeneity of the physical world in smart living environments. In: Proceedings of the 2006 ACM Symposium on Applied Computing. ACM (2006)
11. Kelleher, C., Pausch, R.: Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR) 37(2), 83–137 (2005)
12. McGuinness, D.L., Van Harmelen, F.: OWL web ontology language overview. W3C Recommendation, 10 (March 10, 2004)
13. Alliance, OSGi. Osgi service platform, release 3. IOS Press, Inc. (2003)