5-2017

Framework for embedding physical systems into virtual experiences

Saurabh Hindlekar
Clemson University, shindle@g.clemson.edu

Framework for embedding physical systems into virtual experiences

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
Saurabh Hindlekar
May 2017

Accepted by:
Dr. Victor Zordan, Committee Chair
Dr. Brian Malloy
Dr. Donald House

Table of Contents

List of Figures

Abstract

Figure 1: System in action

We present an immersive Virtual Reality (VR) experience through a combination of technologies including a physical rig; a gamelike experience; and a refined physics model with control. At its heart, the core technology introduces the concept of a physics-based communication that allows force-driven interaction to be shared between the player and game entities in the virtual world. Because the framework is generic and extendable, the application supports a myriad of interaction modes, constrained only by the limitations of the physical rig (see Figure 1). To showcase the technology, we demonstrate a locomoting [11] robot placed in an immersive gamelike setting.

Chapter 1

Introduction

Force-based interaction is proposed as a communication form that allows individuals to "feel" the influence of game entities in an immersive virtual environment. Not unlike a human-sized haptic device, our base hardware is a unique third-party rig [18] (see Figure 1.1) that holds the human player (safely) inside.

Figure 1.1: Physical rig responsible for providing haptic feedback

This real-world physical device is engaged through a simulated (game) world in which players interact. The rig has an analogous virtual entity that acts as a physics-based proxy within the simulation. And, complete with a specialized control system, this unique player avatar responds to the virtual forces applied to it by producing an appropriate disturbance and correction that expresses the force to the human seated inside.

In this framework, the system's controller acts to respond and orient the player avatar in the presence of virtual forces, torques, and impulses. We can apply forces directly to the avatar to simulate impact, for example, from a hit of enemy fire. In addition, we can create the effect of locomotion by producing appropriately timed and placed ground reaction forces. Because the controller can generically respond to

multiple forces, nothing prevents us from creating a two-legged locomotion (e.g. a walking biped) or to do this while also receiving enemy fire. To the extent that our physical rig can accelerate the desired amount, the system can faithfully express the impacts to the player as a set of activations to the rig's motors.

The physical rig has three stepper motors (see Figure 1.2) along the three axes of rotation (roll, pitch and yaw). Each stepper motor is activated through three separate signals roll, pitch and yaw. Each motor is divided into an equal number of steps. The motor is asked to move and stay at one these steps through the signal that is sent to it by custom drivers. Main limitations of using stepper motors are the "chatter" effect at low speeds and lower torque at high speeds.

Figure 1.2: Stepper motors on the physical rig, Roll, Pitch and Yaw (Left- Right)

System overview.

Our physical rig (see Figure 1.1) is a customized third-party hardware that we use as a 3-degree of freedom (roll, pitch, yaw) motion simulator. This device is controlled through stepper-motors that take their input from custom drivers communicating with our physical avatar which is in turn controlled by the players through joysticks mounted inside the rig. The user wears a stereo headset (Oculus) [14] that offers added immersion through look control and stereovision. With this custom hardware set-up, we design a user experience that transport the players from the venue to

a virtual world, allowing them to be trained and tested before pitting them in battle with non-player entities. The game software is written in Unity 3D [20] and takes advantage of Unity's physics and game development tools.

The premiere technologies of the system include: force-based interaction between the player and virtual entities; a generic approach for force production and response; and, along with the latter, a controller for biped stepping. Further, a key to immersion is the consistent combination of visual and movement cues.

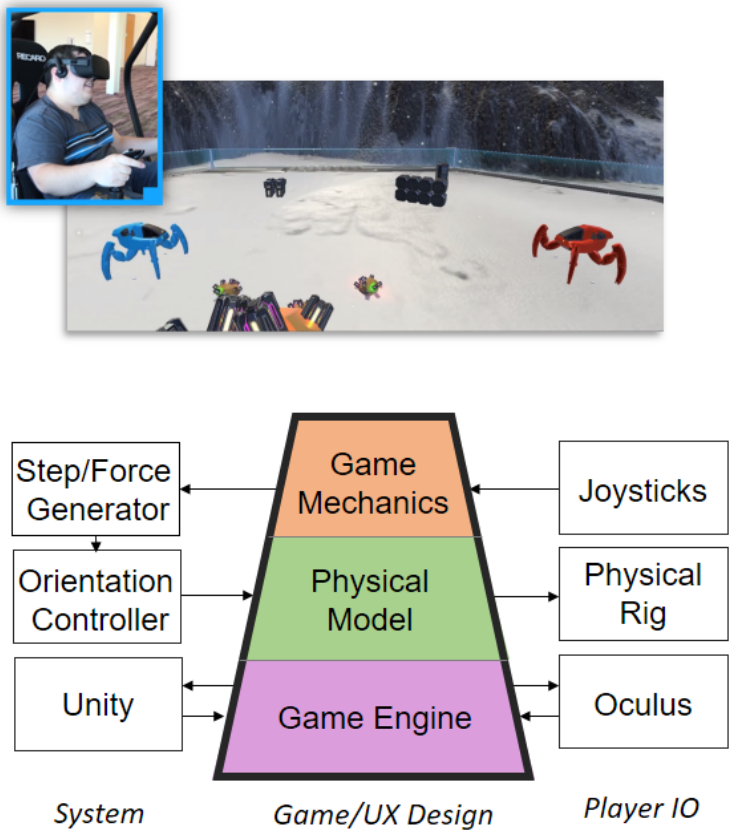

Figure 1.3: System overview

The layout (see Figure 1.3) describes how parts of the system both real (joysticks, physical rig [18], Oculus [14]) and abstract (force generator, orientation con-

troller, game design) interact with the respective modules in the framework.

The physical model that represents the player character in the virtual world includes the orientation controller, stepping controller and the player avatar is discussed in Chapter 2. The interaction between the physical model and the physical rig is also discussed.

The game engine and its intricacies that comprise the heart of this application are discussed in Chapter 3. The input system, impulse generator, level design are some of topics elaborated in this chapter. The unique design decisions made for this game application are also discussed here.

In the following chapters we discuss how each module was designed for this specific application that is an arcade styled game. This framework can serve as a blueprint for other such applications especially since we see a rise in research in the area of such virtual reality simulations.

Chapter 2

Background and related work

In this chapter, we discuss the works that were referred to for guidance in building this framework. First, we discuss works that inspired the physical model for the player character. Then we discuss background research in motion simulation and game design.

2.1 Dynamic physical model and animation of a Whole-Body response

In our work, we splice the kinematics and the dynamics within the simulation following Ye and Liu [23] and Nguyen et al. [13]. The game engine is a layering of two models, the kinematic and the dynamic physical controller. The orientation controller is similar to the effect of a virtual actuator in Pratt et al [15] and Zordan, Hodgins [24], Coros et al. [6].

Nguyen et al. [12] describe an approach to create physically plausible reactions of a rigid body to external forces. They achieve this in two ways, dynamic articulation necessary for local-body responses and whole-body reactions to disturbances, each

computed separately and controlled separately.

The rigid body model in their work is affected by three factors namely the external disturbance force, the force due to gravity and the ground reaction force. The position and the orientation of the rigid body model are updated based on two controls to balance and right the RB model respectively using a Cartesian based servo. The effect of this control is a virtual actuator for the Center of Mass of the RB "pulling" it toward the reference motion. The reference motion here refers to motion capture data.

The central piece of this framework is our own whole-body reaction controller, not unlike the one described in the article above for the rigid body (physical model) embodying the human in the virtual world.

2.2 Motion simulation in Virtual Reality

If real walking is compared with virtual walking and flying in virtual environments then it has to be concluded that real walking feels the most immersive. This is demonstrated in Usoh et al. [21] in which the authors conduct a set of experiments and a questionnaire-based method that they used to determine overall virtual presence. The virtual presence was divided into two behavioral presence and subjective presence.

We carefully design the game to build and maintain the overall virtual presence of the player. The self-motion induced is carefully designed based on bio-mechanical data for biped or quadruped locomotion.

An experiment that is used to study how motion perception in the human visual system of motion can be exploited to employ selective rendering of frames in an animation was demonstrated by Ellis and Chalmers et al. [8]. This can save time

and computing power. The authors use research conducted on vestibular physiology to develop the simulation of motion using a motion simulator. The models of the vestibular system are used to predict when the image quality can be dropped without the user noticing. The experiment involves sitting in the pod which is a hexapod and an Indiana Jones styled mine shaft ride.

Ames et al. [1] describe the process of achieving bipedal robotic walking through controller synthesis inspired by human locomotion. Using Lyapunov functions and quadratic equations the data from human motion is constrained and models are developed for bipedal robotic walking. These controllers are then used on real robots AMBER 1.0 and AMBER 2.0 to achieve the true realisation of the controller.

Simple visual cues like short physical jerks in a wheelchair have been used to simulate self-motion. Riecke et al. [16] analyse the effectiveness of such cues. There were three methods used Button-based method, Joystick-based method and Wheelchair-based method. Force-feedback input device provides the force feedback and this solution is cheap, elegant and easy to set up.

Using passive cueing vs active curing to enhance self-motion has been an area of research in virtual reality[17]. Riecke et al. [17] perform two other investigations namely the effect of rotational velocity on curvilinear vection and effect of minimal motion cueing to enhance self-motion. The study uses a gyroxus gaming chair. The participant has to lean in specific directions for active cueing. In passive cueing, an experimenter performs similar trajectories while the participant is sitting in the chair.

In our framework we give the individual active visual and haptic feedback, design the feedback so that it is immersive with the user experiencing convincing self-motion.

2.3 Game design

Game design and a way to formalize it has been studied in recent years. Taylor et al. [19] survey the different approaches made to describe a game flow process and development. The drawbacks of those approaches and finally they describe an approach based on developing use-cases defined in Unified Modeling Language (UML). They also state that Game design is an emerging software developmental process that must be formalized especially with more complex game designs coming out. It will help to communicate the game at a higher level between the different people involved in building a game like artists, musicians, programmers, UX designers, UA testers and so on. The authors also show why its difficult to formalize the process because there are many end users and everyone looks at the game differently. Their approach works satisfactorily for tightly scripted games but for games that are loose ended it does not work because it becomes too convoluted and complex.

From the analysis of their results we concluded that a tightly scripted story would work better for this game. It would allow us to train and test the individual to control the robot.

Wilson et al. [22] put across the concept of designing a game in which the player tried to understand and have a dialogue with the designer rather than the game being designed to please the player and all the player has to do is understand the game system.

In virtual reality, game design not only pertains to traditional elements of the process but also to the task of maintaining immersion in the virtual world which is an equally challenging design problem. Bian et al. [4] talk about flow and how a VR experience can be designed so that the player is always in flow and never out of it.

We try to always maintain immersion and identify places in the story where

the player could have a break in immersion. In such places the player is re spawned.

Chapter 3

Physical model

At the core of the proposed approach for a physical proxy is an orientation controller that rights the player's avatar following a disturbance. The concept driving this choice is that the agent is self-correcting. In this fashion, one may regard the orientation controller as a set of virtual shocks that comply in the presence of impact and return the character to its upright posture. The physical model is formed from the rigid body representation of the player character - the physical proxy and the player avatar that houses this proxy along with the two different controllers; orientation and stepping.

The physical model forms the black box that takes player input and produces tangible forces, impulses and readjustments to set-points that are fed to the rigid body along with the orientation controller. The orientation of rigid body proxy is fed to the physical rig.

The design of the physical model that handles the interpretation of forces from the virtual world to the real world contributes to a novel virtual experience. It has two main driving factors.

1. Identification of the causes for the forces and position in the world space where

they originate.

2. Detailing the force profile curves that will be applied to the rigid body and the different parameters related to the force profile.

3.1 Player avatar

The player avatar forms the outer shell for the player in the virtual world. The player avatar that the proxy will be inside of can range from four wheeled vehicles to giant Mechs or quadrupeds. We assume that the player avatar has negligible mass and contribute the entire mass of the physical model to the physical proxy rigid body. For this game application, the player avatar is a giant biped robot (see Figure 3.1) or a quadruped Mech (see Figure 3.2). Since this is an imaginary character we looked at biomechanical data for biped and quadruped locomotion [7, 2, 9] and designed the character using information and data on different Mechs available online through Mech wikis [3]. The weight, speed and dimensions of the Mech were some of the important values examined. Using these values the body of the Mech was approximated to one rigid body inside Unity.

Figure 3.1: Player avatar as a biped

Figure 3.2: Player avatar as a quadruped

3.2 Rigid Body Approximation

Applying torques, forces and impulses to the rig and providing sensible haptic feedback from these torques to the player is a critical part of this endeavor. Having the players feel forces that are believable given they are inside a physical rigid body is of the highest importance.

To have an efficient way of bringing out these real life forces although from a virtual simulated environment we use the concept of rigid body approximation [12] to mimic the dynamics of the character that the player would embody in the virtual world.

We create a rigid body approximation of the character the player is going to be inside and call it the proxy. We treat the proxy as a rigid body which is non-flexible and placed inside the character.

This novel technique of abstracting away the dynamics of the real world physical rig and replacing them with the dynamics of a virtual rigid body proxy gives us the ability to simulate a variety of player avatars, experiences and simulations.

Once the rigid body approximation of the player character is complete, all the forces the character experiences are applied to it, the proxy.

3.3 Orientation controller

The orientation controller uses torque-based (virtual) proportional derivative (PD) servos along the three directions of rotation, each analogous to a single stepper-motor in the physical rig. The PD servo has a set-point selected specifically depending on the player avatar required by the application, usually, it is set to zero representing an upright stance. As the controller receives forces from the world or through the

stepping controller the spring yields in the direction of the force, damps out any subsequent velocities and returns the physical proxy rigid body to the desired pose. The system can be tuned to give a range of experiences based on the size of the influences. We opted for a slightly under-damped system after empirical testing.

With the aggregate influence of the forces from the virtual world, we create physically plausible reaction forces for the proxy. The reaction forces are directed by three controls,τ_x, τ_y, τ_z that right the proxy rigid body after impact. The following equations describe the reaction forces due to the three controls, $(\bar{\theta} - \theta)$ represents the tracking error and $(\dot{\theta})$ represents the joint velocity

$$\tau_x = k_x(\ \hat{\theta_x} - \bar{\theta_x}) - d_x(\ \dot{\theta_x})$$
$$\tau_y = k_y(\ \hat{\theta_y} - \bar{\theta_y}) - d_y(\ \dot{\theta_y})$$
$$\tau_z = k_z(\ \hat{\theta_z} - \bar{\theta_z}) - d_z(\ \dot{\theta_z})$$

Thus disturbances to the physical rigid body create accelerations that are defined by the summation of external forces and the reaction forces via the orientation controller. These accelerations for the proxy RB in turn lead to orientation differences in the physical rig, leading to the resulting real-world/physical movement of the player.

With the orientation controller in place, control for locomotion is converted into the force production of footsteps. Likewise, other force-based interactions can be designed through the purposeful development of appropriate forces and their application to the physical rigid body.

While designing these interactions the forces due to friction and due to gravity can be appropriately accounted for within these interactions. The effect of gravity on an unbalanced proxy is an interesting addition that can be implemented in a future work.

3.4 Stepping controller

A fundamental behavior that immerses the player in our game world is the production of stepping and locomotion. The design of the production of these steps depends heavily on the gait. A gait can be defined as the movement of limbs of animals or mechanical limbs in case of robots.

The parameters that control gait (in this case biped gait) include the variance of step length, step frequency with speed. For example, step frequency varies with speed, stabilizing as the speed approaches its maximum. The same applies to step length.

Further, for each step, there is a signature ground reaction force (GRF) profile that is indicative of the mechanical components of the locomotor. Within our step production, our choice of where and when the steps appear is based first in biomechanical data [7, 2, 9] and then tuned to improve the user experience.

To match the GRF of natural animals, we build force profiles using sets of Gaussian curves (see Figure 3.3) and add parameters to vary the mean and variance of the curve.

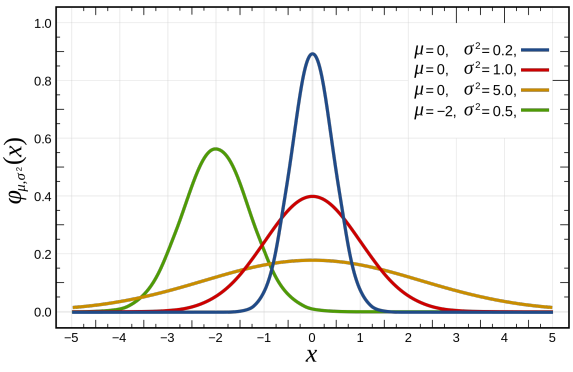

Figure 3.3: Sample set of gaussians

Since our character is an imaginary robot, we designed a stepping profile that *felt* appropriate based on our "Mech" in the game. During runtime, the placement of steps in the virtual world is controlled by the player, e.g. depending on how fast or slow the player wants to move.

A sample Gaussian curve (see Figure 3.4) for one step for a biped would include two peaks where the second peak is lesser than the first one. The first peak represents the ground reaction force when the foot hits the ground and the second peak represents the ground reaction force when the foot pushes off the ground.

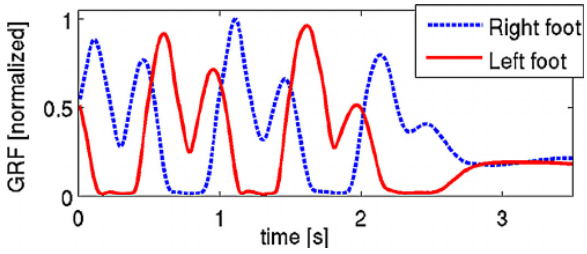

Figure 3.4: Force profile for human gait. Each step includes two peaks where the second peak is lesser than the first one. The first peak represents the ground reaction force when the foot hits the ground and the second peak represents the ground reaction force when the foot pushes off the ground.

Through experimentation, we found that the forces' positions, overall shape, and duration are important aspects in providing a nuanced and believable physical experience.

From the GRF profiles, we account for both the step placement and the forward motion of the robot in the game to compute the torque defined by, $\hat{T}$ for each step. That is, to determine the torque $\hat{T}$

$$\hat{T} = \hat{r_t} \times \hat{F_t}$$

for each frame of the animation (see Figure 3.5), the time-varying torque arm $\hat{r}_t$ is computed based on the state and crossed with the time-varying ground reaction force $\hat{F}_t$.

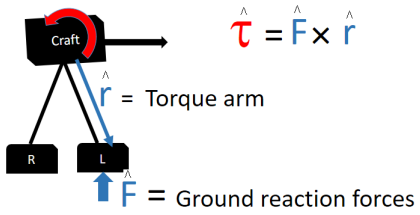

Figure 3.5: Illustration showing how the torque is computed at a particular time step

The aggregate of the resulting set of torques caused by the accurate placement of each foot according to the defined gait (see Figure 3.6) is then applied to the physical proxy rigid body along with the orientation controller.

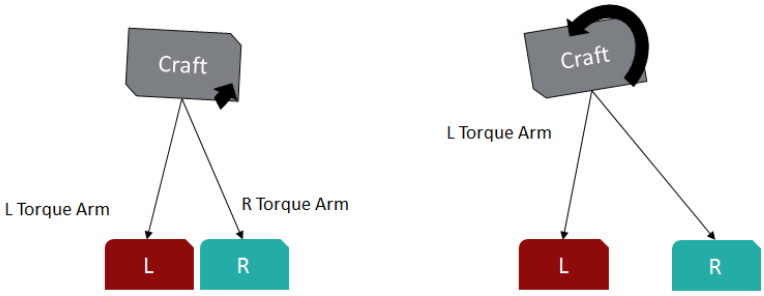

Figure 3.6: Torques generated by the stepping controller applied to the physical rigid body

In this fashion, the design of the ground reaction force profile curve inter-plays

with the orientation controller as it adds its own reaction torques via the PD servos. The aggregate is what is experienced by the player housed in the physical rig every time a force is created in the virtual world.

Generation and implementation of styled stepping gaits

From biomechanical data [9, 7, 5] we derived stepping gaits to test our stepping controller. The generic nature of our controllers allows us to build multi-leg locomotion like that of a quadruped. Most of the time quadrupeds use the walking gait in which the footfalls for the same side are out of phase by about 25% of the step duration and footfalls between the right and left side are out of phase by 75% of the step duration. In the bound gait observed in animals like squirrels the back legs move together, half a period out of phase with the front pair. The generation of these gaits (see Figure 3.7 and Figure 3.8) using our stepping controller is described next.

The design of foot placement, along with the Gaussian curves affect the final experience via haptic feedback in interesting ways. First, the foot patterns are generated, then based on these foot patterns the force profiles for each footfall are produced (see Figure 3.7).

The duration of each step depends on the frequency of stepping which in turn depends on the speed of the player in the virtual world. Each step is divided into the stance state and the swing state. While the step is in the stance state it implies that the foot is on the ground and is receiving forces from the ground. In swing state the player avatar leg is animated in a basic kinematic manner to the next foot position. Since we were able to generate footfall patterns for a specific gait, we explored performing inverse kinematics [10] to animate each leg using the foot positions as end effectors. The duration of each footstep affects the rate of change of the torque arm $\hat{r}_t$. The swing state of the swinging foot occurs simultaneously during

the stance state of the current standing foot.

The changing torque arm $\hat{r_t}$ for each foot is computed and crossed with the force profiles produced. In this way, the torque, $\hat{T}$ from each footfall is computed and applied to the proxy RB along with the orientation controller.

The orientation controller applies its own reaction forces to the proxy RB. The orientation of the proxy RB is transmitted to the physical rig as shown in the final graph (see Figure 3.7).

In this way, our system is capable of generating multi-legged styled stepping locomotion.

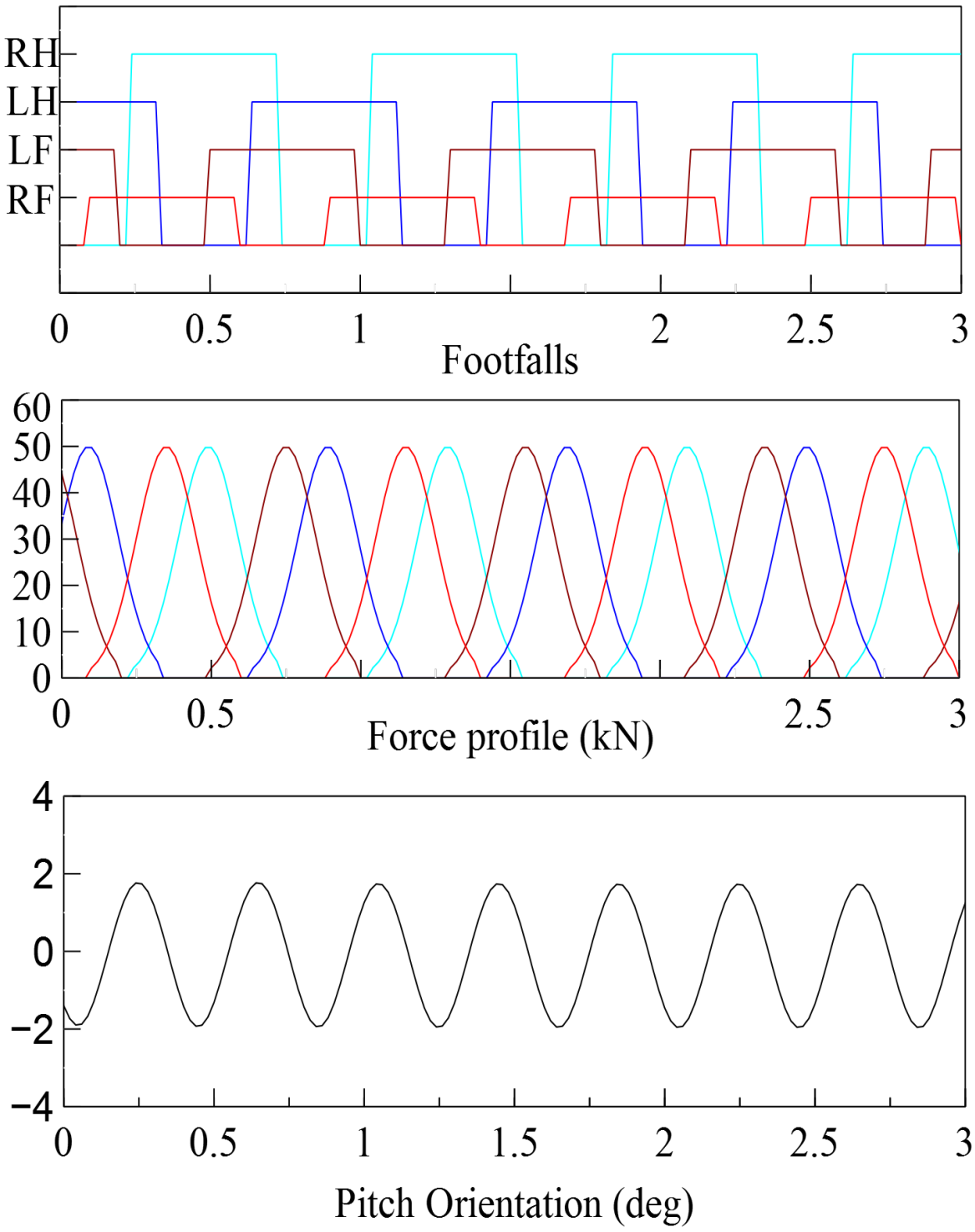

Figure 3.7: System generated walk gait: footfalls, force profiles and the final orientation change for the physical rig, over time (sec)

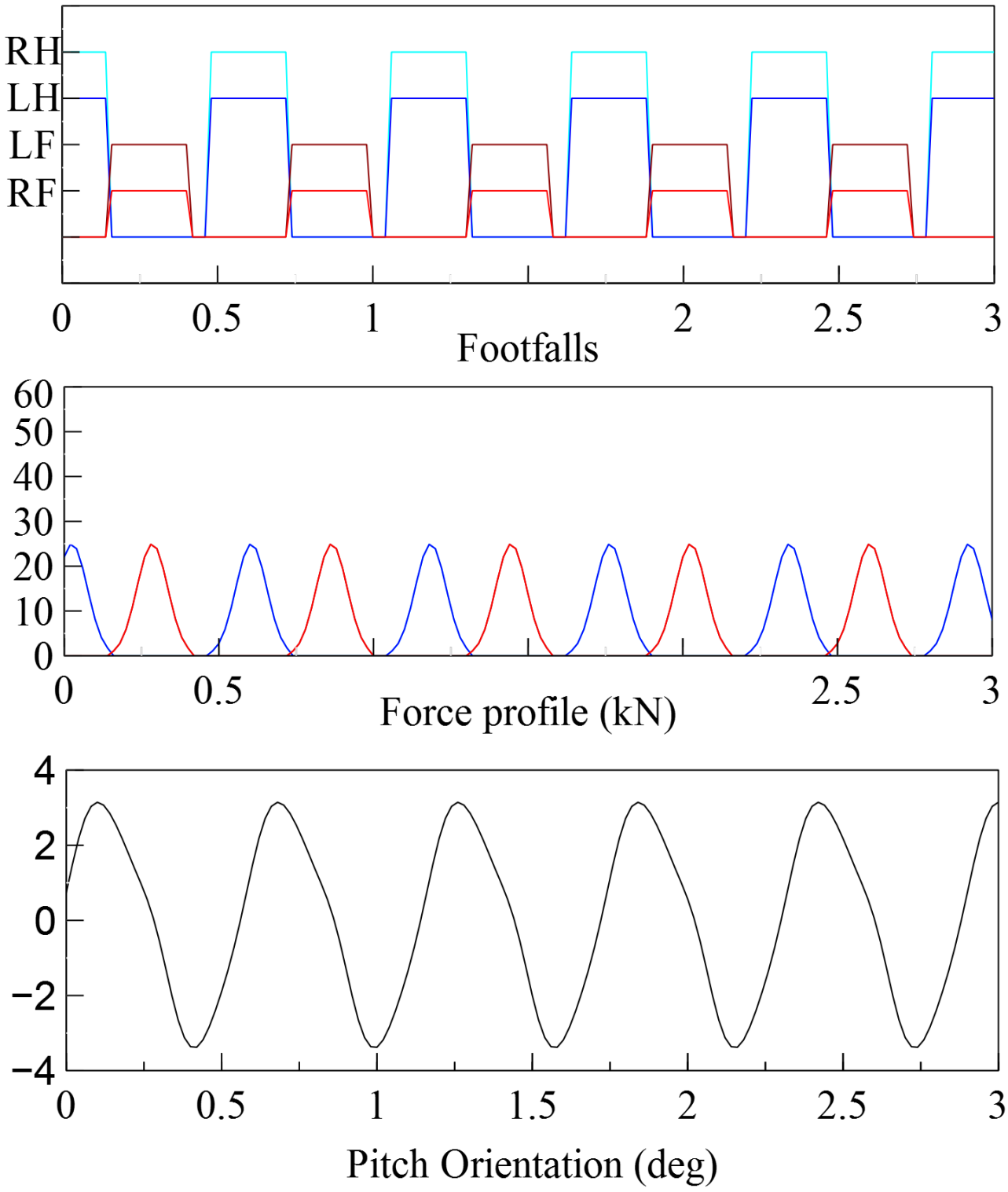

Figure 3.8: System generated bound gait: footfalls, force profiles and the final orientation change for the physical rig, over time (sec)

Chapter 4

Game design

The technologies that affect the game engine design are Unity [20], Oculus and the physical rig. This trio forms a unique combination with each having its own challenges and limitations. The game engine is designed to enable the three components to work in harmony.

Though we found that these particular components work well together, they can be switched out for other components that suit a particular application. The framework still remains the same, but new wrappers would have to be built to allow such replacement of components.

The type of input and how it's obtained has got great weight while deciding how the game engine controller is built. Locomotion was the main focus when deciding on the type of input because it was this interaction that gave the most amount of motion feedback to the player. While the other actions also provide motion feedback to the player through the simulation they are time and context specific.

Any video game has obstacles but having a motion rig allowed us to be more creative. In this regard, for this application, two other design choices that need to be mentioned are - recoil from the blasters of the Mech and tilting floors.

We discuss the system implementation, design of the level, locomotion design tweaks, and designing external force productions in that order. The user design choices made to tweak the final experience of the user inside the application are detailed next.

4.1 System implementation

To support the smooth transfer of the proxy state from inside Unity to the mechanical rig, we use a custom driver which we designed to be robust, consistent and low-latency. Specifically, a client-server was built using the .NET framework to from the bridge. The drivers send orientation values to the mechanical rig in degrees.

Coupled with the technologies we develop a rich game loosely surrounding the theme of escaping a destroyed falling space station. The game experience is designed so that the players gradually ease themselves into the world. We start with basic navigation, target practice and finally progressing to fighting with virtual entities (in this case drones).

It was important to have the players to be able to look around in a hands-free manner which would add to the sense of presence in the virtual proxy.

We wanted the players to be able to focus on controlling the Mech and the finesse required in controlling a big robot. The use of Oculus gives the player freedom shoot where they look. While wearing the Oculus the players can look around the cockpit which helps to build a sense of presence in the virtual world.

In conclusion having the Oculus wearable on the players and track their head movement is a very critical component since it gives the players the freedom to develop a sense of presence in the virtual world and not be burdened with too many controls.

4.2 Input system

The input system is a critical design choice in any entertainment interaction based system, especially in a video game. The ideal design is to have an input system that is not cumbersome, does not overwhelm the players and they can hit the ground running as soon they start playing the game. In designing our system we stuck to these three ideals.

The player control (in our case to control the Mech) is through two joysticks, and two triggers one on each joystick. These controls are similar to tank controls (see Figure 4.1). The player pushes both joysticks forward to move forward, back to move backwards. One forward, one back to change direction and turn towards the left if the left is pushed back. The triggers are for performing special actions/ interactions in the game like shooting. The speed at which the player moves forward is based on how far the joysticks are pushed forward together. Having joysticks enabled the player to move in a more regulated, granular fashion. This form of input significantly provided us with more detail which we harnessed to design smooth player movement.

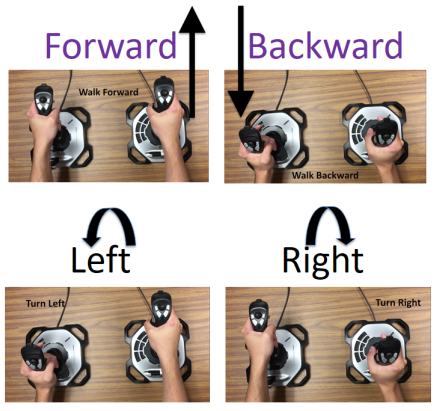

Figure 4.1: Player control

Initially, we did implement a more intuitive, player input invested simulation (single step walking), having each joystick control one leg of the biped (in our case, a Mech). It was too much work on the part of the players and decided not have such low-level input from the players. On the other hand, we did make sure that the control the players have (for the Mech) is granular and sensitive to the extent that the players feel like they are moving but from inside a mechanical body.

Taking intuitive granular and detailed input from the players, to interpret into high-level actions such as fast or slow walking was the design we opted for.

4.3 Impulse generator

The stepping controller outlines the design of force production for locomotion. The design of other force-based interactions in this game application is similar. The player avatar is a Mech which has Mech blasters that allow the player avatar to shoot at enemies. Another important aspect in designing a Mech based game is shooting. Shooting is an essential ingredient for this player avatar (Mech). Having the Mech blasters be rigid would have meant that the player avatar (Mech) can only shoot in the direction it is facing. This would have limited the player.

Whenever the player hits the trigger, based on which trigger the player hits a combination of torques is applied accordingly.

Without the recoil from the blasters, the player would feel a loss in immersion. Small design choices like this one help complete the immersion.

4.4 Readjustment of set-points due to environment

Having an interactive environment helps to provide higher immersion by allowing the environment to give immediate haptic feedback. This is what we tried to achieve with the design of the tilting floors obstacle, (see Figure 4.2). For the tilting floors if the player is standing on one of the tilting floors then the set-point of that particular PD servo is offset according to the angle of the tilt. If the players remain standing on the tilting floor for too long then they lose ground and fall over.

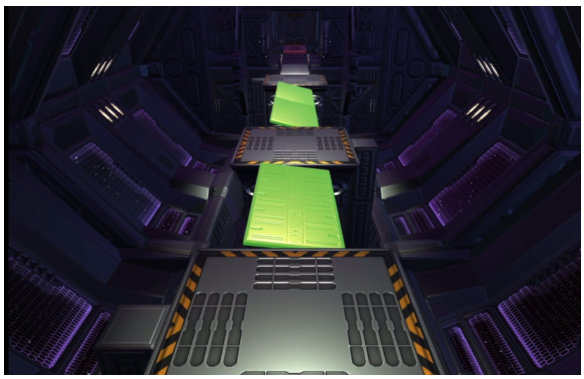

Figure 4.2: Designing interactive environments

The tilting floors obstacle is an interesting scenario because essentially the environment is changing and affecting the player avatar.

4.5 Level design

The virtual environment that the players observe and interacts with the virtual world, it has to gradually pull the players ever deeper into the virtual world. At all times it must be consistent and there should not be any breaks in immersion. The levels were designed so that the players gradually eases themselves into the world

and the obstacles get more and more challenging as the players progress through the world.

The first challenge for the players would be to get accustomed to the controls. Which meant the first level would allow the players to take complete control of the movement, the players should be able to walk, turn and look around. The first level was designed so that it leads the player always forward by intention and allows the players to understand the granular control for movement (of the Mech).

Special actions (shooting) were the next objective for the players to master and level two was designed to enable the players to perform these actions (in our case we had stationary targets) without any harm to themselves. This level is subtle nudge for the players to learn how to perform special actions in a timely and accurate way. The final level is designed so that the players need to use the two interactions i.e. movement and special actions (shooting the Mech blasters) in context (to take down enemies that could hurt the player).

The gradual ascendance (see Figure 4.3) of challenges allows the players to master nuanced control over the game itself and have fun. The players are always progressing and do not feel stuck at a particular place. The story and the environment were designed in such a way that the player experiences one consistent virtual world. Any obstacles, breaks in immersion were all handled so that the player is never jerked back into reality. A tightly scripted story allowed us to identify all the spots in the virtual world where the player could feel a break in immersion. In all such situations, the player was re-spawned to the beginning of that particular level. To complete the immersion the sounds are an important aspect. We have a voice helping the players navigate the levels, and other special sounds like an alarm in the background to display urgency and sounds for steps.

All these design choices keep the player engaged while never letting the player

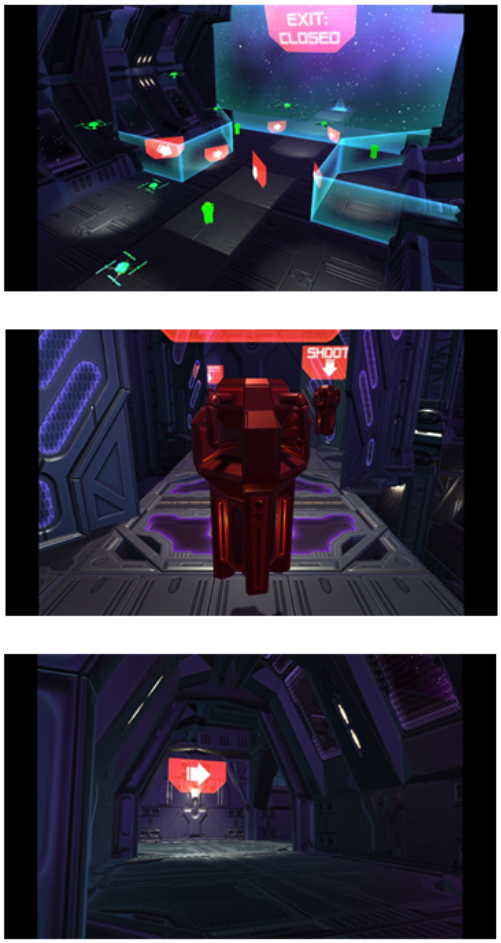

Figure 4.3: Level 1, Level 2, Level 3 (Bottom - Top)

break out of the immersion.

4.6 Perception of steps through the yaw module

When humans take a step in place to turn, we accelerate our torso in that direction. Through the physical system and visual feedback we try to simulate this acceleration for a player avatar. We do this by applying a torque around the y-axis to the rigid body physical proxy along with the orientation controller. The visual feedback constitutes the orientation change of the rigid body. This gives the individual the perception of taking a step in that direction. If the input is provided continuously to the system then the torque is applied in intervals. Each torque corresponding to a step. In this way we give the individual the perception of stepping.

4.7 Perception of lean

As the velocity of a person increases the amount that person leans forward also increases. In short, lean is directly proportional to the velocity of the person in real life. We wanted to make the motion of the Mech as similar to that of a human as possible in the hope that the players experiencing the motion would feel a sense of similarity and recognise the motion instantly. A lean controller was designed to perform the function of setting the lean angle of the Mech based on the velocity of the Mech. It is separate from the stepping controller and directly controls the set point angle of the rigid body approximation much like the tilting floors do. As the speed of the Mech increases the lean angle increases and vice versa.

The availability of the lean controller means that the players have an additional sense of familiarity in the sense of motion that they feel.

Chapter 5

Conclusion and Future Work

The various aspects, from technology to design, culminates in an immersive VR experience that is unique and engaging. We anticipate that this line of research is uncovering new potential in motion simulation that has a strong footing in entertainment, but may also see benefits in serious testing and training.

Accounting for forces due to gravity in locomotion is an interesting addition that we are looking at. The translation of the center of mass using a similar PD Servo based controller is a future work that would be an interesting integration to the already sophisticated controller that rights orientation changes. These are three of the most visible, possible avenues for improvement on this work.

Limitations to date include the limited degrees of freedom and ranges of the rig we adopted. Because the rig only has orientation (and no translation), direct hits (aimed at the center of the proxy) cannot be communicated effectively because their torque is negligible. Six degress of freedom would be necessary to be completely generic, but such systems are cost prohibitive and raise (interesting) engineering challenges. To extend this type of work to real world applications, beyond entertainment, more rigorous testing and verification is necessary.

Bibliography

[1] Aaron D Ames. Human-inspired control of bipedal robots via control lyapunov functions and quadratic programs. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 31–32. ACM, 2013.

[2] TP Andriacchi, JA Ogle, and JO Galante. Walking speed as a basis for normal and abnormal gait measurements. *Journal of biomechanics*, 10(4):261–268, 1977.

[3] BattleMech. `http://www.sarna.net/wiki/BattleMech`, 2017.

[4] Yulong Bian, Chenglei Yang, Fengqiang Gao, Huiyu Li, Shisheng Zhou, Hanchao Li, Xiaowen Sun, and Xiangxu Meng. A framework for physiological indicators of flow in vr games: construction and preliminary evaluation. *Personal and Ubiquitous Computing*, 20(5):821–832, 2016.

[5] James J Collins and Ian N Stewart. Coupled nonlinear oscillators and the symmetries of animal gaits. *Journal of Nonlinear Science*, 3(1):349–392, 1993.

[6] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Generalized biped walking control. In *ACM Transactions on Graphics (TOG)*, volume 29, page 130. ACM, 2010.

[7] Rod Cross. Standing, walking, running, and jumping on a force plate. *American Journal of Physics*, 67(4):304–309, 1999.

[8] G Ellis and Alan Chalmers. The effect of translational ego-motion on the perception of high fidelity animations. In *Proceedings of the 22nd Spring Conference on Computer Graphics*, pages PAGE–7. ACM, 2006.

[9] Timothy M. Griffin, Russell P. Main, and Claire T. Farley. Biomechanics of quadrupedal walking: how do four-legged animals achieve inverted pendulum-like movements? *Journal of Experimental Biology*, 207(20):3545–3558, 2004.

[10] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 522–531, New York, NY, USA, 2004. ACM.

[11] Saurabh Hindlekar, Victor B. Zordan, Emerson E. Smith, John C. Welter, and William Garrett Mckay. Mechvr: Interactive vr motion simulation of "mech" biped robot. In *ACM SIGGRAPH 2016 VR Village*, SIGGRAPH '16, pages 14:1–14:2. ACM, 2016.

[12] N Nguyen, R Arista, K C Liu, and V Zordan. Adaptive dynamics with hybrid response. *Siggraph Asia (Technical Brief)*, 2012.

[13] Nam Nguyen, Nkenge Wheatland, David Brown, Brian Parise, C Karen Liu, and Victor Zordan. Performance capture with physical interaction. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 189–195. Eurographics Association, 2010.

[14] Oculus. `https://www.oculus.com/en-us/`, 2017.

[15] Jerry Pratt, Peter Dilworth, and Gill Pratt. Virtual model control of a bipedal walking robot. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 193–198. IEEE, 1997.

[16] Bernhard E Riecke. Simple user-generated motion cueing can enhance self-motion perception (vection) in virtual reality. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 104–107. ACM, 2006.

[17] Bernhard E Riecke and Daniel Feuereissen. To move or not to move: can active control and user-driven motion cueing enhance self-motion perception (vection) in virtual reality? In *Proceedings of the ACM Symposium on Applied Perception*, pages 17–24. ACM, 2012.

[18] SimCraft. `http://www.simcraft.com/`, 2017.

[19] Mark John Taylor, David Gresty, and Michael Baskett. Computer game-flow design. *Computers in Entertainment (CIE)*, 4(1):5, 2006.

[20] Unity3D. `http://www.unity3d.com/`, 2017.

[21] Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. Walking is greater than walking-in-place which is greater than flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[22] Douglas Wilson and Miguel Sicart. Now it's personal: on abusive game design. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology*, pages 40–47. ACM, 2010.

[23] Yuting Ye and C Karen Liu. Animating responsive characters with dynamic constraints in near-unactuated coordinates. In *ACM Transactions on Graphics (TOG)*, volume 27, page 112. ACM, 2008.

[24] Victor Brian Zordan and Jessica K Hodgins. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96. ACM, 2002.