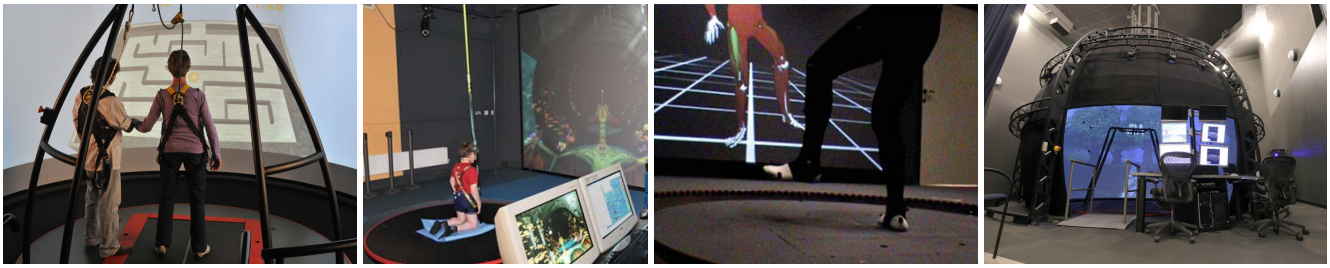
D-Flow: Immersive Virtual Reality and Real-Time Feedback for Rehabilitation

Thomas Geijtenbeek*
Games and Virtual Worlds
Utrecht University

Frans Steenbrink †
Motek Medical B.V.

Bert Otten
Human Movement Science
University of Groningen

Oshri Even-Zohar
Motek Medical B.V.



(a) Leeuwarden, NL (b) Liverpool, UK (c) Tel Aviv, IL (d) San Antonio, Tx, US

Figure 1: *Examples of virtual reality rehabilitation systems using D-Flow.*

Abstract

D-Flow is a software system designed for the development of interactive and immersive virtual reality applications, for the purpose of clinical research and rehabilitation. Key concept of the D-Flow software system is that the subject is regarded as an integral part of a *real-time feedback loop*, in which multi-sensory input devices measure the behavior of the subject, while output devices return motor-sensory, visual and auditory feedback to the subject. The D-Flow software system allows an operator to define feedback strategies through a flexible and extensible application development framework, based on visual programming. We describe the requirements, architecture and design considerations of the D-Flow software system, as well as a number of applications that have been developed using D-Flow, both for clinical research and rehabilitation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; J.3 [Life and Medical Sciences]: Health—;

Keywords: virtual reality, real-time, avatars, rehabilitation, computer simulation, biomechanical engineering, humans, postural balance, multi-display, motion capture

1 Introduction

In the last couple of decades, the idea of using virtual reality for the purpose of rehabilitation has spawned a myriad of custom-built virtual reality systems. The complexity of these systems ranges from high-end systems that use complex displays and hardware devices, to systems based on single displays and low-end hardware. Virtual reality systems have been used for several forms of rehabilitation, including gait training, stability training and treatment of phobias; target patient populations include various musculoskeletal and neurological disorders.

D-Flow is a software system running on Microsoft Windows, intended for clinical research and rehabilitation using a wide range of virtual reality systems. The key concept of D-Flow is that it regards the subject as an integral part of a *real-time feedback loop* (patented under US6774885, EP19990949671, WO0017767, CA2345013). Using D-Flow, an operator can develop and control interactive virtual reality applications that incorporate such a feedback loop, through a flexible and extensible application development framework that is based on visual programming. Since its inception more than a decade ago, the software has been subject to several iterations of redesign, based on feedback from users in both clinical and research settings. Currently, D-Flow is being used in combination with over 30 high-end virtual reality systems worldwide (see Figure 1 for examples).

The D-Flow software system is a combination of the following components: a layer for synchronized hardware communication, a multi-display rendering system, and a modular application development framework based on visual programming. Individually, these components are not new; similar components have been developed and put to use in many virtual reality, machine control or software development applications. The distinctive quality of D-Flow is that it combines these components into a single system, with a specific focus on clinical research and rehabilitation.

The remainder of this paper is organized as follows. In Section 2 we provide a brief overview of related work on virtual reality for clinical research and rehabilitation. Section 3 describes the requirements, architecture and design considerations of the D-Flow software system and its individual components. Section 4 contains a selection of module extensions that have been developed for D-Flow. In Section 5 we describe some of the safety considerations that have been taken into account while designing and implementing D-Flow. Section 6 contains a selection of applications that have been developed by clients using D-Flow, both for clinical research and rehabilitation. We conclude with an overview and pointers for future development (Section 7).

*e-mail:t.geijtenbeek@uu.nl
†e-mail:frans.steenbrink@motekmedical.com

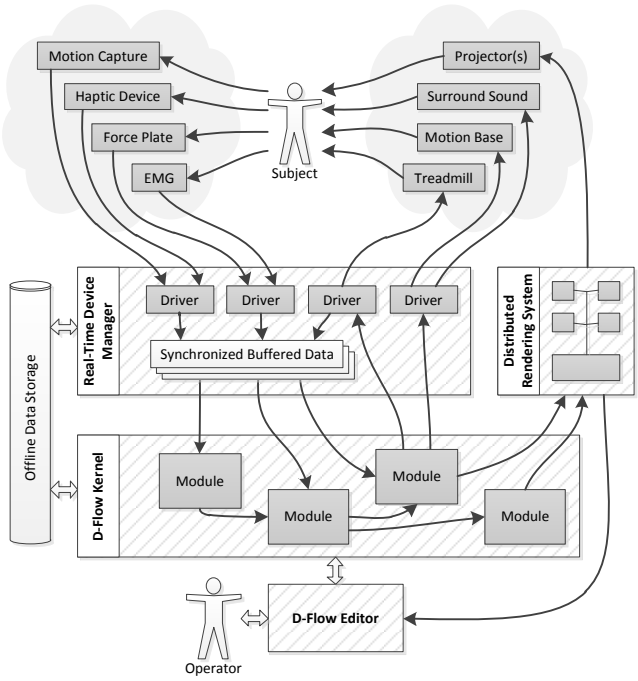


Figure 2: *An overview of a virtual reality system using D-Flow.*

2 Related Work

Rizzo and Kim [2005] provide an extensive overview of several of the virtual reality systems and software tools that have been developed for rehabilitation purposes. They point out that most of these systems are interesting as 'proof of concept', but not ready for clinical use. Recent publications on rehabilitation using virtual reality focus on stroke patients [Henderson et al. 2007; Saposnik and Levin 2011], traumatic brain injury [Rose et al. 2005], and various pediatric rehabilitation applications [Parsons et al. 2009]. Another area in which virtual reality has been applied is rehabilitation of service members, for instance in the *virtual Iraq* simulation [Gerardi et al. 2008].

3 The D-Flow Software System

In this section we provide an overview of the requirements, architecture and design considerations of the D-Flow software system. First, we will describe a virtual reality system using D-Flow as a whole, followed by descriptions of its individual components.

3.1 System Overview

Figure 2 displays an overview of a virtual reality system using D-Flow. The top section of the figure shows the subject as an integral part of a *real-time feedback loop* (patented under US6774885, EP19990949671, WO0017767, CA2345013 [Even-Zohar 2004]). In this feedback loop, a subject is measured through a set of *input devices*, whose data is processed in real-time and channeled to a set of *output devices* that return motor-sensory, visual and auditory feedback to the subject. Example input devices are motion capture systems, force plates and electromyography (EMG) devices. Example output devices are motion platforms, treadmills, audio devices and displays. The D-Flow software system itself consists of the following components:

Device type	Supported vendors / products
Motion Capture System	Vicon, NDI, NaturalPoint, Motion Analysis, XSens
6-DOF Motion Platform	Bosch, Moog, Sarnicola
3-DOF Motion Platform	Sarnicola, Forcelink
Treadmill	Bertec, Forcelink, HP-Cosmos
Force Plate	AMTI, Bertec, Kistler, Forcelink
Analog Input / EMG	Vicon, National Instruments TMSI, Phidgets, Labjack
Haptic Device	HapticMASTER CyberGlove, CyberGrasp
Input Controller	DirectX, Phidgets

Table 1: *An overview of devices currently supported by RTDM. Additional devices are supported on request.*

- The *Real-Time Device Manager* (RTDM), which is responsible for real-time synchronized communication with all connected devices.
- The *Distributed Rendering System*, which is responsible for synchronized rendering of a virtual environment on different display devices.
- The *D-Flow kernel*, which is the real-time framework running the actual interactive virtual reality programs (so-called *D-Flow applications*).
- The *D-Flow Editor*, which is the user interface that allows efficient and user-friendly development of D-Flow applications.

In the remainder of this section we will describe in detail each of these four components.

3.2 Real-Time Device Manager (RTDM)

The *Real-Time Device Manager* (RTDM) is responsible for managing communication between the D-Flow kernel and hardware devices. The requirements of the RTDM are defined as follows:

- Perform safe, high-performance communication with both input and output devices.
- Provide consistent interfaces to different types of devices, hiding vendor-specific details from the application framework where possible.
- Organize all data from acquisition devices into a single, synchronized, buffered data block, with full and flexible access.
- Allow data to be stored in a file, using popular formats that can be opened in other applications.
- Allow simulation of hardware devices in an off-line session, using previously stored data.

In essence, the RTDM maintains a set of device drivers, each of which can implement a number of interfaces for specific device types. Table 1 contains an overview of device types, along with supported vendors. To ensure continuous operation as much as possible, each device driver runs in a separate thread. Like other components in the D-Flow software framework, drivers are implemented in C++ to achieve a proper balance between performance and maintainability. In most cases, drivers do not directly control hardware devices; instead they communicate with dedicated host machines (either through TCP/IP or UDP/IP).

Data from acquisition devices is stored in a *synchronized data buffer*, which can be flexibly accessed by the D-Flow kernel. Data is stored buffered in memory with timing and frame number information. The individual drivers are responsible for converting timing information and taking into account device-specific delays. Data can be interpolated or extrapolated when a requested time stamp has no explicit counterpart in a data stream. Such interpolation or extrapolation only occurs on data retrieval, ensuring an optimal memory footprint without data loss. Components in the D-Flow kernel that perform real-time data processing can be signaled when data of a specific type becomes available.

The synchronized data buffer can optionally be streamed to a file, supporting many formats, including the C3D file format (`www.c3d.org`) and a custom ASCII-based format. Previously stored data can be used for off-line simulation of specific data streams.

3.3 D-Flow Kernel

The D-Flow kernel is the framework in which the actual virtual reality applications can be defined. Its requirements are as follows:

- The data flow performance must allow for a real-time feedback loop, as described in [Even-Zohar 2004].
- The framework must (both in terms of flexibility and functionality) enable development of several types of interactive applications, suitable for rehabilitation and clinical research.
- It should be possible to easily add new functionalities to the framework: the framework must be *extensible*.
- It should be possible to modify and control the application parameters in real-time.
- The software should incorporate features that ensure safe operation of the virtual reality system.

Modules D-Flow is based on the concept of *modules*: manageable components with a specific functionality, which can be combined to create complex, interactive virtual reality applications. There are various types of modules present in the D-Flow software system. Some modules directly control specific hardware devices, such as a treadmill or a motion base. Other hardware modules provide access to real-time data streams from live input devices. To allow interaction between the subject and the virtual environment, there are modules that manipulate virtual objects, control playback of animations in the virtual environment, or detect collisions between objects. Finally, several modules perform specific low-level tasks, and can be regarded as building blocks for high-level functionalities. Examples are modules for controlling decision logic, variable storage and function generation. D-Flow also contains a general purpose scripting module and a module for expression parsing.

New modules are being developed regularly, while existing modules are continuously enhanced. A total overview of all modules is beyond the scope of this paper; however, in Section 4 we present an overview of a selection of D-Flow modules.

Inter-Module Communication The D-Flow kernel supports two methods of communication between module instances: data-based and event-based. Each module contains a number of named input and output channels. An output channel from one module can be connected to input channels from other modules, effectively generating a flow of data from one module to the next. Any module can access data streams from acquisition devices through the Real-Time Device Manager, by querying the synchronized data buffer described in Section 3.2.

In addition to data-based communication, the D-Flow kernel framework allows for event-based communication between modules. The operator can define a set of *global events*, which can be broadcast by modules at specific occurrences (i.e. when a collision between virtual objects has occurred, or a value reaches a certain threshold). Next to that, each module exposes a set of *module actions* that affect the behavior of the module in a specific way (i.e. enabling or disabling it, showing or hiding an object, or increasing a counter). For each module instance, any global event can be set to trigger any module action, enabling maximum flexibility in event-based communication.

Main Processing Loop The D-Flow kernel operates in a frame-by-frame fashion, where each frame consists of four main steps:

1. Wait for a signal from real-time device manager that new data is ready from a specific device (optional).
2. For each module, perform the *module actions* linked to events scheduled for broadcast during the previous frame.
3. For each module, perform internal processing:
   (a) Update user interface parameters (if appropriate).
   (b) Generate new output data based on input data.
   (c) Modify objects in the virtual reality environment.
   (d) Schedule broadcast of any number of global events for the upcoming frame.
4. All operations modifying the virtual reality environment are handled and displays are updated through the Distributed Rendering System (see Section 3.4).

The order in which modules are processed is based on the direction in which they are connected. In cases of loops, modules that use streamed input data from the synchronized data buffer are processed first.

3.4 Distributed Rendering System (DRS)

The Distributed Rendering System (DRS) is responsible for maintaining a virtual environment, and for synchronized rendering on multiple displays. Its requirements are:

- It should have an API that is independent of the real-time rendering engine, to allow for use of different implementations.
- It should support synchronized rendering on a large number of external displays, each with different camera setups and resolutions, independent of display type (monitor, projector or HMD). It should also support stereoscopic displays.
- All elements should be controlled with a latency low enough to support real-time feedback.
- The level of control should include any operation desired for real-time virtual reality applications, such as object transformation, skeleton-based character animation, real-time avatar display and texture animation.
- The API must be customizable and extensible.

Figure 3 shows an overview of the DRS components, each of which is described in the remainder of this section.

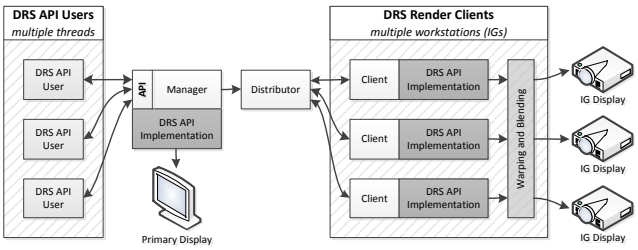


Figure 3: *A schematic overview of the Distributed Rendering System (DRS).*

DRS API The virtual environment is accessed through the DRS API, which is based on the concept of operations: atomic functionalities that can be serialized and efficiently transported over a network connection. Operations are initially handled by the *manager*, who stores them in an *operation queue* and applies them to a primary environment (see also Figure 4). At specific intervals, the buffered operations are transferred to the *distributor*, who transfers the serialized operations to each individual client through a TCP/IP connection. The distributor is also responsible for synchronized rendering: after sending all operations to individual clients, it waits for conformation from each client that operations have been processed and rendered off-screen. When all conformations are received, the distributor sends a signal to swap the on-screen and off-screen display buffers. After another conformation, it begins collecting operations for the next update. Exact frame-locking between the client displays can optionally be implemented in hardware.

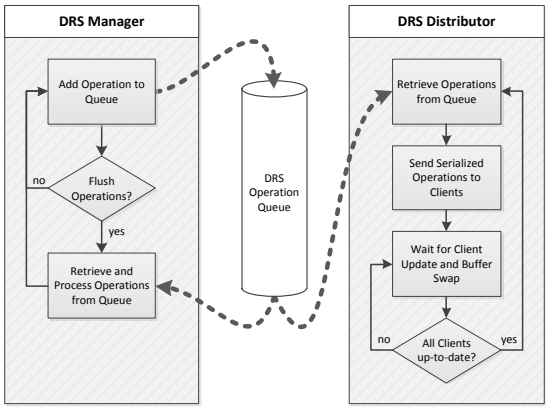


Figure 4: *Detailed display of the roles of the DRS Manager and DRS Distributer.*

Operations DRS operations include loading and destroying of objects, object transformations and control of animation. Real-time avatars can be controlled by setting angles of the joints of a skinned mesh object. Multi-camera and stereoscopy are performed by setting appropriate camera and frustum settings for each client display individually.

API Implementation The actual rendering of the environment is performed by an implementation of the DRS API. The current API implementation is based on the Ogre rendering engine (`www.ogre3d.org`).

3.5 D-Flow Editor

The D-Flow editor is the component that forms the bridge between the operator and the application defined in the D-Flow kernel. It is the interface by which applications are developed, and by which the parameters of an application can be controlled during real-time operation. The design of the D-Flow editor is roughly based on the following set of requirements:

- Provide interfaces for application development and testing, as well as day-to-day use of an application.
- The development interface should be easy to use by operators without an engineering background.
- The user interface should be responsive at all times, suitable for smooth real-time modification and control of application parameters.

Editor Figure 5 displays an overview of the D-Flow Editor window. There exist separate sections for organizing modules, editing connections and assigning global events to module actions. Applications can be documented through grouping, labeling, and commenting. All editing is done in a separate thread, updates to the D-Flow kernel are scheduled and applied at specific moments.

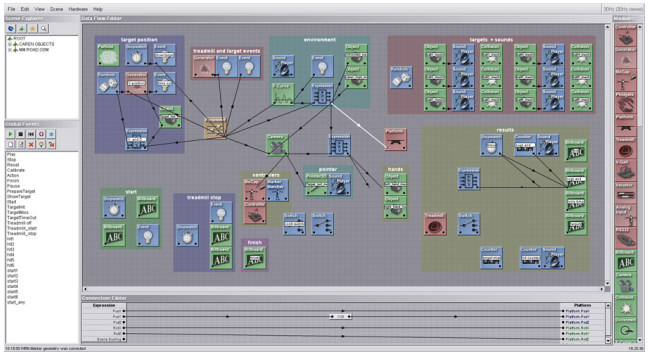


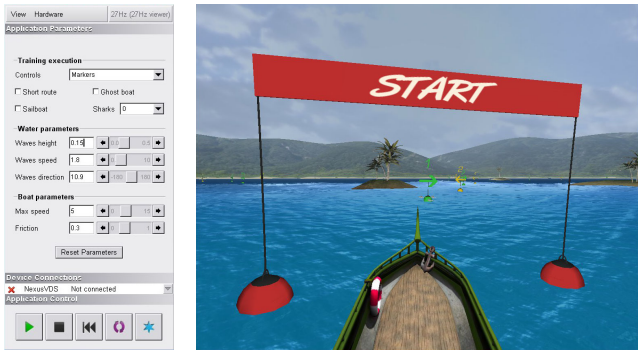
Figure 5: *The D-Flow editor.*

3D Display The 3D display allows the operator to see what the subject sees. Currently, the operator can perform camera orbit, zoom, panning, and object selection. Future versions will add the possibility to modify object transformation and generate paths for animation. For virtual reality systems containing a single display, the operator 3D display can be directly cloned to be used as 3D display for the subject, omitting the use of image generators.

Runtime Console The Runtime Console is a customizable interface that can be used to control D-Flow applications (see Figure 6(a)). The goal of the interface is to provide a clean and efficient interface, suitable for day-to-day use of D-Flow applications.

The Runtime Console GUI can be edited interactively. It is incorporated into the D-Flow kernel through a module that automatically generates an output channel for each GUI element. GUI elements that link to data channels include sliders, drop-down boxes and check boxes, while buttons can be created and set to broadcast specific global events.

In addition to user-defined controls, the runtime console contains an interface for controlling the state of the motion base, as well as a control for monitoring connection the status of all hardware devices in the virtual reality system.



(a) (b)

Figure 6: *The D-Flow Runtime Console (a) and the operator 3D Display (b).*

4 D-Flow Modules

In this section we present a non-exhaustive selection of D-Flow modules, along with intended purpose and motivation.

Human Body Model (HBM) The *Human Body Model* (HBM) performs a biomechanical analysis to estimate and visualize muscle forces of a human subject in real-time (as patented in [Even-Zohar and van den Bogert 2009]). For input, it uses a real-time data stream consisting of marker and force plate data. The output of the module consists of joint angles, joint moments and powers, muscle lengths and speeds, and muscle forces. The muscle forces are estimated using static optimization of squared muscle stress, based on modeled maximum forces. The different processing steps of the HBM module are schematically presented in Figure 7; a detailed description is provided by Van den Bogert et al. [2007].

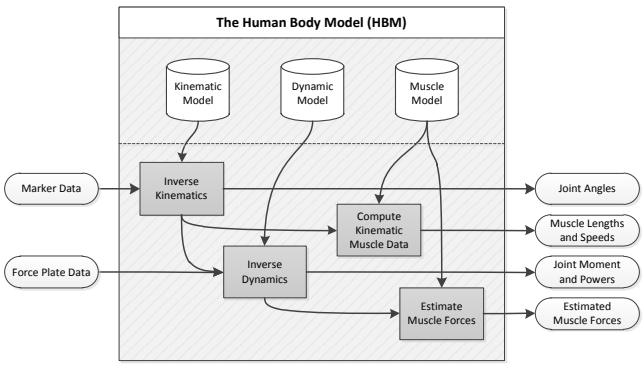


Figure 7: *A schematic overview of the different components of the HBM Module.*

Figure 8 shows the HBM module in a live situation, with real-time visualization of muscle forces on an avatar. The estimated muscle forces can be used to drive any element in the virtual reality, opening up several possibilities for rehabilitation, sports or clinical research.

Self-Paced Treadmill The *Self-Paced Treadmill module* enables self-selected walking speed for a subject on a treadmill. The speed of the treadmill is controlled using the position and acceleration of the subject (as measured by a motion capture device), using a proprietary algorithm. The measured speed of the treadmill can subsequently be used to control the visual flow in the virtual environment.

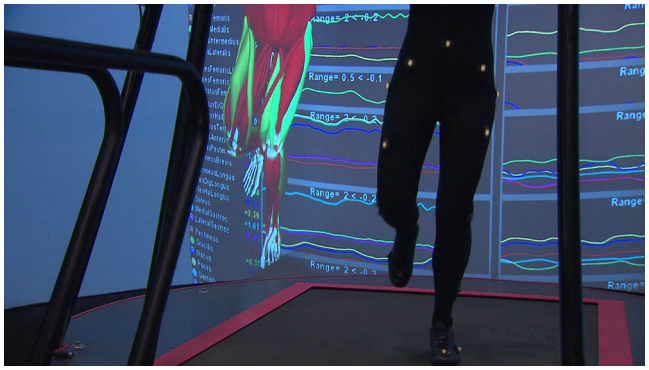
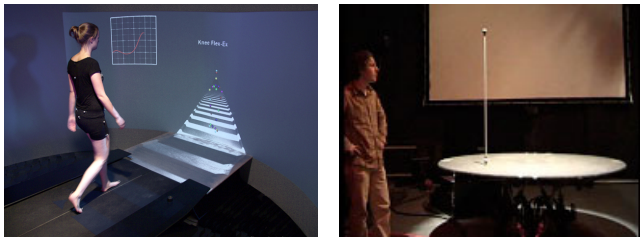


Figure 8: *The Human Body Model in action, with real-time avatar displaying estimated muscle force.*

Network Communication This module has been designed to allow safe, real-time interaction between different virtual reality systems that use D-Flow. This allows for applications in which multiple subjects in different locations and different hardware setups are engaged in a single virtual environment.

Balance Stick This module can balance a stick on a motion platform, based on the position of two optical markers. It is developed as to demonstrate the real-time feedback loop (see Figure 9(b)).



(a) Gait Analysis (b) Balance Stick

Figure 9: *Examples of D-Flow modules in real-life applications.*

Gait Analysis The *Gait Analysis module* produces various parameters useful for gait analysis, such as step information and joint angles (see Figure 9(a)). The module uses both motion capture and force plate data. Part of the module is a step detection algorithm [Roerdink et al. 2008], which is used to determine spatial gait parameters like step duration and step size. All parameters are computed in real-time, and can be used as part of a training application. This module can be used in conjunction with the self-paced treadmill module described earlier.

Pointer The *Pointer module* can use input from a motion capture device to simulate a 3D pointer. It computes pointing direction, and uses the collision detection framework of the D-Flow kernel to measure if specific virtual objects intersect with the virtual line by two 3D markers. The module can be used in shooting simulations, or other applications requiring pointing gestures.

Script The *Script module* is a general purpose module in which the application developer can program any functionality missing from other modules. The user can define inputs and outputs, broadcast and receive global events, and control virtual reality objects. The script module is based on the Lua scripting language [Ierusalimschy et al. 2007].

5 Risk Management and Safety

Since the D-Flow software system is intended for use with live subjects in a medical environment, the software needs to be compliant with risk management and safety regulations for medical devices. Therefore, all development takes place under the regime of an ISO 13485 Quality Management System and is in compliance with the IEC 62304 standard for software life-cycle processes. Risk management is enforced as an integral part of the D-Flow software development process, according to the ISO 14971 standard. The IEC 62304 and ISO 14971 standards are adopted as European norm and are recognized by the US FDA and many other regulatory authorities.

An example in which these standards are applied is *emergency response*. This is implemented in the Real-Time Device Manager through a *MIP driver* (short for *Main Integration Panel*). The MIP driver is in constant communication with a MIP device that monitors the current state of the system. In case of an emergency situation, the software will automatically suspend its operation and wait for a manual emergency reset. Similarly, software failures are monitored through communication with a watchdog component in the MIP device. In case of a communication timeout, the emergency circuit will break and all devices will respond in an appropriate manner.

Hardware support is also developed according to the ISO 13485, which includes the ISO 14971 for risk management and the IEC 60601-1 for General Product Safety. An example of this is that on installation, systems that include a motion base or treadmill must be equipped with a safety portal or safety harness to prevent subjects from falling.

6 D-Flow Applications

In this section we will provide of overview of various ways in which D-Flow has been used in both research and clinical settings. Its purpose is to demonstrate applicability of D-Flow, not to evaluate results of any publications mentioned.

6.1 Research Applications

Several institutions have used D-Flow to perform research on human motion. Most publications involve research on response to virtual reality, stability or compensation strategies.

Stability D-Flow has been used in several occasions for research on human stability. McAndrew et al. [2010; 2011] have performed an extensive set of studies on identification of dynamic instability. They have utilized the D-Flow software system in combination with an instrumented treadmill on a motion base and a 300° surround display using 8 projectors. In their application, they apply pseudo-random perturbations to the support surface and visual field. Additionally, Hak et al. [2011] examined changes in gait parameters after platform perturbations, using a cylindrical screen, a motion base and an instrumented treadmill. Jessop and McFadyen [2008] have performed research on postural stability using binaural-bipolar galvanic vestibular stimulation.

Figure 10: *A D-Flow application developed for research on fear-of-heights.*

Applicability of Virtual Reality The D-Flow software system has been used in various publications that investigate to what degree virtual reality applications can be used to simulate real life situations. Gerin-Lajoie et al. [2008] compare obstacle avoidance behavior in both physical and virtual reality, using a head-mounted display. Darekar et al. [2011] use D-Flow for additional research on obstacle avoidance in virtual environments. Kizony et al. [2010] show how functional virtual reality can be useful for dual-task training for elderly.

Hawkins et al. [2008] show a relation between game speed and performance in terms of core stability in healthy subjects. Makssoud et al. [2009] show how control of a motion base can be used to optimize walking in a virtual environment. D-Flow has also been used for research on the effect of virtual reality on subjects with fear-of-heights, using a setup with a semi-cylindrical screen, a 3-DOF motion platform and an instrumented treadmill (see Figure 10).

6.2 Clinical Applications

Balance Training In addition to research on human balance, the D-Flow software has also been used for developing applications for balance training. Barton et al. [2011a] have developed a serious gaming application targeted for children suffering from *Cerebral Palsy* (CP). In this application (entitled *Goblin's Post Office*), children use their pelvis to steer a flying dragon towards target objects, while avoiding others (See Figures 1(b) and 11).

Bugnariu and Fung [2007] use D-Flow to develop a training application for postural stability of elders, aiming at fall prevention. Vrieling et al. [2008] use D-Flow to create a training program in which amputees are tested on their balancing abilities. In their application, the subject is offered complex dual tasks, while being engaged in a challenging VR environment on a moving motion platform. Everding and Kruger [2011] demonstrate how a virtual boat-ride application can be used for amputee balance training.

Gait Training Another well-studied clinical application of the D-Flow software system is gait training. Fung et al. [2006] develop an application for post-stroke gait training. In this application, they use visual flow in combination with a self-paced treadmill to help increase walking spreads for post-stroke patients. Lamontagne et al. [2007] show how walking speeds can be increased by changing the optical flow of the environment. Makssoud et al. [2009] demonstrate a D-Flow application intended for stroke rehabilitation, in which subjects walk on a self-paced treadmill, mounted on a motion platform that mimics changes in terrain encountered in the virtual environment. Darter and Wilken [2011] demonstrate a D-Flow application for gait training that makes a subject adapt to a

Figure 11: *Balance training for Cerebral Palsy children using the Goblin's Post Office application at John Moores University, Liverpool, UK [Barton 2010].*

symmetrical gait pattern (See Figure 12). Mert at al. [2010] demonstrate an application that helps patients with decreased hip power to improve their gait by applying contra-lateral platform rotations.

Miscellaneous Barton et al. [2011b] use D-Flow to create a virtual mirror box. This research project is based on reported improvement of motion following treatment with a real mirror box [Ramachandran and Rogers-Ramachandran 1996]. To overcome the limitations of physical mirrors, and to extent freedom-of-motion, D-Flow has been used to generate delayed mirrored images of non-affected limbs during gait (see Figure 13).

De Groot et al. [2003] use D-Flow to perform a body analysis that is used to increase shoe efficacy of subjects with movements problems. Lees et al. [2007] develop an application to find the best treatment option for balance training, by measuring the kinematic response characteristics after a perturbation of a subject on a motion platform. Berard et al. [2011] use D-Flow to develop an application that uses visual flow to provoke changes in head motion, which can be used for training of stroke patients. Subramanian et al. [2007] develop an application for arm rehabilitation for patients with motor deficits due to central nervous system lesions.

7 Summary

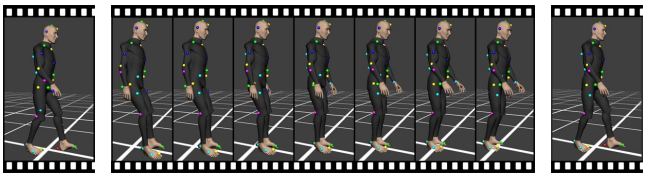
We have described the D-Flow software system, as well as several applications developed for clinical research and rehabilitation. The wide range of applications that have been developed so far supports the claim of flexibility and applicability of the software system.

In the future, we expect an increasing number of research and rehabilitation facilities to be using the D-Flow software system for development of immersive virtual reality applications that contain a real-time feedback loop.

Acknowledgements

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO), as well as the GALA project, funded by the European Union in FP7.

Figure 12: *Gait training for amputees at the Brooke Army Medical Center (BAMC), San Antonio, Tx [Darter and Wilken 2011].*



(a) (b) (c)

Figure 13: *The virtual mirror box: (a) the original avatar; (b) the same avatar, mirrored; (c) the same avatar, mirrored and delayed on one side to simulate symmetric gait. [Barton et al. 2011b]*

References

BARTON, G. J., HAWKEN, M. B., FOSTER, R. J., HOLMES, G., AND BUTLER, P. B. 2011. Playing the Goblin Post Office game improves movement control of the core: A case study. In *Int. Conf. on Virtual Rehabilitation 2011*, no. 978.

BARTON, G., DE ASHA, A., GEIJTENBEEK, T., AND ROBINSON, M. 2011. Development of a Virtual Mirror Box for spatial and temporal manipulation of visual feedback on body movement during gait: A technical evaluation. In *ESMAC*.

BARTON, G. 2010. Virtual rehabilitationa focus on movement function. *Biomechanica Hungarica 2*, 2, 7–14.

BERARD, J., FUNG, J., AND LAMONTAGNE, A. 2011. Optic flow in a virtual environment can impact on locomotor steering post stroke. In *Int. Conf. on Virtual Rehabilitation*.

BUGNARIU, N., AND FUNG, J. 2007. Aging and selective sensorimotor strategies in the regulation of upright balance. *Journal of neuroengineering and rehabilitation 4* (Jan.), 19.

DAREKAR, A., ARAVIND, G., LAMONTAGNE, A., AND FUNG, J. 2011. Perceptual and navigational strategies for obstacle circumvention in a virtual environment. In *Int. Conf. on Virtual Rehabilitation 2011*.

DARTER, B., AND WILKEN, J. 2011. Gait Training With Virtual RealityBased Real-Time Feedback: Improving Gait Performance Following Transfemoral Amputation. *Physical Therapy 21*, 2 (Apr.), 111–2.

DE GROOT, I., EVEN-ZOHAR, O., HASPELS, R., VAN KEEKEN, H., AND OTTEN, E. 2003. Case study: CAREN: A novel way to improve shoe efficacy. *Prosthetics and Orthotics International 27*, 2 (Jan.), 158–162.

EVEN-ZOHAR, O., AND VAN DEN BOGERT, A. J., 2009. Method for real time interactive visualization of muscle forces and joint torques in the human body (US2009082701, US7931604), Apr.

EVEN-ZOHAR, O., 2004. System for dynamic registration, evaluation, and correction of functional human behavior (US6774885, EP19990949671, WO0017767, CA2345013).

EVERDING, V. Q., AND KRUGER, S. E. 2011. Virtual reality enhanced balance training for Service Members with amputations. In *Int. Conf. on Virtual Rehabilitation 2011*.

FUNG, J., RICHARDS, C. L., MALOUIN, F., MCFADYEN, B. J., AND LAMONTAGNE, A. 2006. A treadmill and motion coupled virtual reality system for gait training post-stroke. *Cyberpsychology & behavior 9*, 2 (Apr.), 157–62.

GERARDI, M., ROTHBAUM, B. O., RESSLER, K., HEEKIN, M., AND RIZZO, A. 2008. Virtual Reality Exposure Therapy Using a Virtual Iraq : Case Report. *Journal of Traumatic Stress 21*, 2, 209–213.

GÉRIN-LAJOIE, M., RICHARDS, C. L., FUNG, J., AND MCFADYEN, B. J. 2008. Characteristics of personal space during obstacle circumvention in physical and virtual environments. *Gait & posture 27*, 2 (Mar.), 239–47.

HAK, L., HOUDIJK, H., STEENBRINK, F., MERT, A., BEEK, P. J., AND DIEËN, J. V. 2011. Speeding Up or Slowing Down; How to Deal with Balance Perturbations During Gait? In *ISB*.

HAWKINS, P., HAWKEN, M., AND BARTON, G. 2008. Effect of game speed and surface perturbations on postural control in a virtual environment. In *Proc. of the 7th ICDVRAT*, 311–318.

HENDERSON, A., KORNER-BITENSKY, N., AND LEVIN, M. 2007. Virtual reality in stroke rehabilitation: a systematic review of its effectiveness for upper limb motor recovery. *Topics in stroke rehabilitation 14*, 2, 52–61.

IERUSALIMSCHY, R., DE FIGUEIREDO, L. H., AND CELES, W. 2007. The evolution of Lua. *Proc. of the third ACM SIGPLAN conf. on History of programming languages*, 2–1–2–26.

JESSOP, D., AND MCFADYEN, B. J. 2008. The regulation of vestibular afferent information during monocular vision while standing. *Neuroscience letters 441*, 3 (Aug.), 253–6.

KIZONY, R., LEVIN, M., HUGHEY, L., PEREZ, C., AND FUNG, J. 2010. Cognitive load and dual-task performance during locomotion poststroke: a feasibility study using a functional virtual environment. *Physical therapy 90*, 2, 252.

LAMONTAGNE, A., FUNG, J., MCFADYEN, B. J., AND FAUBERT, J. 2007. Modulation of walking speed by changing optic flow in persons with stroke. *Journal of neuroengineering and rehabilitation 4* (Jan.), 22.

LEES, A., VANRENTERGHEM, J., BARTON, G., AND LAKE, M. 2007. Kinematic response characteristics of the CAREN moving platform system for use in posture and balance research. *Medical engineering & physics 29*, 5 (June), 629–35.

MAKSSOUD, H. E., RICHARDS, C. L., AND COMEAU, F. 2009. Dynamic control of a moving platform using the CAREN system to optimize walking in virtual reality environments. In *Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, vol. 2009, 2384–7.

MCANDREW, P. M., DINGWELL, J. B., AND WILKEN, J. M. 2010. Walking variability during continuous pseudo-random oscillations of the support surface and visual field. *Journal of biomechanics 43*, 8 (May), 1470–5.

MCANDREW, P. M., WILKEN, J. M., AND DINGWELL, J. B. 2011. Dynamic stability of human walking in visually and mechanically destabilizing environments. *Journal of biomechanics 44*, 4 (Feb.), 644–9.

MERT, A., GLASHOUWER, P., WERTHEIM, W. J., FRUNT, T., AND VAN DER WURFF, P. 2010. Case report (Dutch). *Nederlands Militair Geneeskundig Tijdschrift 63*, 5, 161–164.

PARSONS, T., RIZZO, A., ROGERS, S., AND YORK, P. 2009. Virtual reality in paediatric rehabilitation: A review. *Developmental Neurorehabilitation 12*, 4, 224–238.

RAMACHANDRAN, V. S., AND ROGERS-RAMACHANDRAN, D. 1996. Synaesthesia in phantom limbs induced with mirrors. *Proceedings. Biological sciences / The Royal Society 263*, 1369 (Apr.), 377–86.

RIZZO, A. S., AND KIM, G. J. 2005. A SWOT Analysis of the Field of Virtual Reality Rehabilitation and Therapy. *Presence: Teleoperators and Virtual Environments 14*, 2 (Apr.), 119–146.

ROERDINK, M., COOLEN, B. H., CLAIRBOIS, B. H. E., LAMOTH, C. J. C., AND BEEK, P. J. 2008. Online gait event detection using a large force platform embedded in a treadmill. *Journal of biomechanics 41*, 12 (Aug.), 2628–32.

ROSE, F. D., BROOKS, B. M., AND RIZZO, A. A. 2005. Virtual reality in brain damage rehabilitation: review. *Cyberpsychology & behavior 8*, 3 (June), 241–62; discussion 263–71.

SAPOSNIK, G., AND LEVIN, M. 2011. Virtual reality in stroke rehabilitation: a meta-analysis and implications for clinicians. *Stroke 42*, 5 (May), 1380–6.

SUBRAMANIAN, S., KNAUT, L. A., BEAUDOIN, C., MCFADYEN, B. J., FELDMAN, A. G., AND LEVIN, M. F. 2007. Virtual reality environments for post-stroke arm rehabilitation. *Journal of neuroengineering and rehabilitation 4* (Jan.), 20.

VAN DEN BOGERT, A. J., GEIJTENBEEK, T., AND EVEN-ZOHAR, O. 2007. Real-time estimation of muscle forces from inverse dynamics. *health.uottawa.ca*, 5–6.

VRIELING, A. H., VAN KEEKEN, H. G., SCHOPPEN, T., OTTEN, E., HOF, A. L., HALBERTSMA, J. P. K., AND POSTEMA, K. 2008. Balance control on a moving platform in unilateral lower limb amputees. *Gait & posture 28*, 2 (Aug.), 222–8.