Ray, Camera, Action! A Technique for Collaborative 3D Manipulation

Wallace Lages
Center for Human Computer Interaction, Virginia Tech, U.S.A.
Universidade Federal de Minas Gerais, Brazil

ABSTRACT

In this paper we present a technique to support collaborative 3D manipulation. Our approach is based on two or more users jointly specifying the parameters of each transformation using a point, a ray, and a scalar value. We discuss how this concept can be coupled with a camera system to create a scalable technique that can accommodate both parallel and serial collaboration.

Keywords: Collaboration, 3D interaction, input device.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Direct manipulation, Haptic I/O;

1 INTRODUCTION

Manipulation is currently considered a fundamental task in 3D interaction. In spite of that, designing a good technique can be still very challenging [1]. The motif of the 2016 3DUI contest was to devise a 3D interface for manipulation able to benefit from the collaboration of multiple users. To comply with the requirements, the proposed technique must allow two or more users to apply the same operation (e.g. rotation) on a single object.

Good techniques are usually application specific, and capitalize on the features of the input devices used, required level of precision, opportunity for training and so forth. Common challenges such as poor depth judgment and the high number of degrees of freedom (DOF) make the use of stereoscopy and 3D input techniques attractive choices. Only the latter was required in the contest.

The literature on 3D manipulation techniques is extensive and a large body of work exists in the Collaborative Virtual Environments field. Aguerreche et al. [2] classify the approaches for synchronous manipulation into three categories. The first category consists in splitting the DOF between the different users. Pinho et al. [3] describe a framework for collaboration where each user manipulates only part of the DOF of a technique. For example, one user controls the object position while the other controls the orientation. This division is specified *a priori*, using the same or different techniques for each user. Our concept, on the contrary, encodes the parameters of each operation using a 3D point, a 3D vector and a scalar value. This allows us to specify how each operation can be performed collaboratively instead of just splitting spatial DOF or operations between the users.

A second category of solutions computes a final transformation as a function of the operations applied by each user. In the SkeweR technique [4], each user holds the virtual object through a point, using a 3DOF position tracker. The individual movements are then combined to produce the resulting object transformation. The Bent Pick Ray [5] is another technique that merges the inputs of individual users according to the amount of hand movement tracked by each input device. When the scene is changed by other users, the traced ray is bent to accommodate the mismatch. The main issue with this approach is the low natural efficiency of the actions. Since the actions are not necessarily disjoint, some input effort is lost when averaging or solving conflicting actions. In our approach (as in the DOF separation idea) there is no redundancy between the users and there is no need to combine actions.

The last category includes those that use a physical device to mechanically merge the input of the multiple users. Salzmann et al. [6] evaluate the use of a tangible prop to keep the hands of two users at the same distance during a translation/orientation task. The outcome of the 'haptic dialog' between the users is then retrieved directly by tracking the position and pose of the prop. Users of this technique reported a foreseeable feeling of "fighting for the prop" [7]. However, the most pronounced issues are arguably the limitation to co-located settings and the physical restriction on the number of users holding the prop. In our technique the tangible device is used by only one user.

2 TECHNIQUE DESIGN

During the analysis phase, we concluded that in order to support the collaborative execution of manipulation tasks a technique should provide means to:

- Decide which operations to perform, the sequence to be followed, and the target objects;
- Split the operation execution between the users.

Most of the aforementioned work focuses on the latter issue. The former, although not directly specified as a requirement, provided valuable insight to our design. Before committing to a particular sequence of operations, one must gather sufficient understanding of the scene. In the real world we can move around to improve depth perception or avoid occlusions. When using desktop editing software, we can have multiple simultaneous virtual cameras. In any case, what one *can see* circumscribe the possible set of actions. Therefore, a reasonable way to split operations among users is to take into account what is visible and which tasks are easy to pursue with little ambiguity. At the same time, collaboration should make action easier than a single or independent operation.

Another driving principle was to aim for better scalability by minimizing the need for communication and the need for specialized devices. We felt that a fully distributed decision process would easily lead to complex or inefficient arbitration. We also avoided relying on expensive I/O devices such as stereoscopic displays and absolute position trackers to prioritize more available 2D displays and sensors. Would it be possible to have dozens of remote users collaboratively specifying 3D operations on a scene? How could we avoid the standard 3 or 4 different cameras used by desktop 3D software?

This line of thought resulted in a technique inspired on a filmmaking metaphor which we call Ray-Camera. In this technique a director and one or more actors collaboratively carry out camera and object manipulation. The director controls the

* wlages@vt.edu

orientation of a master camera and use it to choose the next operation to be performed (Figure 1a). The operation is then assigned to an actor, along with an appropriate viewpoint. The actor uses this camera to define the remaining parameters of the operation (as he sees fit) and executes it (Figure 1b).

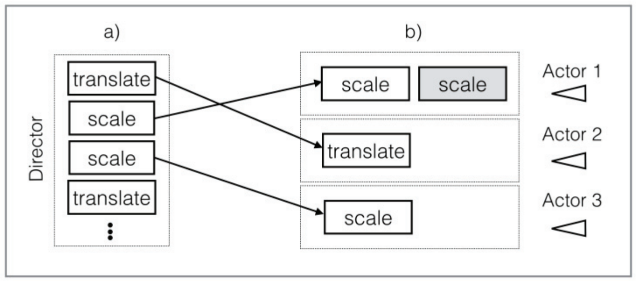

Figure 1: Figure 1. Relationship between the director and actors: a) the director sends an operation to an actor, b) the actor complete the parameters before applying the transformation.

2.1 User Coordination

The director is the only user allowed to decide which operations should be performed. He is responsible for investigating the scene and finding the best way to act using a master camera. This master camera can be rotated freely but is always targeted at the current selected object.

Once satisfied, the director can update the actor's camera based on his own. In the parallel (asynchronous) execution model, the director only sends the instantaneous camera parameters. In this case both cameras remain decoupled and the director can proceed looking for the next operation. In the synchronous mode, the cameras are coupled and the changes to director's camera are seen live by the actor. In our implementation cameras are coupled as long as a button is pressed.

The actors use their currently assigned cameras to complete the operation defined for that "shot". For translation operation, for example, the actor can choose how to move in the plane, drag or just use his pointer as a target.

We see every change of camera as the creation of a small sub-task defined by the new camera, the current operation and the selected object. This information can be easily communicated through the system GUI. To achieve collaboration, however, all users must share a previous understanding of the goals to be achieved.

2.2 Manipulation Parameters

Each manipulation operation happens in a camera plane defined by the director, using a ray controlled by an actor. The ray can be used to point, grab, move or specify parameters of operation. A point on the scene is determined by intersecting the ray with the camera plane and scalar values (or other vectors) can be derived from the target object origin and this point. Translation is achieved by moving the object in the camera plane after selecting it with a ray-casting variant or by using the intersection point as a target. Rotation can happen around the ray and scale symmetrically or in the axis defined by the object-point axis.

Whenever the camera is changed, the ray specified by the affected actor can become out of synch with its physical position (as indicated by actor's 3D interface). This can be resolved by a sharp camera change, interpolation or using the Pick Bend Ray [4] technique mentioned before. All object changes are reflected back on the director camera which can alter it to provide a better framing to the upcoming action (Figure 2).

3 Implementation

In our proof-of-concept, we used a RGBD camera as the input device for an actor. Using the camera we tracked the tip of the finger of the actor to construct an input ray into the scene. The z-axis was used as a clutch. The director controls the master camera and chooses the operations using Capsule, a 3 DOF free space input device. Capsule has two sliding covers that move symmetrically from the center of the device and 2 pressure sensors that can act as buttons. Although the form-factor is convenient for the task, the inertial tracking itself is within the reach of most modern smartphones.

4 Conclusion and future work

In this paper we present the design of the Ray-Camera technique and show how it can support scalable collaborative manipulation while allowing two or more users to perform the same operation on a single object. Among the contributions and innovations we list:

- The design of a technique to collaboratively define manipulation parameters on a single object,
- A way to seamlessly support serial and parallel collaboration,
- An actual implementation using both tangible and in-air interaction.

Several extensions are possible to this work. One example is to allow the director to select objects in the scene. Actors could also benefit from automatic camera adjustments that do not invalidate the pointing ray. On the model side, adding a pipeline for the sub-tasks would improve the parallelism when working with a small number of actors and objects. In this case, the actor interface could be modified to support gestures for navigating within the sub-task pipeline.

5 Acknowledgements

The authors would like to thank Najla Mouchrek for the help in evaluating the implementation and the anonymous reviewers for the useful comments.

References

[1] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, "3D User Interfaces: theory and practice". Addison-Wesley 2004.

[2] L. Aguerreche, T. Duval, and A. Lécuyer. "Comparison of three interactive techniques for collaborative manipulation of objects in virtual reality", *in CGI 2010 (Computer Graphics International). 2010*.

[3] M. S. Pinho, D. A. Bowman, and C. M. Freitas, "Cooperative object manipulation in immersive virtual environments: framework and techniques" *in Proceedings of the ACM symposium on Virtual reality software and technology 2002*. ACM 2002.

[4] T. Duval, A. Lecuyer, and S. Thomas, "SkeweR: a 3D Interaction Technique for 2-User Collaborative Manipulation of Objects in Virtual Environments", in *IEEE VR 2006*. IEEE 2006.

[5] K. Riege, T. Holtkämper, G. Wesche, and B. Fröhlich, "The bent pick ray: An extended pointing technique for multi-user interaction", *in IEEE Symposium on 3D User Interfaces 2006*. IEEE 2006.

[6] H. Salzmann, J. Jacobs, and B. Froehlich, "Collaborative interaction in co-located two-user scenarios", *in Proceedings of JVRC 2009 (Joint Virtual Reality Conference of EGVE - ICAT - EuroVR)*. 2009.

[7] L. Aguerreche, T. Duval, A. Lécuyer, "Evaluation of a Reconfigurable Tangible Device for Collaborative Manipulation of Objects in Virtual Reality", *in UK Eurographics Chapter. Theory and Practice of Computer Graphics* 2011.