Adaptive Hand-Tracked System for 3D Authoring

Alexis Heloir, Fabrizio Nunnari, Christophe Kolski

Adaptive Hand-Tracked System for 3D Authoring

Alexis Heloir
LAMIH-UMR CNRS 8201, Valenciennes, France
DFKI-MMCI, Saarbrücken
alexis.heloir@univ-valenciennes.fr

Fabrizio Nunnari
DFKI-MMCI, Saarbrücken, Germany
fabrizio.nunnari@dfki.de

Christophe Kolski
LAMIH-UMR CNRS 8201, Valenciennes, France
christophe.kolski@univ-valenciennes.fr

ABSTRACT
We present the interaction design and the component architecture of an adaptive authoring system based on a consumer-range 3D input device. We claim that this system can help both novice and experienced users performing authoring tasks in a 3D authoring environment. The system uses a keyboardless self-adaptive interaction controller built upon a rule-based system that learns and infers the user's behavior/condition on the fly according to her actions; rearranging rules when necessary and suggesting breaks to avoid performance drops caused by fatigue or the so-called gorilla-arm effect.

Key Words
Gestural input, 3D authoring

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Animated characters are paramount in interactive media, where movement quality ad gesture repertoire has now become an intrinsic characteristic of the essence of a character, at the same level as appearance, story background and skills. Animated characters might be a privileged support of the narrative and they also have the capability to support the expression of deaf individuals on the internet, by allowing them to express opinions anonymously in their primary language: Sign Language. All these applications – customization of character's movement, animated narrative and sign language generation – share a common requirement: they all need a convenient and straightforward 3D animation authoring system.

Recent advances in consumer-range interaction devices like the Kinect [1] or the Leap Motion [2] has opened the door to an unpreceded range of new user interfaces, interaction modalities and metaphors where gesture and bodily interaction are the cornerstones [10]. Taking inspiration from these trends, we propose a self adaptive architecture that has the potential to assist users in 3D authoring task. We currently focus on position objects in 3D space along 6 degrees of freedom. We believe that the interaction metaphors we propose in this paper can help user perform 3D editing in the following applications: character customization, animated narrative, and sign language synthesis.

Our major contribution consists of a keyboardless self-adaptive interaction controller built upon a rule-based system that learns and infers the user's behavior/condition on the fly according to her actions; rearranging rules when necessary and suggesting breaks to avoid performance drops caused by fatigue or the so-called gorilla-arm effect.

1. http://www.xbox.com/kinect (25 Feb 2014)
2. http://www.leapmotion.com (25 Feb 2014)

http://dx.doi.org/10.1145/2670444.2670456

RELATED WORK
Animation is a highly cross-disciplinary domain which spans across acting, psychology, movie-making, computer graphics, and programming. Therefore, learning the art of traditional animation requires time, effort, and dedication.However, recent consumer-range technology has proved to be capable of enabling inexperienced users to either author animation of human-like bodies or to interactively control physical and digital puppets. [9, 7]
When an animation is realized using a specialized input device or motion capture system, is often referred to as performance-driven animation or puppeteering [9]. In performance-driven animation, the animator interactively operates a dedicated interface that allows him or her to control a synthetic character with a reinterpretation of the animator's original performance. For instance, Sanna et al. [7] used the Microsoft Kinect as an input-device for driving the locomotion of an animated character.
Other work proved that specific input devices and interactive user interfaces can improve the authoring and animation process. On the one hand, some devices are better suited to record straight-ahead animation by capturing the motion or the dynamics of the user [2], on the other hand, other interaction devices are more suited to locate and rotate objects in a 3D space [5]. Only a few systems are actually suited to both methods [4] and to our knowledge, no proposed architecture accounts for the possibility to switch seamlessly between the two modes during an editing session. We believe that the system we present has the potential to fit both modes during an edit session. Even if, in the current paper, we focus on object displacement, the hand tracking device we currently use could also be used to record the velocity profile of the user's hand.

The level of maturity of 3D UI design principles is still

lacking behind those for 2D graphical user interfaces. as stated in Bowman et al in 2013 [1]: "There is currently no standard 3D UI (and it's not clear that there could be, given the diversity of input devices, displays, and interaction techniques), and few well-established guidelines for 3D UI design. While general HCI principles such as Nielsen's heuristics [6] still apply, they are not sufficient for understanding how to design a usable 3D UI." For many novice users, effective operation of 3D UIs requires going through a learning phase, even if these UIs are well designed. Even if it might seems unrealistic to completely get rid of such a learning phase, we would like to test the relevance of a self adaptive architecture for 3D UI. This architecture would be, at the beginning, very error-tolerant and easy to interact with, yet rather inefficient in terms of efficiency and accuracy. The UI would however accompany the user from apprentice to mastery, reconfiguring it's interaction mechanics and thresholds, achieving better efficiency as the user gains confidence and proficiency while interacting with the UI. Previous work on adaptive UIs indeed shown that increasing accuracy led to strongly improved satisfaction. Increasing accuracy also resulted in improved performance and higher utilization of the adaptive interface. [3]

INTERACTION DESIGN

With a mouse-based interface or a multitouch screen, users can control at most three to four degrees of freedom at the same time ([X, Y, scroll] or [X, Y, pinch, rotate]). In contrast, a 3D input device such as the Leap Motion gives simultaneous access to 6 degrees of freedom: translation and rotation along the three axis. We thus expect users to be faster when directly manipulating the 3D scene rather than when using the mouse and the keyboard. Indeed, for single target selection, Sears and Shneiderman [8] have shown that direct-touch outperforms the mouse.

We design our prototype using the Blender [3] 3D editing tool. The standard editing mechanism of Blender uses a combination of mouse and key strokes to select the mapping between the 2 DOFs of the mouse and the spatial properties of a virtual object. The editing system is albeit quite complex, and works as follows [4]. By pressing the key G (Grab) the object location in space and the mouse position are stored. From this moment the editor enters a modal state where the 2D offset of the mouse on the real desktop is mapped to a 3D translation of the virtual object. The translation occurs on a virtual plane parallel to the screen projection plane. During the interaction, the use of keys X, Y and Z can constraint the movement of the object to a different reference axes (x,y,z) or planes (xy, xz, yz). Hitting the left mouse button terminates the modal control, sticking the object to its current location. Hitting the right mouse button, or the ESC key, cancel the modal control, bringing the object back to its initial location. This last case does not appear in the figure for clarity reasons. A similar behavior is achieved by pressing R (rotate), where the default rotation is around an axis perpendicular to the projection plane, and x,y,z constraints work similarly to the grab operation.

The interaction mechanism we implemented for the Leap Motion is much simpler: After selecting a virtual object, the user presses T to begin a simultaneous modal control for both rotation and translation, i.e 6 DOFs. We call the phase within the begin of the modal control and its confirmation/cancellation an *editing action*. During early trials, we observed a number of phenomena suggesting that performance editing with the HT&M approach can further improve together with the reliability of tracking technologies. For instance, when performing docking tasks, the user has a tendency to grab the virtual object as if he/she was holding a physical object.

This observations suggest that we should design an interaction mode based solely on the hand tracking device, aiming at increasing the performance and eliminating the need to learn keyboard commands. Finally, we believe that self-adaptation can be useful among different users as well as during long working sessions of a single user. In the latter case, the self-adaptation will tune the interaction mechanic of the system according to the fatigue condition of the user. The following section describes the design we propose endow self-adaptation capabilities to our system.

SELF-ADAPTIVE ARCHITECTURE

The overall architecture of our interactive system is summarized in Figure 1. It follows a feedback-controlled loop model where the user input (hand motion) is filtered out and analyzed by a component called the Motion Analyzer. This component infers a set of mid-level motion primitives and sends them to the interaction manager: a reconfigurable rule based system in charge of triggering the right interaction mode according to its input motion primitives. The interaction manager delivers a flow of edit actions. This flow is continuously analyzed by the Status and performance assessor.

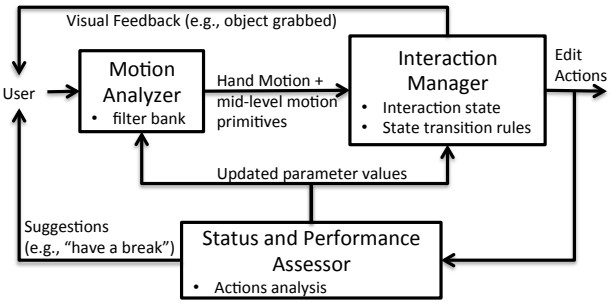

Figure 1. Our architecture is inspired by a feedback controller pattern

Our system should adapt to the user, whether the user is experienced or novice. Furthermore, if the user is novice, the system should accompany the user from beginner to master level. Doing so, it should enforce best-practices and monitor user's progress while adapting to the user's performance improvement. The system should also be capable of recognizing performance level deterioration and take measures against it. Again, by enforcing good practices or by suggesting pauses. Indeed gestural interaction, when used extensively, might induce fatigue, sometimes

3. http://www.blender.org, 6 Aug 2014
4. A comprehensive description of the editing basics can be found in the early tutorials of the Blender software.

called "gorilla arm" [5]. To avoid this, the assessor maintains a vector of descriptors that could be viewed as a simplistic user model accounting for proficiency and fatigue level. When the fatigue level increases too much, the system suggests a short break to the user. Also, the Status and Performance Assessor is continuously tuning the rules of the interaction manager to improve the user's comfort and level of performance.

Motion Analyzer

The activity of the Motion Analyzer is based on the information stream received from the hand tracking device. Each data frame contains, among others, information about the number of detected hands and a set of data-structure instances storing the position and the orientation of each palm and each detected fingertip.

The Motion Analyzer filters out and analyzes the flow of frames streamed from the Leap. It consists of a set of primitive functions performing the analysis on a time-sliding window buffer of the Leap frames received in the last 2 seconds. These functions are predicates and a selection of them can be arranged according to the instructions delivered by the Status and Performance Assessor. These predicates indicate if the user's hand appeared in the current frame (*newHand()*), of if the hand disappeared (*handLost()*. Some predicates have parameters, for instace the function telling if the hand has been stable for a certain amount of time supports two parameters: time span and distance threshold (*isHandStable[lookback_time, stability_radius]()*). Furthermore, *handFastMovement(threshold_speed)* detects if the user made a fast movement, which should be interpreted as moving away the hand from the editing task.

Interaction Manager

The Interaction Manager is an online reconfigurable rule-based/production system in charge of switching to the right interaction state according to the values computed by the mid-level motion primitives introduced in the previous section. The system handles three interaction states, as depicted in the right side of Fig. 1:

- HOVER: this state is active when a hand has been detected by the tracking system, but no editing action is actually carried on,
- GRAB: active when the user has selected an object and is moving it (visual hint),
- IDLE: active when no hand is visible by the tracking device.

When defined, new motion analysis primitives are bound to a selection of dynamic variables. They mostly represent time and distance thresholds. The value of these variables is updated on the fly according to how the user performs.

A selection of rules govern the basic state transition triggered by the presence/absence of the hand. When a new hand is detected, the system switches to state HOVER, regardless of its previous state. Similarly, when a hand tracking is lost, the system switches to the IDLE state.

5. http://en.wikipedia.org/wiki/Touchscreen #.22Gorilla_arm.22 (20 Jan 2014)

The rest of the interaction is based on the principle that when the user wants to start an editing action he needs to stabilize her hand into the sensor action space. When the hand is stable for enough time, the selected 3D object will start following the hand. This rule requires the user hand to be somehow far from the position where an object was dropped (GRAB state exited) in order to avoid undesired re-grabbing. This is done with a call to the isDistantFromLastDrop() primitive and was inserted according to preliminary observation of users' behaviour. Stabilizing the hand again terminates the editing action.

Self-adaptivity – Status and Performance Assessor

In the previous section, we saw how the state-transition was governed by a set of rules in the Interaction Manager. These rules can be modified by reparameterizing the primitive functions composing the rules and their arrangement. Such changes have an influence on the interaction dynamics. The goal of the Status and Performance Assessor is to guarantee that the current rule arrangement and parameterization maximize the user's comfort and efficiency.

We monitor aspects of the user interaction, such as her ability to keep her hand still, to control the velocity profile of the movement, as well as the reaction time to visual cues. This architecture is user-agnostic, which means that it does not build and track a model of the user, it rather tracks the short-term evolution of the user interaction and apply adaptation strategies in order to either increase the user's efficiency or to limit the decrease of performance level.

We now present how the Status and Performance Assessor monitors the user's performance level. Table 1 lists the variables we use to measure user performances. The values of the variables are calculated through the observation of the last $N_o = 25$ recognized actions. In the following, we describe each assessment variable and its influence on the Interaction Manager's ruleset parameterization.

Here, we apply the principle behind closed feedback loops systems, where the output (user performance) is constantly monitored the amplification (or attenuation) of the input signal (user actions) until, with a certain delay and oscillation, the system becomes stable. So, as the user performance is observed to increase, the thresholds will be decreased, to accommodate user need of system reactivity. On the contrary, if the user performance is decreasing, thresholds will be relaxed to make the system less prone to timeouts and movement range exceeding.

The value e is the exponential moving average of the last 25 edit actions. If Y_i is the duration of the last edit action, e_i is computed as follows:

Table 1. List of variables used to assess user performance

Name	meaning
e	action edit time
m	action linear hand movement
c	cancelled actions
l	lost hand tracking
f	fast hand movement detected

$$e_i = \alpha \,.\, Y_i + (1 - \alpha) \,.\, e_{i-1}$$

The coefficient α represents the degree of weighting decrease, a constant smoothing factor between 0 and 1. A higher α discounts older observations faster. Since α may be expressed in terms of N time periods, where $\alpha = 2/(N + 1)$. And we want N to be around 25, we use a value of 0.077 for α.

The evolution of the resulting e value gives us a hint about the user's ability to perform faster or slower edit actions. If e decreases, we assume that the user needs a more responsive system. Therefore, we decrease the value of the GRAB_START/STOP_STABILITY_TIME parameters (Rules 6 and 9). On the opposite, if e increases, we increase the value of the GRAB_START/STOP_STABILITY_TIME parameters. Practically, e is used as an adjustment for a feedback gain which multiplies GRAB_START/STOP_STABILITY_TIME by $(1 + e_i)$ at each iteration.

m is computed exactly the same way as e on the cumulative distance which is traveled by the hand during an action. Since an experienced user is capable of accomplishing an edit with only a few large actions (GRABS) of the hand in space, a decreasing value of m indicates the user tendency to perform longer movements, including large positioning (hand is fast) and fine positioning (hand is slow) of the manipulated object. This suggests that the user is gaining in efficiency and that the interactions rules may be accommodated by decreasing the GRAB_START/STOP_STABILITY_RADIUS required to enter and exit the GRAB status. In other words GRAB_START/STOP_STABILITY_RADIUS are multiplied at each iteration by the following feedback gain: $(1 + m_i)$.

The value of c depends on how many of the last N_o operations have been canceled. An operation is canceled when the user presses the ESC key in order to restore the 3D object in its initial position. For each operation with $1 < i < N_o$, we consider $c_i = 0$ if the operation has not been canceled and $c_i = 1$ id it did. We calculate the tendency by interpolating a line among the sampled results. The tendency c' is calculated as the tangent of the interpolated line. We consider a positive tendency as the fact that that too many operations started when the user didn't really mean to do. This will be used to increase both the GRAB_START/STOP_STABILITY_.... TIME and RADIUS thresholds.

The value of l is calculated, similarly to c, by counting how many times the hand tracking has been lost while performing the last N_o operations. A positive tendency tells us that the user hand is exiting too often from the editing space. This means that the user hardly feel uncomfortable in extreme hand positions. We use this as hint that we can increase the sensibility of the overall system, i.e., increase the ration between the quantity of motion performed by the dragged 3D object with respect to the same quantity of motion performed by the hand in real world. Consequently, also the value of DROP_OFF_DISTANCE (rule 6) and DIRECTION_CHANGE_SIZE (rule 8) are modulated.

The value of f is calculated similarly to c and l, by counting how many times the GRAB state has exited because a fast hand movement has been detected. If this occurs too frequently, the system modulates the value of the FAST_MOVEMENT_SPEED variable (rule 7), and may suggest the user to have a short break so that she could recover from the fatigue that might be induced by the gorilla ram effect. We are conducting further tests involving long edit session to correctly adjust this variable.

CONCLUSION AND PERSPECTIVES

To sum up, we have presented an animation system that has the potential, on the one hand, to enable novice users to author complex 3D edits and/or animations of humanoid characters and on the other hand, to increase the productivity of experienced users. This system is built upon both the Leap Motion and provides an intuitive authoring interface that has the potential to support two different edit schemes: performance capture and pose-to-pose animation. We further proposed a self-adaptive architecture inspired by a closed-loop controller that monitors the user's performance and tune on the fly the interaction controller in order to either increase the user's efficiency or to limit the decrease of performance. Since the device we used is commercially available at a consumer-range price and since we used open source software to build this system, we are publishing on-line [6] all the sources that are necessary to build and reproduce the described architecture. We will assess the benefit brought by such a system in further work.

BIBLIOGRAPHIE

1. Bowman D. A. *The Encyclopedia of Human-Computer Interaction: 3D User Interfaces, 2nd Ed.* Interaction Design Foundation, 2013.
2. Chai J. & jessica Hodgins J. K. Performance animation from low-dimensional control signals. In *Proc. of ACM Transactions on Graphics (SIGGRAPH 2005)* (2005).
3. Gajos K. Z., Everitt K., Tan D. S., Czerwinski M. & Weld D. S. Predictability and accuracy in adaptive user interfaces. In *In CHI'08*, ACM Press (2008).
4. Kipp M. & Nguyen Q. Multitouch puppetry: creating coordinated 3D motion for an articulated arm. ACM Press (2010), 147.
5. Lin J., Igarashi T., Mitani J., Liao M. & He Y. A sketching interface for sitting pose design in the virtual environment. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1979–1991.
6. Nielsen J. & Molich R. Heuristic evaluation of user interfaces. In Proc. *CHI '90*, ACM (1990), 249–256.
7. Sanna A., Lamberti F., Paravati G. & Rocha F. D. A kinect-based interface to animate virtual characters. *Journal on Multimodal User Interfaces* (2012).
8. Sears A. & Shneiderman B. High precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies 34*, 4 (1991), 593–613.
9. Sturman D. Computer puppetry. *IEEE Computer Graphics and Applications 18*, 1 (1998), 38–45.
10. Wigdor D. *Brave NUI world: designing natural user interfaces for touch and gesture*. Morgan Kaufmann, Burlington, Mass, 2011.

6. http://slsi.dfki.de