REAL-TIME BARE-HANDS-TRACKING FOR 3D GAMES

Markus Schlattmann, Johannes Broekelschen and Reinhard Klein
University of Bonn, Institute of Computer Science II – Computer Graphics, Germany
{markus, broekels, rk}@cs.uni-bonn.de

Figure 1: Bare-handed interaction for a racing game (left), a flight simulator (middle) and a first-person shooter (right).

ABSTRACT

Recently, stable markerless 6 DOF video-based hand-tracking devices became available. These devices simultaneously track the positions and orientations of both user hands in different postures with at least 40 frames per second. Such hand-tracking allows for using the human hands as natural input devices. To exploit these devices for interaction with an application, suitable interaction interfaces and techniques are required. Especially in the field of 3D gaming further developments and analyses of bare-handed interaction are needed. Therefore, we introduce a novel general interface for gaming purposes and analyzed various interaction modes/metaphors by coupling it to three popular 3D games (see Figure 1): a racing game, a flight simulator and a first-person shooter. Furthermore, we present an evaluation and discussion based on an informal user study.

KEYWORDS

Human-Computer-Interaction, Hand-Tracking, Gesture-Based Interfaces, Games, Mixed Reality.

1. INTRODUCTION

For interaction with a virtual environment, hand-tracking is one of the favorite approaches because it directly exploits the ease and perfection with which humans employ their hands in everyday life. In order to support immersive user experience, markerless real-time hand-tracking without the need of special initialization procedures gained a lot of interest in recent years. Presently, methods fulfilling these properties are capable of simultaneously tracking up to 6 continuous degrees of freedom (DOF) of both hand poses (global positions and orientations) and recognizing several stiff postures (specified configuration of finger limb positions and orientations relative to the hand pose, e.g. the fist posture) for each hand. Note that by tracking the hand poses over time also arbitrary hand gestures (predefined movements of the hand pose(s), e.g. writing a letter in the air) can be recognized and exploited.

Having a system to real-time track hand movements, suitable interfaces are needed to fully exploit its capability for efficient and intuitive interaction. Such an interface has to account for the users' capabilities and incapabilities, the applications' specifications as well as the requirements and drawbacks of the used hand-tracking method.

In the past decades numerous interfaces and interaction techniques for industrial applications were introduced such as assembly simulation or handling virtual menus. However, comparatively only little

research was performed on bare-handed interaction with games. In the last years research in this special field gained more interest, also because of the great success of novel interaction devices as for example the Nintendo Wii-Remote controller. But the Wii-Remote includes the need of holding a physical device and is not able to fully recover the global pose. In contrary an adequate markerless vision-based hand-tracking system does not suffer from these drawbacks.

Therefore, we introduce a novel 3D interface which can be adopted for any game having joystick support. We furthermore provide detailed information about suitable interaction with different kinds of 3D games. In particular, we implemented and connected a markerless real-time two-hand-tracking system to a racing game, a flight simulator and a first person shooter and analyzed the applicability of various interaction modes/metaphors by performing an informal user study. Last but not least, we discuss the results and experiences, present several reasonable solutions and extract some general principles for interaction design.

2. RELATED WORK

The different types of interfaces to augment user interaction in Mixed Reality interfaces can be classified into three different categories (according to Wanderley et al., 2006):

1. Tangible interfaces: physical objects (e.g. a cube, a tool, etc.) are exploited to control manipulations of a virtual counterpart. For example, in the game mulTetris (Audet et al., 2006) the bricks of the traditional tetris game could be manipulated by real brick-like devices.
2. Traditional VR interfaces: physical controller devices (e.g. gloves, joysticks, wands, infrared sensors, etc.) are used for user interaction. For example in the game ARQuake (Thomas et al., 2002) augmented reality technology (e.g. head mounted displays, GPS-tracked laptops) is massively used for interaction. Also the Wii-Remote belongs to this category.
3. Bare-handed interfaces: the user body (or parts of it) is exploited as an interaction device. No markers or physical devices are needed. This way, more natural interaction can be designed (according to Winkler et al., 2007).

This work presents a novel bare-handed interaction interface for gaming purposes. Therefore, in the following we will only focus on literature in this particular field. We refer readers being interested in other approaches to (von Hardenberg and Bérard, 2001) and (Bowman et al., 2004). Von Hardenberg and Bérard (2001) provided a brief overview of earlier bare-handed human-computer interaction approaches and applications. (Bowman et al., 2004) provides an extensive overview over 3D interaction interfaces and techniques.

Freeman et al. (1996) developed hardware and algorithms for a bare-hand-tracking system to fulfill the high speed and low-cost requirement for computer games. Four hand parameters could be recognized this way: the two coordinates of the position in the camera image, the orientation parallel to the image plane and the width of the hand. These parameters were exploited to control a racing car in a simple way. Unfortunately, the interface design was strongly restricted by the little amount of available DOFs.

Segen and Kumar (1998) introduced the 3D hand interface system GestureVR. Two cameras were used to capture the players' hand motion, and three postures were defined by tracking the thumb and the index finger in real time (60Hz). For each finger, five spatial parameters, consisting of finger tip positions (X, Y, Z), azimuth angle and elevation angle of the finger axis, were calculated. Two applications were developed based on this system: a fly-through application and a first person shooter (FPS). In the fly-through application the user can control the flight of a virtual camera based on the index fingertip position and the hand's pitch, yaw and roll angles. For the FPS only one camera is used for tracking the hand. Because of that, less DOFs were available for interaction. Only forward/reverse speed, lateral motion and the angular velocity of turning the player left/right were controlled by the hand pose. Opening a door and firing a weapon were mapped to different hand postures. However, in current FPS additional controls have to be handled (e.g. look up/down).

For a FPS game Kang et al. (2004) were the first to exploit bare-handed upper-body gestures for game control. 10 predefined gestures were mapped to specific controls in the game Quake II. Park et al. (2006) improved this interface mostly with respect to the requirements on the environment the person is tracked in. Though such an interface enables users to play with natural gestures, it is quite exhausting to use full body motion especially in such a game scenario, because a FPS requires fast and frequent actions.

Lu et al. (2005) presented a vision-based fist tracking method to control their own racing application. Both hand positions are tracked and a steering wheel gesture (angle of the line connecting both hand positions) is used for steering left and right. However, this approach was very simple and the hand orientation was not taken into account.

Recently, Song et al. (2008) introduced a vision-based 3D finger interaction approach to control two self-implemented games: a finger fishing game, where small virtual objects have to be angled with the index finger, and a virtual version of Jenga, where parts of a tower of wooden blocks have to be extracted successively while trying not to topple the tower. These games are nice examples of how traditional games with physical interaction can be implemented in Virtual Reality using vision and physics simulation. But while Song et al. (2008) designed novel games for bare-handed interaction, the focus of our work is designing suitable bare-handed interaction interfaces for existing standard games.

Many more articles exist in the field of bare-handed interaction, however, for lack of space we had to restrict the literature overview to this selection of most related ones.

3. THE HAND-TRACKING DEVICE

To markerless track the 6 continuous DOFs of the user's both global hand poses in several different stiff postures we implemented the method of Schlattmann and Klein (2007). We decided to use this approach due to the real-time capability and automatic initialization. In contrast to (Schlattmann and Klein, 2007), we connected all three cameras to one and the same computer where the hand-tracking is computed. The intersection of the viewing volumes of the three cameras defines the working volume of the hand-tracking. It describes the physical space the user's hand poses and postures are determined in (approximately $80cm \times 50cm \times 50cm$).

Figure 2: Our hand-tracking prototype.

In this method for each hand the fingertip positions of two stretched fingers are identified and the hand center (the center is defined to be the hand position) is computed as a third point to sufficiently determine the hand pose in each frame. Note that due to the dependency on the fingertip positions movements of the stretched fingers slightly influence the derived hand pose. The different hand postures are distinguished by evaluating some heuristics based on the local surroundings of the fingertip points.

For our current setting, industrial cameras (DMK 21AU04 from The Imaging Source) with a resolution of 640x480 pixels at 60 frames per second (fps) (see Figure 2) are adopted. The hand-tracking is computed on a standard PC (E6600, Geforce GTX 280) and induces a latency of about 12 milliseconds. We also built a cheaper prototype running together with low cost webcams, however, for gaming purposes the typical frame rates of up to 30 fps are not sufficient. But we expect that in the near future the prices of webcams with 50 or 60 fps will heavily decrease.

4. THE JOYSTICK DEVICE

In order to connect a game to our hand-tracking device the pose and posture of the hands have to be mapped to specific controls in the game. To this end, a good solution is using a virtual joystick driver due to the joystick support in most games. The virtual joystick driver receives information from the hand-tracking device about the hands' poses and postures. It is acting like a physical controller allowing for mapping real hand movements to specific actions in the game. To implement such a driver we used the freeware program Parallel Port Joystick (Version 0.83) of Deon van der Westhuysen. This framework provides a virtual

joystick driver for Windows 32 with up to 8 analog controls (finite continuous values) and 32 digital buttons (boolean values) and allows for coupling them to arbitrary hand movements. In the following two subsections, we describe which analog and digital controls are available in our implementation. However, which hand movements are mapped to which controls depends on the current game and interaction mode.

4.1 Analog Controls

The analog controls can be coupled to the coordinates of the hand positions or to the pitch, yaw and roll angles of the hand orientations. Moreover, combinations of both hands' poses can be used. For two-handed analog input we implemented the steering wheel gesture (Cutler et al., 1999), where the position is determined by the average hand positions and the orientation is determined by two different perpendicular orientations: the orientation of the line connecting the two hand positions and the average hand orientations with respect to this line.

The limits and deadzones of the respective joystick analog controls can individually be set. The lower and upper limits describe an interval in the hand coordinate system and determine the mapping of the hand coordinates or angles to the deflections of the respective virtual joystick's analog control. Values less or equal than the lower limit are mapped to the maximal negative deflection -1 and values greater or equal than the upper limit to the maximal positive deflection 1. The center between the lower and upper limits is mapped to deflection zero. Dependent on the size of the deadzone also values close to the center are set to zero deflection. Additionally, it is possible to apply an arbitrary non-linear mapping to an analog control, which can simplify the handling in a game (e.g. steering in a racing game can be improved this way).

Mapping the 3D position to 3 independent analog values can be solved straightforwardly by using the X, Y and Z coordinates of the hand positions. To split the 3D orientation into 3 independent angular values, we first exploit the horizontal angles to determine pitch and yaw. Thereby, pitch is defined to be the elevation angle and yaw the azimuth angle. Both angles are zero if the hand's index finger is pointing toward the screen. Note that this way the pitch angle α_p is restricted to the interval $(-\frac{\pi}{2};\frac{\pi}{2})$ and the yaw angle α_y to $(-\pi;\pi)$. Second, the roll angle α_r is determined by the rotation around the pointing direction. We define it to be zero if the thumb's pointing direction is parallel to the plane spanned by the up-vector and the index finger's pointing direction. Note that α_r is also restricted to the interval $(-\pi;\pi)$.

This way, the angles of the hand orientations are unambiguously defined. In case the index finger's pointing direction nearly points up, the computation of the yaw and roll angles gets unstable, however, in practice this rarely happens.

The world coordinate frame is oriented so that the positive X-axis points from left to right, the positive Y-axis from bottom to top and the Z-axis from the user to the screen. The pointing directions of index finger and thumb can be approximated by applying the hand(s) rotation to the vectors $(0,0,1)$ and $(0,1,0)$, respectively, because the identity rotation describes the hand pointing toward the front with the thumb pointing upward. Note that in the two-handed case (using the steering wheel gesture) the orientation is described by the identity when the Y and Z-coordinates of both hand positions are equal and the average orientation around the connecting line is the identity. In this case, the angles can be computed the same way.

4.2 Digital Controls

A virtual digital button can either be pressed or not. We can map arbitrary functionalities to button events. Besides mapping different postures to different buttons we exploit various hand gestures determined for example by thresholding on the hand position or velocity with respect to one coordinate or angle. Moreover, we implemented the roll click (Schlattmann et al., 2009), where a button can be pressed by a small hand movement around the hand's roll axis to the left or to the right, and the air tap gesture (Vogel and Balakrishnan, 2005), where a button can be pressed by a small movement of the index finger.

We distinguish two kinds of buttons in games: first, buttons that are used for infinitesimal events like "jump" or "reposition" and second, buttons that also need to be pressed a longer while like "crouch" or "brake". The first class of buttons can be assigned to all kinds of hand movements while the second class needs the ability to regulate the duration the button is pressed.

5. THE GAMES

Our markerless hand-tracking based virtual joystick was tested as an input device for three different representative 3D games: a racing game, a flight simulator and a first-person shooter. For each game we investigated different modes of interaction and present one or more suitable solutions.

To analyze the performance and usability of different interaction modes we conducted an informal user study. Five university students (all male) took part in several experiments concerning the different games and different joystick configurations. The analyses of the different games and interaction modes are based on the comments of the subjects and our observations on how they were capable to handle it. Please note that objective testing in gaming is extremely difficult and costly, because it would need a great many of subjects due to the typically very high rate of contingencies in games. Moreover, the subjects' familiarization with a game or interaction mode aggravates this problem.

To avoid other controllers than our device being involved during interaction also menu navigation has to be addressed. This is solved by using the hand-tracking device to simulate the standard 2D mouse. Therefore, the pointing direction is exploited to move the cursor and clicking can be performed via the roll click (see Sec. 4.2). In games not supporting mouse movements, small rotational hand movements around the pitch and yaw axis are exploited to move the selection up/down and left/right. Clicking is also done via the roll click.

5.1 Racing Game

The first game we investigated is a racing game named Re-Volt released by Acclaim Entertainment in 1999. In this game the user steers a toy car through everyday environments (e.g. a street or a garden) and can collect items (randomly assigned) as for example rockets or turbo boosts. The items can be activated by pressing a button. This game has two continuous DOFs: steering left/right and accelerate/reverse. Moreover, several digital controls are available: fire, flip car, reposition, pause and horn.

We experimented with different analog and digital mappings of hand movements to game controls. For one-handed left and right steering it turned out that only the X-axis coordinate of the hand position or the rotational yaw and roll angles can be adopted in an intuitive and easy-to-use interface, because otherwise the direction of virtual movement does not correspond to the direction of real movement. For two-handed interaction the roll was by far the best choice for steering left/right. To further improve the steering movements (both for one and two-handed), a non-linear mapping is applied to the according analog control. To this end, the absolute values of the negative and positive deflections are raised to the power of 0.7. Note that the deflections have values between -1 and 1. This way, turning the car is more sensitive to small hand movements (simplifies fast reactions), but the impact of spacious hand movements is decreased (minimizes unintentional skidding of the car). Experiments with the subjects showed this to be controllable more easily.

For acceleration and reverse we identified two different possibilities which could easily be handled by the subjects: either using the Y-axis coordinate (the hand(s) has to be moved near to the user to reverse and vice versa to accelerate) or the pitch angle(s) with a range of 0 to $\frac{\pi}{2}$ (see Sec. 4.1). While using the Y-axis coordinate was a bit more intuitive than the pitch angle it unfortunately induced more fatigue due to the need of leaving the arm at a specified position to accelerate. In contrast, using the pitch angle with this limits was very comfortable because in a racing game the user normally accelerates most of the time, so the user's hand can mostly remain in the pitch center orientation.

For the digital controls we experimented with various configurations. However, for lack of space we will only discuss our final solutions here. In our final one-handed solution, which served best for most subjects, we use pitch for accelerate/reverse, yaw for left/right, a left roll click for fire, a right roll click for flip car, the 'palm'-posture (all fingers are stretched) for reposition, the 'pointingB'-posture (only index finger is stretched) for pause and a small and fast movement down for horn. In the two-handed case, we use the combined roll angle for left/right steer. An exemplary image sequence of the one-handed solution is depicted in Figure 3.

Figure 3: An exemplary image sequence of playing Re-Volt using our interface. Performed actions from left to right: turn right, turn left, drive straight, flip car, accelerate.

Both solutions were found to be intuitive. All subjects were able to use it without a long familiarization. They liked to use this kind of interaction for a racing game. However, using both hands turned out to be more exhausting due to the need of holding both hands in midair. On the contrary, the one-handed technique induced only little fatigue because the hand and arm could rest on the table.

5.2 Flight Simulator

In order to investigate the capability of bare-handed input for a flight simulator, we connected our joystick device to the game Yager of Yager Entertainment released in 2003. In this 3D game a futuristic fighter jet has to be controlled, so military actions are also part of the game. Yager offers two modes of operation called “jet-mode” and “hover-mode”. We focus on the first one, since it provides a control similar to most other flight simulators where typically throttle, pitch, yaw and roll have to be assigned to the input device(s). Obviously, the most intuitive mapping is assigning these controls to the corresponding hand movements. Therefore, the hand pitch, yaw and roll are used for the plane’s pitch, yaw and roll controls and the Z-coordinate of the hand(s) position is used for throttle. This approach is similar to the typically adopted control in fly-through applications (e.g. Segen and Kumar, 1998). Resulting from several experiments with the subjects this solution was found to be most efficient, too. Similar to the racing game we noticed the two-handed interaction to be significantly more exhausting. But on the contrary the subjects could handle the plane more accurately by using the steering wheel gesture. Because of that, switching between one and two-handed interaction was allowed at any time during play.

The most relevant digital controls in such a flight game are: primary fire (e.g. fire the gun), secondary fire (e.g. launch a rocket) and extend/retract undercarriage. After experimenting with several different combinations we came to the following solution: primary fire can be performed by moving the hand (only single-handed) to the right side of the working volume and secondary fire can be performed by a short and fast hand movement toward the left followed by a similar movement back to the right. This way, the duration of primary fire can easily be regulated and secondary fire can simultaneously be performed. Note that involuntary use of secondary fire induced by stopping primary fire is avoided by also conditioning secondary fire on the hand movement back to the right. An image sequence of a captured video of one of the subjects using our interface is depicted in Figure 4.

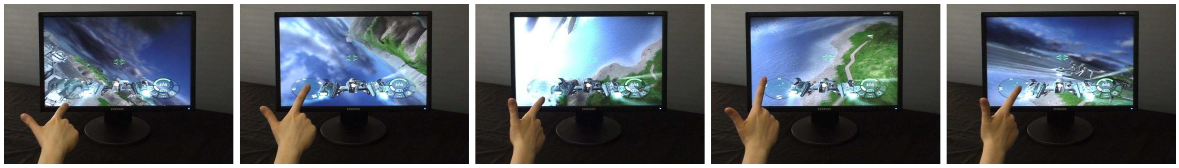

Figure 4: An exemplary image sequence of playing Yager using our interface.

If both hands are tracked, the distance between the hands is used for firing instead of the X-coordinate of one hand; a large hand distance for primary and a short fast decrease and increase of the distance for secondary fire.

The undercarriage is extended/retracted when the user forms the ‘palm’-posture with one or both hands. This accounts for the higher demands on the fine motor skills for performing posture changes because this functionality is needed very infrequently.

This kind of interface turned out to be intuitive for the analog controls and easy-to-learn as well as easy-to-control for the digital mappings. The feedback from the subjects was very positive.

5.3 First-Person Shooter

The last game we have chosen is Quake II from ID Software released in 1997, a well known representative of the first person shooter genre. Looking from the perspective of the player character, the user has to shoot enemies and make his way trough a futuristic alien environment. The game features items like weapons and power-ups which can be collected by moving the virtual player nearby. While power-ups are activated immediately, the player can switch between all the weapons already collected.

The player has 4 continuous DOFs: moving forward/backward, shift left/right, turn left/right, look up/down. The most relevant digital controls are: fire, next/previous weapon, jump and crouch.

For this game we first concentrated on a single-hand mapping that should gather all the controls as intuitive and efficient as possible. Therefore the hand position is used for "walking" and "running". We map the controls such that the hand movement direction corresponds to the direction of the resulting in-game movement. The Z-coordinate is used for moving forward/backward and the X-coordinate for shifting left/right. The in-game speed of movement along an axis is determined by the hand's distance to the origin. The sizes of the according deadzones strongly influenced the subjects' handling of the virtual player. If set to a small range (e.g. 2 cm in our final setting) an imminent change of movement is easily possible with just a slight change in hand position. This allows fast control for an experienced player. However an unfamiliar user will experience problems finding a resting position. An enlarged deadzone would solve this problem but demands more spacious hand movements. Therefore, we use an upsized deadzone for unfamiliar users and allow for dynamically decreasing deadzones for experienced users.

The hand orientation is used for „looking around". Note that this is also exploited for aiming at enemies, since weapons are always fired in direction of the screen center. The hands pitch can be used directly as the absolute pitch for in-game view (scaled by 0.5 in our setting) for comfortable and precise control. Yaw however needs to be at least partly relative, since the virtual player has to be turned around horizontally in the full 360 degrees range, which is not applicable for the user hand. We tried several combinations of absolute and relative control, but unfortunately this always resulted in unintuitive handling. Therefore, we exploited a relative control with the non-linear mapping $f(x) = |x|x + \frac{x}{2}$, where x is the deflection of the yaw angle's analog control and the resulting value determines the horizontal rotational velocity in the game. This mapping allows for precise as well as fast turning and showed the best results in our experiments.

For the digital controls we used a left roll click for fire (see Sec. 4.2), a right roll click for next weapon, a lower bound on the Y-axis for crouch and a fast and small up-movement for jump. Since in Quake II all usable objects are activated instantly by touching (e.g. doors and buttons) we had not to address this functionality. For games where an activate functionality is needed we suggest to make use of the "pointingB"-posture. An image sequence of a captured video of one of the subjects using our interface is depicted in Figure 5.

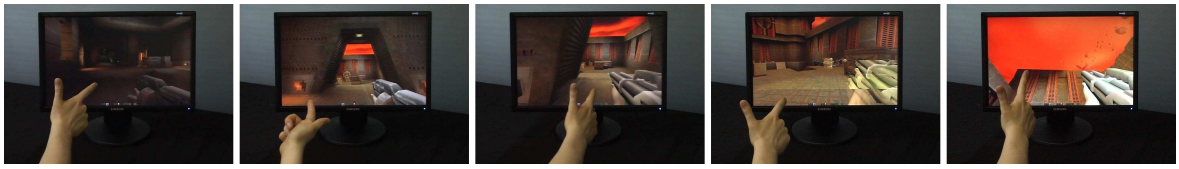

Figure 5: An exemplary image sequence of playing Quake II using our interface. Performed actions from left to right: turn right, change weapon, move straight, shoot, look up.

Since our first approach bundles all controls in one hand it turned out to be hard to handle for some subjects. Therefore, we also tried different two-handed configurations in order to distribute the functions on both hands. Unfortunately, when "moving" and "looking" were mapped on different hands the required asymmetric coordination made control less intuitive for the subjects. Even after a decent time of testing the subjects did not show significant improvements. A more suitable setting is mapping the digital controls to one and the analog controls to the other hand. Like the one-hand setting, this was perceived as rather intuitive. Since it also bears the disadvantage of being more exhausting we integrated it as an optional approach to single-handed control.

The one-hand approach was found to be rather understandable and intuitive but also a bit hard to begin with. Problems occurred when subjects got into stressful in-game situations such as being simultaneously attacked by multiple enemies. The necessary movements like shifting aside and aiming at the enemies often

resulted in a loss of orientation. Fortunately, the subjects adapted quickly and were in particular able to notably improve the accuracy of aiming after a short while. Nonetheless, we have to conclude that in terms of mere accuracy, our approach can not compete with a classic two-handed mouse/keyboard control. But considering the positive feedback of the subjects we think that bare-handed interaction might be similarly or even more enjoyable for this kind of game. Last but not least the one-hand approach is a possibility for disabled people not having the option of using both hands.

6. DISCUSSION

Whenever hand motion is used for interaction, an important part is reducing fatigue. Moreover, a hand movement adopted for a specified action in-game should be intuitive or at least easy-to-learn. Based on our experimental tests several design principles for bare-handed interaction became apparent in this context:

- Minimize need of stretched arms in midair. Physical support (e.g. desk) or closer working space can help.
- Prefer rotational degrees of freedom over translational for movements needed very frequently, because for translations always the whole forearm has to be moved.
- Ensure that the mostly applied hand orientation is comfortable for the user. This is especially important for pitch and yaw, because bending the wrist joint is exhausting.
- Avoid high demands on the fine motor skills. For this reason, changing hand postures should only be used for infrequent actions.
- Avoid fast and spacious hand translations except exhaustion is part of the game (e.g. sport games).
- Give the opportunity to use only one hand. Using two hands is more exhausting than using only one hand.
- The directions of movement should correspond to the directions in the game (e.g. steering left should be performed by a left hand rotation or translation).
- Avoid exploiting too many DOFs if only little training time is available, because more DOFs need more familiarization time.
- Implement redundancies if possible. This better accounts for different user preferences.

These principles might support other developers to design suitable interaction metaphors for gaming.

7. CONCLUSION

We introduced a novel bare-handed interaction interface for 3D gaming purposes. Furthermore, an evaluation of this interface for three typical games belonging to different genres was presented and discussed. This way some general design principles were identified supporting other developers to design reasonable bare-handed interaction.

In the future we plan to investigate the application of our interface to further kinds of games. We think that in this context also new interaction techniques might be needed. This will probably be the focus of our future work. Moreover, we are working on improvements of the hand-tracking system for example to further reduce the latency and the requirements on the environment (e.g. controlled background).

ACKNOWLEDGEMENT

We thank Tan Zheng for his help to find suitable games and his suggestions concerning the related work.

REFERENCES

Audet, S. et al., 2006. MulTetris: A test of graspable user interfaces in collaborative games. *Course project*. Canada.

Bowman, D. A. et al., 2004. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Cutler, L. D. et al., 1997. Two-handed direct manipulation on the responsive workbench. *In: Symposium on Interactive 3D graphics (SI3D '97).* New York, USA, pp. 107–114.

Freeman, W.T. et al, 1996. Computer Vision for Computer Games. *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition.* Killington, Vermont, USA, pp. 100-105.

Kang, H. et al, 2004. Recognition-based Gesture Spotting in Video Games. *In Pattern Recognition Letters*, Vol. 25, No.15, pp.1701–1714.

Lu, P. et al, 2005. A Vision Based Game Control Method. *In HCI/ICCV 2005*, Vol. 3766, pp. 70–78.

Park, H. S. et al., 2006. Vision-Based Game Interface Using Human Gesture. *Lecture Notes in Computer Science*. Germany, pp. 662-671.

Schlattmann, M. and Klein, R., 2007. Simultaneous 4 gestures 6 DOF real-time two-hand tracking without any markers. *In: ACM Symposium on Virtual Reality Software and Technology (VRST '07).* Newport Beach, California, pp. 39-42.

Schlattmann, M. et al., 2009. 3D Interaction Techniques for 6 DOF Markerless Hand-Tracking. *In: The International Conference on Computer Graphics, Visualization and Computer Vision (WSCG '09).* Plzen, Czech Republic.

Segen, J. and Kumar, S., 1998. Gesture VR: Vision-based 3D Hand Interface for Spatial Interaction. *Proceedings of the sixth ACM International Conference on Multimedia*. Bristol, United Kingdom, pp. 455-464.

Song, P. et al., 2008. Vision-based 3D finger interactions for mixed reality games with physics simulation. *Proceedings of the Virtual Reality Continuum and its Applications.* New York, USA.

Thomas, B. et al., 2002. First Person Indoor/Outdoor Augmented Reality Application: ARQuake. *Personal Ubiquitous Computing*. Vol. 6, No. 1, pp. 75-86.

von Hardenberg, C. and Bérard, F., 2001. Bare-hand Human-computer Interaction. *Proceedings of the 2001 Workshop on Perceptive User Interfaces*. Orlando, Florida, pp. 113-120.

Wanderley, I. et al, 2006. A Survey of Interaction in Mixed Reality Systems. *In Symposium on Virtual Reality*, pp. 1-4.

Winkler, S. et al., 2007. Tangible reality desktop for digital media management. *In SPIE Engineering Reality of Virtual Reality*. Singapore, Vol. 6490.