VirtualBricks: Exploring a Scalable, Modular Toolkit for Enabling Physical Manipulation in VR

Jatin Arora
Weave Lab, IIIT-Delhi
New Delhi, India
jatina@iiitd.ac.in

Aryan Saini
Weave Lab, IIIT-Delhi
New Delhi, India
aryan15134@iiitd.ac.in

Nirmita Mehra
Weave Lab, IIIT-Delhi
New Delhi, India
nirmita16174@iiitd.ac.in

Varnit Jain
Weave Lab, IIIT-Delhi
New Delhi, India
varnit15112@iiitd.ac.in

Shwetank Shrey
Weave Lab, IIIT-Delhi
New Delhi, India
shwetank16095@iiitd.ac.in

Aman Parnami
Weave Lab, IIIT-Delhi
New Delhi, India
aman@iiitd.ac.in

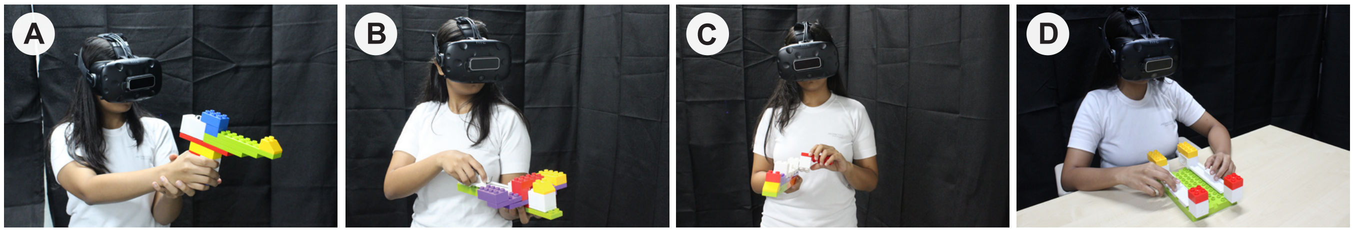


Figure 1: We present VirtualBricks, a toolkit to create custom controllers for Virtual Reality that enable a variety of Physical Manipulation interactions [A: Shooting targets using a gun, B: Launching objects by pulling the strings of a Slingshot, C: Catching a fish by rotating the fishing reel, D: Moving fences to let the horses escape.]

ABSTRACT

Often Virtual Reality (VR) experiences are limited by the design of standard controllers. This work aims to liberate a VR developer from these limitations in the physical realm to provide an expressive match to the limitless possibilities in the virtual realm. VirtualBricks is a LEGO based toolkit that enables construction of a variety of physical-manipulation enabled controllers for VR, by offering a set of feature bricks that emulate as well as extend the capabilities of default controllers. Based on the LEGO platform, the toolkit provides a modular, scalable solution for enabling passive haptics in VR. We demonstrate the versatility of our designs through a rich set of applications including re-implementations of artifacts from recent research. We share a VR Integration package for integration with Unity VR IDE, the CAD models for the feature bricks, for easy deployment of VirtualBricks within the community.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; **User interface toolkits**.

KEYWORDS

Passive haptics, Physical manipulation, Construction Toolkit

ACM Reference Format:
Jatin Arora, Aryan Saini, Nirmita Mehra, Varnit Jain, Shwetank Shrey, and Aman Parnami. 2019. VirtualBricks: Exploring a Scalable, Modular toolkit for Enabling Physical Manipulation in VR. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland Uk.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300286

1 INTRODUCTION

Virtual Reality (VR) aims to mimic and augment the real world to create an alternate reality for the user. Recent advancements in head-mounted displays (HMDs), body tracking and ambisonics have enabled the development of immersive VR experiences. However, the controllers that come along with the commodity VR systems (like HTC Vive[1] and Oculus[2]) are unable to haptically render the diverse shapes of the virtual objects. Moreover, these controllers support free object manipulation in six degrees of freedom, as a standard. However, many of the objects that we encounter in everyday lives are not 6DoF manipulatable, but allow motion across specific axes — other axes being constrained by structure. For instance, a steering wheel can be rotated across its hinge but cannot be manipulated in any other manner. Though such interactions have been approximated using existing controllers, the object's structure remains missing, leading to improper physical constraints and hence limited immersion. To support the diverse form-factor, interaction needs of such virtual objects, we aim to provide a modular, toolkit-based solution that enables easy, need-based construction of controllers that allow a range of manipulation interactions.

Our work falls within the domain of *passive haptics*, that involves the use of physical props as proxies of objects in the virtual world [15]. Proposed as a simpler alternative to active haptics [21], these techniques have been observed to significantly enhance perception of the virtual [16]. To address the diversity in shape of virtual objects, researchers have explored *reconfigurable props* that can be manually configured by the VR developer to mimic multiple objects [5, 23]. Although interesting, these techniques only support a narrow range of geometrically-similar virtual objects. Other efforts include the use of 2.5D systems, but are constrained by the complexity of the grounded hardware [27]. Moreover, most of these systems only support free, 6DoF manipulation on the props. In this scenario, our toolkit based approach to render objects of different shapes, manipulation interactions, can greatly enhance the range of passive haptics in VR.

We base our system on the LEGO®Duplo® platform. As a construction toolkit based on snap-together bricks, the LEGO offers infinite scalability, and is easy-to-use. We utilize unmodified LEGO bricks (*structure bricks*) to construct the skeleton of a virtual object, be it of any shape or size. To spatially map the physical proxy to the virtual object, we include our pose detection module that works in tandem with the HTC Vive base stations. Next, we include a set of modules to enable the object manipulation interactions discussed above. Specifically, these modules support rotation and linear translation interactions across a single axis — the other axes being physically constrained, preventing any motion. Finally, we also include modules to support button input and vibration feedback. Together, we define these custom-made modules as *feature bricks*. Also modeled as LEGO bricks, these feature bricks can be easily snapped into the construction of the physical prop. We also developed a VR Integration software package for the Unity® platform that can assist VR developers in integrating the functionalities provided by these feature bricks, while developing their applications. In summary, the contributions of this work are as follows.

- We present the design and technical implementation of VirtualBricks, a modular toolkit that enables easy, need-based construction of physical props for Virtual environments.
- We demonstrate the versatility of our toolkit through a rich set of applications including re-implementation of artifacts from recent research.
- We provide a VR Integration package that enables easy integration of VirtualBricks into applications developed using Unity.

All necessary material and documentation for implementing VirtualBricks can be found at:
https://github.com/weave-lab-iiitd/VirtualBricks

2 RELATED WORK

Our work falls within the category of designing haptics for Virtual Reality. We expand on, contribute to the knowledge of object manipulation (in VR) and borrow perspectives from the design of construction toolkits.

Haptics for Virtual Reality

In the past, considerable efforts have been made to tackle the need for a complete, immersive virtual experiences by adding haptics. These efforts can be broadly classified into active and passive haptics. Active haptic techniques exert forces onto the user to dynamically simulate virtual objects. These techniques can be divided into grounded [22, 25, 28] and ungrounded devices [7, 20, 31]. While grounded devices are fixed in the environment, ungrounded devices are hand-held or worn on the body of the user. A common disadvantage of these techniques is the complexity of their computational and mechanical models — sophisticated algorithms and hardware systems must be used to render the object properties such as weight, texture or temperature. Moreover, the complex hardware usually encumbers the user, restricting him/her into a small working space [33].

In contrast to active haptics, passive haptic approaches provide the user with physical props that act as proxies of the virtual objects. Early work in this direction was done by Hinckley et al. who used such props as handles for neurosurgical data visualization [15]. Since then researchers have

[1]https://www.vive.com/us/product/vive-virtual-reality-system/
[2]https://www.oculus.com/rift/

developed numerous techniques to harness this approach in VR [3, 6, 9, 10, 14, 32].

There exist two major challenges in the domain of passive haptics. First, to find a physical prop to match every virtual object. Recently, VR commodity hardware developers have launched props for virtual objects commonplace in popular VR games like gun [2] or tennis racket [1]. However, it is unrealistic to require a user to purchase a custom set of props for every virtual object. The second challenge is preparing the physical environment (including the props) w.r.t the changing virtual environment. For this problem, one major solution that has been explored is *encounter-type haptics* [24], wherein the user is presented with appropriate physical props to match the state in VR. Researchers have utilized actuation techniques including robotic arms [6], hand-held actuated wheel assemblies [30], drones [3] and human workers [10] to achieve this effect.

The first problem though remains relatively unsolved. There have been efforts to re-use the physical props to represent multiple virtual objects. Using techniques such as redirected walking [18] and portals [9], researchers have been able to redirect the user to the same physical prop — while exploring different virtual objects in VR. Another approach involves *reconfigurable props* that can be manually manipulated to match the shape of multiple virtual objects [5, 23]. Although interesting, both these approaches support a narrow range of very similar virtual objects. Hettiarachchi et al. have explored opportunistic use of everyday objects as proxies for simple shapes in the virtual world [14]. Siu et al. have explored the use of 2.5D shape display to dynamically render virtual objects [27]. However, the grounded system means that the object cannot be pickup up or manipulated in the 3D space. Finally, Zhao et al. have used assemblies of actuated, magnetically-attachable bricks to create different shapes [33]. Although the props are constructed in real-time, the complex assembly process can limit the geometries that can be achieved. Moreover, the props don't support object manipulations that involve single-axis motion. Addressing this first problem, in *VirtualBricks*, we propose a toolkit-based solution that enables easy, *need-based* construction of props.

Object Manipulation in Virtual Reality

Most systems utilizing passive haptics allow the user to manipulate the prop in six degrees of freedom — three degrees of linear translation and three of rotation [15, 33]. Similarly, this is a standard in the commodity VR controllers that utilize a 6DoF IMU and/or optical tracking to enable sensing. However, many of the objects that we encounter in everyday lives are not 6DoF manipulatable, but allow motion across specific axes — other axes being constrained by structure. For instance, a steering wheel allows rotation across its hinge but cannot be manipulated in any other manner. There is not much existing work to provide haptics for such objects and their corresponding interactions. Within the *VirtualBricks* toolkit, we include feature bricks that enable single-axis rotation and linear translation manipulations, hence allowing a realistic rendering of such objects in VR. Probably, the closest to our intention is the work of Strasnick et al. who have recently presented their techniques of adding physical constraints between two VR controllers to render such objects. Using their three versions of electro-mechanically actuated *Haptic Links*, they were able to implement two-handed object manipulations such as launching an arrow from a bow or playing a trombone (by linearly translating the *slide* across the *bell* section). Another related work is that of Chang et al. who developed a VR stimulation to let users manipulate tangibles across a single, linear dimension to solve puzzles, with the specific goal of developing spatial cognition within STEM [8]. In comparison, our toolkit based approach can enable haptics for a larger variety of such objects.

Construction Toolkits in HCI

Construction toolkits include simple parts that can be combined together to construct complex assemblies. Within HCI, construction toolkits have been used to satisfy the needs of do-it-yourself communities and for STEM education. Systems like SensorTape [11], ChainForm [26] and ActiveCube [29] have been proposed to enable users with different skill sets to rapidly prototype sensing applications and interactive tangibles. Similarly for STEM, researchers have developed educational toolkits like Cubelets[3] and MakerWear [17] to enable children to learn by creating artifacts for personal significance. Modularity, scalability and ease-of-use have been defined as important design features in all of these systems. In *VirtualBricks*, we present a toolkit-based approach to easily create passive haptics for VR — relying on the LEGO platform [12]. Recently, Ledo et al. have presented their work, defining suitable validation techniques for toolkit research [19]. In line with their recommendation, we use *demonstration* to evaluate our design, including a rich set of applications, re-implementation of artifacts from previous research.

3 USING VIRTUALBRICKS TO BUILD A VIRTUAL EXPERIENCE

In this section, we provide a high-level walk through of how a developer uses VirtualBricks to create a virtual experience. Jane is a developer who wants to design a VR application of driving a bike (Figure 2.D). She builds a physical proxy of the bike handle using a combination of feature and structure bricks (Figure 2.C). She uses two feature Bricks which allow rotation (for steering and throttle) and completes the physical

[3]https://www.modrobotics.com/cubelets/

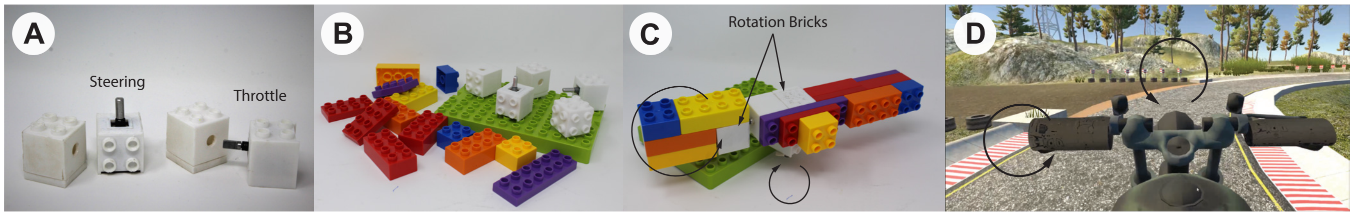


Figure 2: Building a virtual experience using VirtualBricks [A: Feature bricks, B: Structure Bricks, C: Completed physical proxy, D: VR application.]

form using structure bricks. Single-axis rotation interaction for steer and throttle imitate the functionality of a real bike. We show this process of building the handle's proxy in the accompanying video figure.

For the virtual end of her application, Jane first imports 3D model of the bike, including a manipulatable handle. She includes the VR Integration software package in the Unity environment and runs an *aggregator script* to receive rotation data from the feature bricks. She utilizes the *getAngle()* function included within the package to get rotation angles (in degrees) and maps them to the handle steer and throttle of her virtual bike. Jane can now ride the bike using this controller.

4 DESIGN OF FEATURE BRICKS

Feature bricks are custom designed LEGO bricks to enable pose detection, single-axis rotation, translation manipulation and other interactive functionalities for VR controllers. As seen in the previous section, these bricks can be easily added on to multiple points within the controller assembly. To allow this convenient integration, we designed these bricks to be self-sufficient, each containing a micro-controller, a wireless communication module and a battery to support its functionality. Next, we discuss the different functionalities, the design of the corresponding bricks in detail.

Pose Detection

Figure 3 shows a Pose Brick. It determines the absolute position, the orientation of custom-made controllers to map them with their virtual counterparts in real time. Positioning is achieved with the help of HTC Vive Lighthouse system, that emits laser beams to provide position reference. The brick utilizes three IR sensors arranged symmetrically on the top of its enclosure to sense these laser signals. With each sensor proving a field of view 120°, the three sensors together enable complete view of the surroundings. The sensor data is processed in a Teensy 3.1[4] microcontroller to triangulate the block's position in the space. This setup is based on an open source project[5]. Finally, the brick uses a 9-DoF (IMU) sensor to calculate the roll, pitch, and yaw, and hence determine its orientation.

Single-axis Rotation Manipulation

Single-axis rotation manipulation between two rigid objects can be observed in a doorknob or the steering wheel of a car. The objects can be rotated w.r.t to each other but cannot be manipulated in any other manner. We include two different kinds of rotation bricks to enable such manipulation in VR controllers.

[4]https://www.pjrc.com/teensy/teensy31.html
[5]DIY Position Tracking using HTC Vive's Lighthouse : https://github.com/ashtuchkin/vive-diy-position-sensor

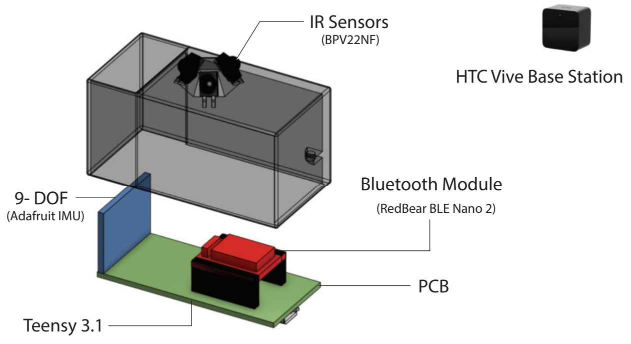


Figure 3: Pose brick

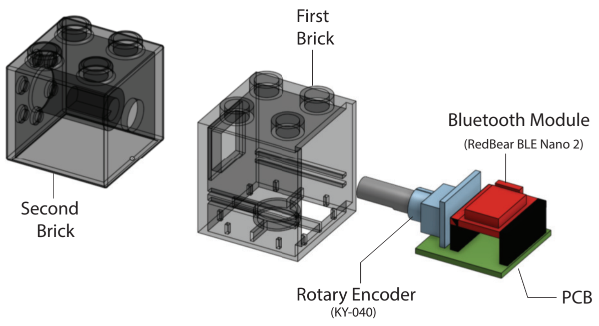


Figure 4: Rotation brick

Rotation Brick. Figure 4 shows a simple Rotation Brick. It enables rotation interaction between two LEGO bricks. The bricks can be rotated with respect to each other on a common axis. We use a rotary encoder for recording the above movement. These bricks are designed such that the first brick contains a window through which the rotating arm of the sensor protrudes; the second brick is attached to this part so that it can rotate w.r.t the first brick. Owing to the toolkit paradigm, this assembly (and hence the rotational axis) can be added to any point within the custom-made controllers.

Actuated Rotation Brick. Figure 5 shows an *Actuated* Rotation Brick. The brick facilitates rotational sensing and actuation using a servo motor. The user can still rotate the LEGO bricks w.r.t each other as in the previous case, but now they can also be actuated by the system as per the requirement of the virtual world. The potentiometer circuitry built into the servo records the degrees that it rotates when the user applies torque. In case of actuation, the servo motor moves according to the angle commands received over Bluetooth.

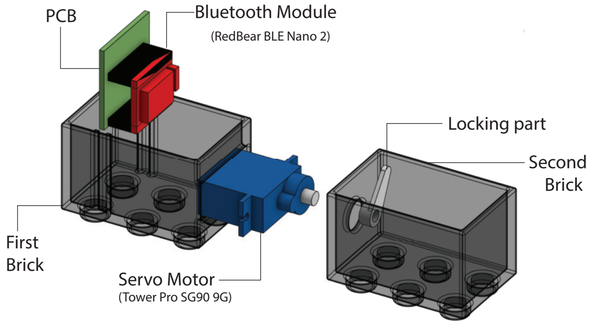


Figure 5: Actuated rotation brick

Single-axis Linear Translation Manipulation

Single-axis linear translation manipulation between two rigid objects is fairly common. A simple example is a sliding drawer mechanism — to open (or close) the drawer, one must translate it linearly w.r.t the frame that remains static. Next we describe two feature bricks that enable such interaction for VR controllers. While the first one allows free translation, the second one includes a retraction mechanism that pulls back the object as it is moved by the user. As in case of rotation, both these assemblies (and the corresponding translation axes) can be added to multiple points within the VR controller.

Proximity Brick. Figure 6 shows a Proximity Brick. It uses a proximity sensor to sense the linear translation interaction between two LEGO bricks. The main brick contains a window which exposes the proximity sensor towards the second brick

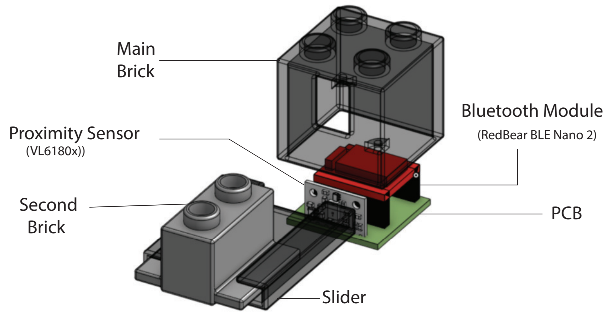


Figure 6: Proximity brick

that is movable on a linear slider (see figure). To measure the distance from the opaque surface of the movable brick, we use a LIDAR-based proximity sensor VL6180X[6] which calculates the distance using a time-of-flight based approach.

Retraction Brick. The Retraction Brick (Figure 7) adds retraction functionality to the linear translation. This main brick contains a mainspring attached to a rotary encoder. The second brick is attached to a thread that is wound over the spindle that contains the mainspring. As the second brick is pulled, the rotation motion of the spindle rotates the rotary encoder, contracting the mainspring. If the user releases the brick after pulling it to a certain distance, the mainspring returns to its original, relaxed state, pulling back the brick — hence retraction. This pull is also felt by the user when trying to move the brick, thus providing a increasing force haptic feedback, i.e. the pulling force increases as the object is pulled more. Continuing with the drawer example, this system makes an auto-closing drawer that closes once left by the user.

[6]https://www.st.com/en/imaging-and-photonics-solutions/vl6180x.html

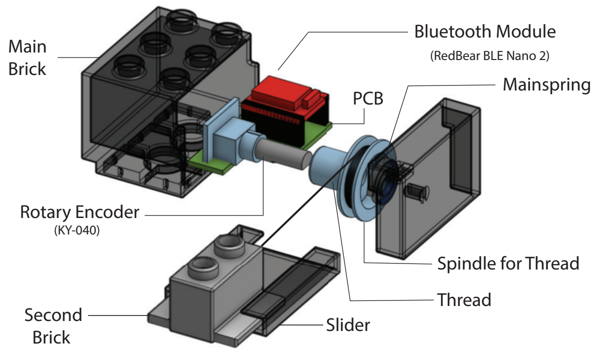


Figure 7: Retraction brick

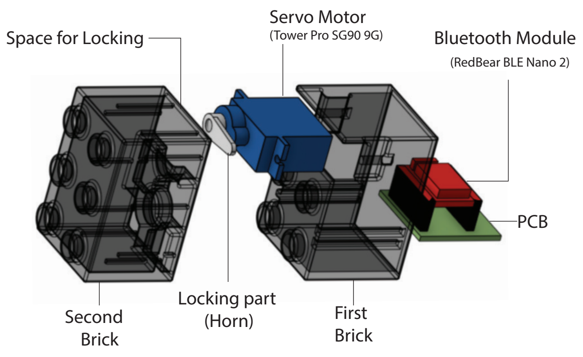


Figure 8: Lock brick

Locking Mechanism

The set of Lock Bricks (Figure 8) provide locking and unlocking functionality using a servo motor. When locked, the user will not be able to take the LEGO bricks apart; The horn of the servo is pressed against the base of the other block hence preventing separation. When unlocked, the horn turns and moves away from the base and hence the bricks can be separated like two normal LEGO bricks. This system can be used to create a prop assembly that can be separated into two different physical props (or controllers) as required in VR. Opening a bottle, separating its cap is a simple example.

Button Input and Vibration Feedback

Button input and vibration feedback are common entities in commodity VR controllers. Hence, we include two bricks to enable these functionalities for our toolkit. The Button Brick (Figure 9.A) contains a push button that protrudes out from the brick through a window. It can be used to trigger different events in an application like selection or shooting through a gun. Finally, the Vibration Brick (Figure 9.B) provides haptic feedback in the form of vibration. It contains a vibration motor attached to its wall.

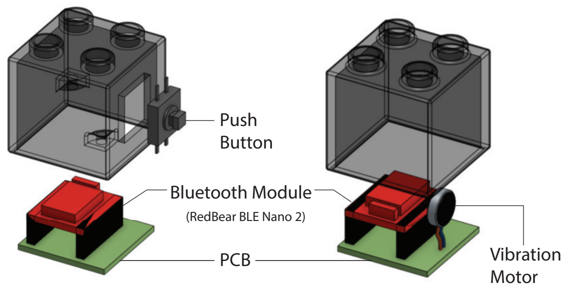


Figure 9: A: Button brick, B: Vibration brick

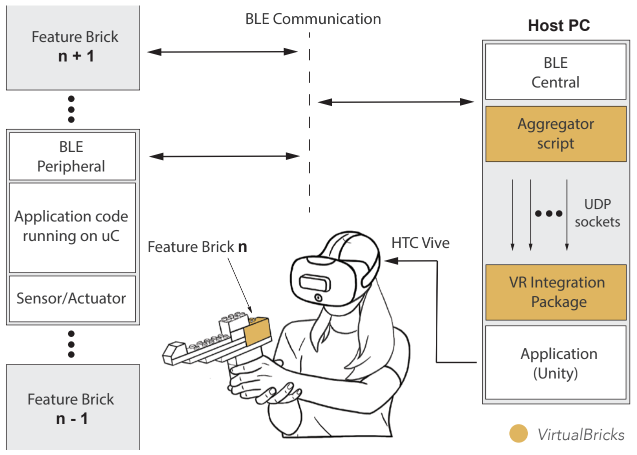


Figure 10: System Design

5 SYSTEM DESIGN

VirtualBricks enable easy, need-based construction of controllers for Virtual Reality. The toolkit is designed to work with the HTC Vive system controlled through Unity software running on a Windows 10 PC. The Host PC runs a Node.js based aggregator script to connect with all the feature bricks. This forms a star connection topology, a piconet wherein data is exchanged between the Computer - a BLECentral, and every feature brick - BLE Peripheral(s). BLECentral and BLE Peripheral functionalities are implemented using *Noble*[7], an open-source packages for cross-platform Bluetooth communication. The aggregator script transfers the received data to Unity via the User Datagram Protocol (UDP) protocol. The data is compiled in a JavaScript Object Notation (JSON) format and is pushed into the machine's UDP sockets. Figure 10 shows this communication design.

For Unity, we provide a custom VR Integration software package including a set of functions that enable easy integration of the received data into VR applications. There are functions specific to the functionality of each feature brick. For instance, for the Rotation brick, the developer can access the number of rotations, the angle of rotation and the speed of rotation through these functions. For the Lock brick, the developer can use the *lock()* and *unlock()* functions to lock and unlock the mechanism respectively. Figure 14 (added in the end) provides a complete list of functions that we developed.

6 APPLICATIONS

Using VirtualBricks, a developer can create a large variety of applications. We present a set of example applications to demonstrate the versatility of our toolkit. In each application,

[7]https://github.com/noble

we describe the user experience and the feature bricks used along with their function in the application.

First Person Shooting Game

This application allows the user to shoot at objects using a gun. The gun also has a scope attached with it that helps the user aim at zombies and glass bottles (Figure 11.A). To complete the physical proxy we first created the guns skeleton using LEGO bricks. Then we added a Button Brick to shoot. Finally, we added in a Pose Brick to map the prop with its virtual counter part. The functions used from the VR Integration package are *getAngle()* and *getOrientation()* for pose, and *getState()* to check regarding the button press event.

Slingshot

This application enables the user to shoot at the targets using a slingshot (Figure 11.B). The user pulls back the object, aims at the target and then leaves the object to shoot. We used a Retraction Brick to implement the pull mechanism and a Pose Brick for aiming at the targets. As the user pulls back the object, s/he feels an increasing force feedback (caused by contraction of the mainspring within the Retraction Brick) that imitates the stretch of the strings in the slingshot. The functions used from the VR Integration package are *getAngle()* and *getOrientation()* for pose, and *getDistance()* for retraction.

Fishing

This application allows the user to catch fishes from a pond using a fishing line (Figure 11.C). The user first elongates the fishing line to the appropriate length and then rotates the fishing reel to catch the fish. We used a Proximity Brick for elongation of the line and a Rotation Brick for the reel. Functions *getDistance()* and *getAngle()* are used to map the physical prop and the virtual fishing rod.

VRgoniometer

This application provides the user with a Goniometer (Figure 11.D), a device that helps to measure distance and angle between two points. The user can change the angle between two arms of the Goniometer and can alter their lengths to make the measurements. We used a Rotation Brick to alter the angle, and two Proximity Bricks for length adjustments. The functions used from the VR Integration package are *getAngle()* and *getDistance()*.

Disco

This application simulates a disco console (Figure 11.E). The user can adjust the lighting in the form of RGB values using linear sliders implemented using Proximity Bricks. S/he can also control the number of people dancing inside the disco using a knob made of a Rotation Brick. The functions used

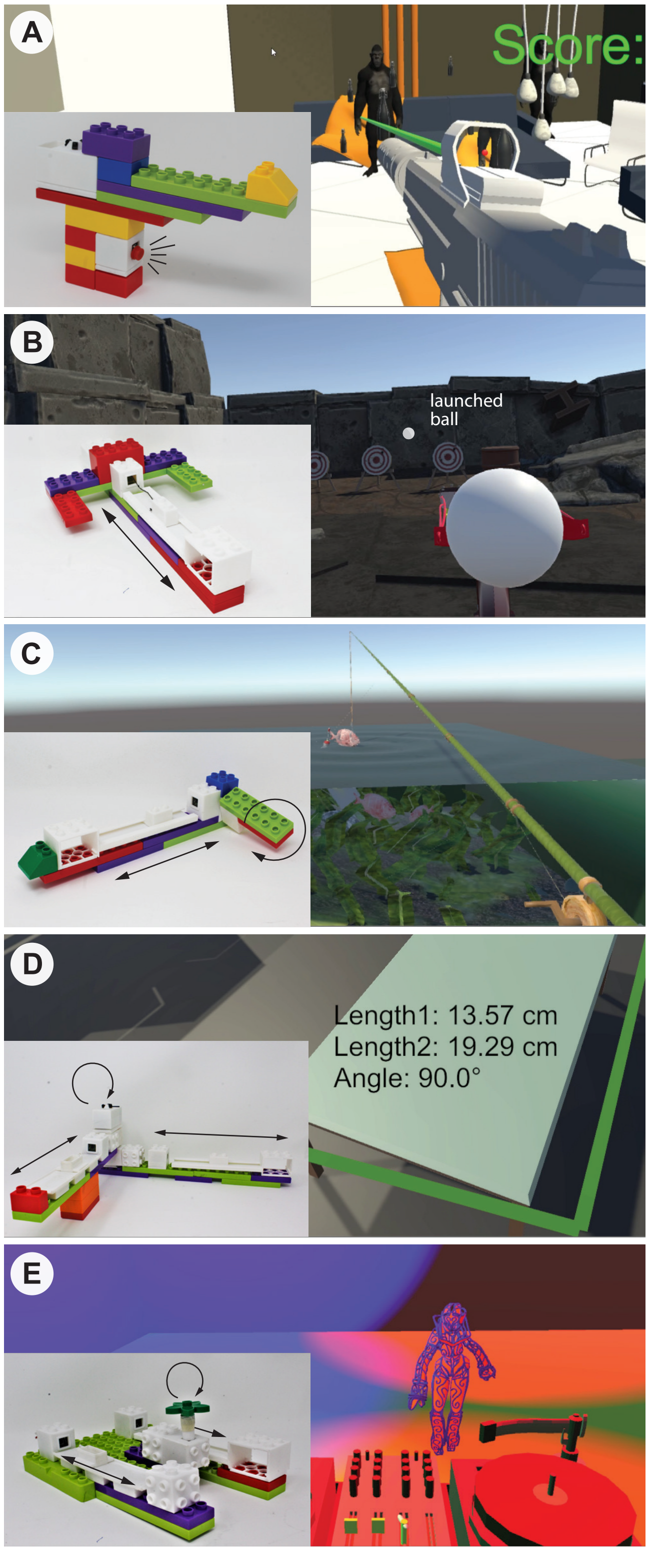


Figure 11: Applications [A: First Person Shooting Game, B: Slingshot, C: Fishing, D: VRgoniometer, E: Disco]; Inset images [Corresponding physical props]

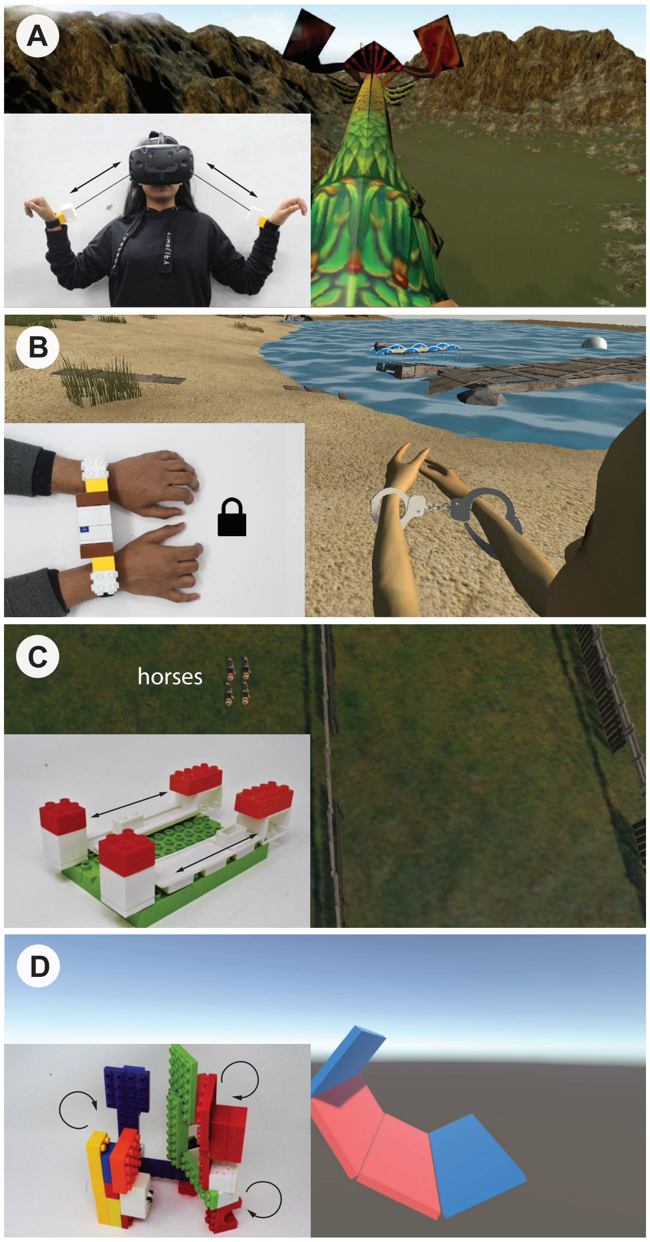


Figure 12: Applications [A: Dragon, B: Handcuffs]; Reimplementations [C: TASC, D: HaptoBend]

from the VR Integration package are *getAngle()* for rotation, and *getDistance()* for linear sliding.

Dragon

This application enables the user to perceive oneself as a dragon and fly around in the mountains (Figure 12.A). To fly higher, the user needs to flap his/her wings (hands) otherwise the dragon descends. The setup includes two Retraction Brick assemblies mounted between the user's head and his/her hands. The bricks are attached onto the body using velcro bands worn on the head, and on the hands. The force feedback provided by the retraction enhances the haptics of flying. The application uses *getDistance()* function to check flapping of the wings.

Handcuffs

This application enables the user to perceive handcuffs (Figure 12.B). The handcuffs are simulated using a Lock brick assembly mounted between the two hands of the user. When the bricks are locked, the user is unable to move his/her hands apart and feels the presence of handcuffs. Once the handcuffs are released (after solving an under-water puzzle), the bricks are unlocked and user is able to freely move his/her hands. The application uses *lock()* and *unlock()* functions to control the Lock Brick.

7 REIMPLEMENTATIONS

Recently, Ledo et al. have proposed *demonstration* as a way of validating toolkit research [19]. The applications that we have presented above fall in line with their idea of *novel examples*. Similarly, to go with their recommendation of *replicated examples*, we present systems from past research that we reimplemented using our VirtualBricks. We chose TASC [8] and HaptoBend [23] considering the diversity of their solutions.

TASC

TASC (Tangibles for Augmenting Spatial Cognition) [8] is a system which uses tangible objects to engage the spatial ability of the user through a series of visual-physical puzzles. Their final implementation includes a table with two long tangible bricks that can be moved along linear axes to control their virtual counterparts inside the game. We were able to easily construct this physical setup using our toolkit. We used two Proximity Bricks to implement the linearly manipulatable tangible bricks. We build their Gen2 puzzle: *Finding the Horse* as the VR application, where the user moves the bricks to control two long fences (Figure 12.C). The objective of the puzzle is to align the fences' openings to create a pathway for the horse. We used the *getDistance()* method from the VR Integration Package to sense the manipulation of the fences.

HaptoBend

HaptoBend [23] is an example of reconfigurable props that can be manually manipulated to match the shape of multiple objects in VR. Their prototype constitutes of four rigid sections with hinges, thus creating a bendable plane. The prop includes potentiometers for each hinge to sense the bend angles of the sections and a common IMU to sense the orientation of the controller. We used Rotation Bricks to act as the hinges, measure the hinge angles, and a Pose Brick for the orientation information. The corresponding VR application consists of a bendable plane which when folded to a cylindrical form, transforms into a torch (Figure 12.D). The functions used from the VR Integration Package are *getPosition()* and *getOrientation()* for pose, and *getAngle()* for the hinge angles.

8 DISCUSSION

In the previous sections, we have presented a rich set of applications to demonstrate the versatility of our toolkit. In this section, we classify those applications into a generalizable design space to access the broader implications of our work with respect to enabling haptics illusions for VR.

Custom-made Props to enable Passive Haptics in VR

We present a toolkit that enables a developer to construct custom props to imitate the shape and functionality of a variety of objects in VR. We classify these props into three distinct categories: *hand-held* props, *environment-mounted* props and *body augmentation*. The first four applications are examples of hand-held props. For these cases, addition of the Pose Brick maps the physical proxy to its virtual counterpart as the object is moved in free space by the user. The *Bike* and the *Disco* application fall in the second category wherein the prop is static and is mounted in the environment, like on a table or a wall. The Bike application is especially interesting as the bike handle is not static (as the bike is moving) but remains stationary w.r.t the user, who is riding the bike while sitting on a chair. Using hand-held props or commodity VR controllers (that can be moved freely in space) for such scenarios leads to unrealistic physical constraints and can reduce immersion. In our case, a LEGO baseplate[8] was first affixed on the table, and then the bike handle prop was constructed on top of it — hence providing realistic constraints as required by the bike's physical structure.

Within body augmentation, we explore the use of haptics enabled by VirtualBricks to add physical constraints to body movements — like relative motion between two hands, as in case of *Handcuffs*, or the hand flapping motion in the *Dragon* application. Though there have been few recent efforts in this direction (like the work of Achibet et al. who used an arm-mounted elastic armature to stimulate virtual objects [4]), overall this dimension presents interesting possibilities that need further exploration.

Enabling Scalable Object Manipulation in VR

Through VirtualBricks, we enable single-axis rotation and linear translation interaction for VR controllers. Following the construction toolkit approach, these manipulation axes can be added to multiple points within the controller. For instance, the *Disco* application uses three Proximity Bricks and a Rotation Brick to simulate linear sliders and knobs on a disco console. Moreover, to support the inherent diversity in such interactions, we include three different types of mechanisms, with each providing a distinct experience of manipulation. The basic Rotation Brick and the Proximity Brick enable *free* manipulation across the rotational or linear axis. The Retraction Brick provides a *force resistance* to manipulation and pulls back the object after it is left by the user. Finally, the Actuated Rotation Brick enables *dual manipulation* by the user and the system.

The first two approaches can be broadly classified on whether the prop returns to its original position, after being manipulated by the user. Although similar scenarios have been explored by other researchers (like in the work of Cheng et al. [9]), this distinction hasn't been explicitly observed as a design dimension. This distinction can lead to new interaction paradigms like by associating the returning physical prop to a different virtual object (like a new ball in the *Slingshot* application), that can then be manipulated using the same prop. Our third approach is relatively unexplored in VR. Although actuation has been explored for passive props in the past (like in *Dynamic Passive Haptics* to simulate different props [32]), dual manipulation of a prop by the user and the system is a new concept that we explore.

9 LIMITATIONS AND FUTURE WORK

The feature bricks are designed to be self-sufficient, each containing its own computation and communication circuitry, and a battery to support its functionality. Though this feature enables convenient integration of feature bricks into LEGO-based constructions, it also imposes some limitations on their operation that are important to discuss here. Firstly, since each feature brick forms a separate communication channel with the host PC, the maximum number of feature bricks that can operate at once are limited by the communication bandwidth of the PC [13], and the minimum data rate required for the application. Figure 13 shows the theoretical maximum number of feature bricks for different data rates. An application that requires all bricks to run at an update rate of 25 Hz can only utilize 15 bricks at once. Secondly, the batteries inside the bricks need to be removed and recharged after continuous use. As per the current consumption of different components, a 400 mAH battery (currently fitted into the feature bricks) can provide a continuous operation time of ~6 hours for a Proximity Brick (with the proximity sensor running continuously), and ~2.5 hours for a Lock Brick (with the servo moving between its lock and unlock positions twice a second).

[8]https://shop.lego.com/en-LT/product/Gray-Baseplate-10701

Other limitations of VirtualBricks come from the use of LEGO bricks. LEGO is a versatile toolkit comprising of snap-together bricks, and is greatly accessible and easy-to-use. However, the default LEGO connectors are unable to handle human-scale forces. To enable stronger joints, we modeled the CAD of feature bricks to have a tighter fitting, and they worked well, perhaps a bit too tight at times. In future, we do plan to explore more sophisticated designs to enable even better joints. Secondly, it must be noted that the LEGO platform cannot be used to construct all objects — specifically objects of spherical shape or the ones that don't have a rigid structure or ones with intricate designs.

For future work, we plan to create a software tool that assists the developer in constructing the physical proxies for different virtual objects. The input of this tool would be a CAD model of the object, on which the developer would then mark axes for rotation, linear translation manipulation or other desired interactions. Processing this input, the tool would suggest the required feature bricks and their appropriate arrangement to complete the prop. We also plan to explore feature bricks of different weight, texture, thermal signature, to further increase the diversity of controllers that can be made using VirtualBricks, and for better immersion.

10 CONCLUSION

While Virtual Reality is a platform for endless explorations in the digital world, its developer and user have been limited by one-size-fits-all proprietary controllers. With VirtualBricks, we attempt to empower the VR developer with a toolkit that enables the creation of custom controllers to match the different application scenarios. With our LEGO-based feature bricks that provide a range of functionalities, a VR Integration software package that facilitates easy integration in Unity, we open up the design space of interactive controllers that allow a range of manipulation interactions. We demonstrate the inherent modularity, and scalability of our design through a set of implemented applications. We hope that the creative community of VR designers and developers will adopt and extend this toolkit to their heart's desire.

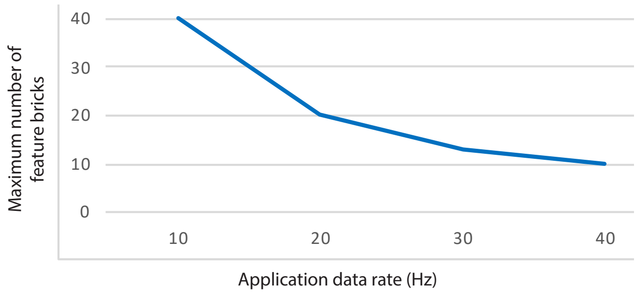
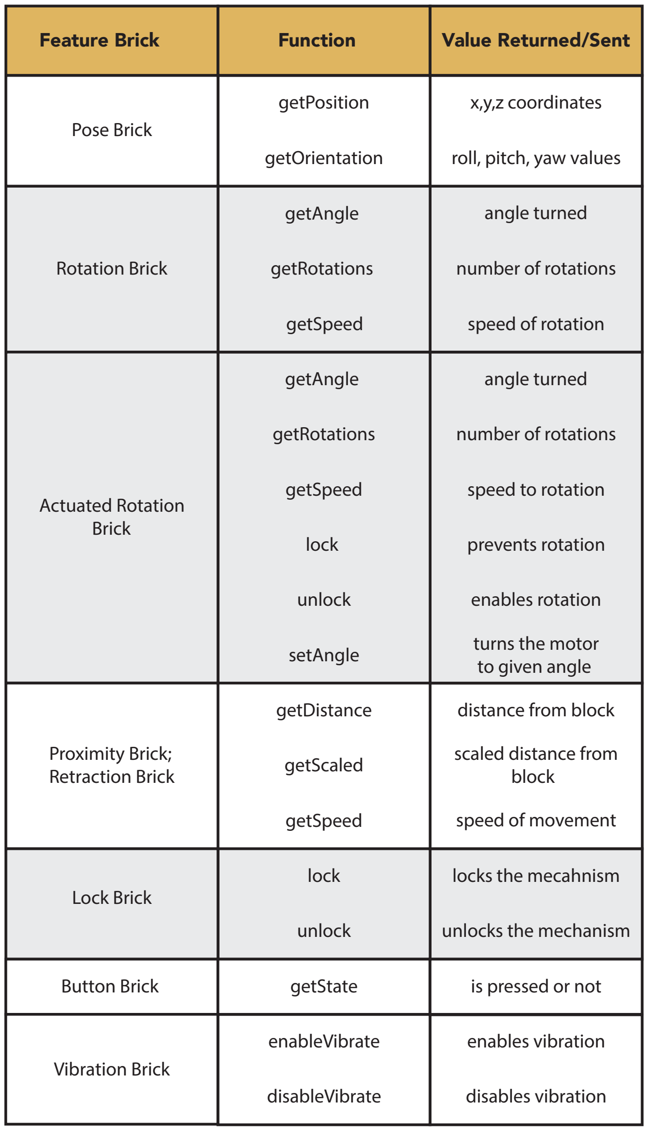


Figure 13: The graph shows the maximum number of feature bricks for different application data rates



Feature Brick	Function	Value Returned/Sent
Pose Brick	getPosition	x,y,z coordinates
	getOrientation	roll, pitch, yaw values
Rotation Brick	getAngle	angle turned
	getRotations	number of rotations
	getSpeed	speed of rotation
Actuated Rotation Brick	getAngle	angle turned
	getRotations	number of rotations
	getSpeed	speed to rotation
	lock	prevents rotation
	unlock	enables rotation
	setAngle	turns the motor to given angle
Proximity Brick; Retraction Brick	getDistance	distance from block
	getScaled	scaled distance from block
	getSpeed	speed of movement
Lock Brick	lock	locks the mecahnism
	unlock	unlocks the mechanism
Button Brick	getState	is pressed or not
Vibration Brick	enableVibrate	enables vibration
	disableVibrate	disables vibration

Figure 14: This table enlists the functions available for different Feature Bricks, the values that they return.

REFERENCES

[1] 2017. Racket Sports Set with VIVE Tracker. (2017). https://www.vive.com/us/VR-racket-sports-set-with-tracker/

[2] 2018. PlayStation VR aim controller. (2018). https://www.playstation.com/en-in/explore/accessories/playstation-vr-aim-controller/

[3] M. Abdullah, M. Kim, W. Hassan, Y. Kuroda, and S. Jeon. 2018. HapticDrone: An encountered-type kinesthetic haptic interface with controllable force feedback: Example of stiffness and weight rendering. In *2018 IEEE Haptics Symposium (HAPTICS)*. 334–339. DOI: http://dx.doi.org/10.1109/HAPTICS.2018.8357197

[4] M. Achibet, A. Girard, A. Talvas, M. Marchal, and A. Lécuyer. 2015. Elastic-Arm: Human-scale passive haptic feedback for augmenting interaction and perception in virtual environments. In *2015 IEEE Virtual Reality (VR)*. 63–68. DOI: http://dx.doi.org/10.1109/VR.2015.7223325

[5] Laurent Aguerreche, Thierry Duval, and Anatole Lécuyer. 2010. Reconfigurable tangible devices for 3D virtual object manipulation by single or multiple users. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*. ACM Press, 227. DOI: http://dx.doi.org/10.1145/1889863.1889913

[6] Bruno Araujo, Ricardo Jota, Varun Perumal, Jia Xian Yao, Karan Singh, and Daniel Wigdor. 2016. Snake Charmer: Physically Enabling Virtual Objects. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '16)*. ACM Press, 218–226. DOI: http://dx.doi.org/10.1145/2839462.2839484

[7] M. Bouzit, G. Burdea, G. Popescu, and R. Boian. 2002. The Rutgers Master II-ND force feedback glove. *IEEE/ASME Transactions on Mechatronics* 7, 2 (2002), 256–263. DOI: http://dx.doi.org/10.1109/TMECH.2002.1011262

[8] Jack Shen-Kuen Chang, Georgina Yeboah, Alison Doucette, Paul Clifton, Michael Nitsche, Timothy Welsh, and Ali Mazalek. 2017. TASC: Combining Virtual Reality with Tangible and Embodied Interactions to Support Spatial Cognition. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. ACM Press, 1239–1251. DOI: http://dx.doi.org/10.1145/3064663.3064675

[9] Lung-Pan Cheng, Li Chang, Sebastian Marwecki, and Patrick Baudisch. 2018. iTurk: Turning Passive Haptics into Active Haptics by Making Users Reconfigure Props in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM Press, 1–10. DOI: http://dx.doi.org/10.1145/3173574.3173663

[10] Lung-Pan Cheng, Thijs Roumen, Hannes Rantzsch, Sven Köhler, Patrick Schmidt, Robert Kovacs, Johannes Jasper, Jonas Kemper, and Patrick Baudisch. 2015. TurkDeck: Physical Virtual Reality Based on People. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. ACM Press, Daegu, Kyungpook, Republic of Korea, 417–426. DOI: http://dx.doi.org/10.1145/2807442.2807463

[11] Artem Dementyev, Hsin-Liu (Cindy) Kao, and Joseph A. Paradiso. 2015. SensorTape: Modular and Programmable 3D-Aware Dense Sensor Network on a Tape. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 649–658. DOI: http://dx.doi.org/10.1145/2807442.2807507

[12] David Gauntlett. 2014. The LEGO System as a tool for thinking, creativity, and changing the world. (2014), 16.

[13] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* 12, 9 (Sept. 2012), 11734–11753. DOI: http://dx.doi.org/10.3390/s120911734

[14] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM Press, 1957–1967. DOI: http://dx.doi.org/10.1145/2858036.2858134

[15] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. Passive Real-world Interface Props for Neurosurgical Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 452–458. DOI: http://dx.doi.org/10.1145/191666.191821

[16] Brent Edward Insko. 2001. *Passive Haptics Significantly Enhances Virtual Environments*. Ph.D. Dissertation. Advisor(s) Brooks,Jr., Frederick P. AAI3007820.

[17] Majeed Kazemitabaar, Jason McPeak, Alexander Jiao, Liang He, Thomas Outing, and Jon E. Froehlich. 2017. MakerWear: A Tangible Approach to Interactive Wearable Creation for Children. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 133–145. DOI: http://dx.doi.org/10.1145/3025453.3025887

[18] Luv Kohli, Eric Burns, Dorian Miller, and Henry Fuchs. 2005. Combining passive haptics with redirected walking. In *Proceedings of the 2005 International Conference on Augmented Tele-existence*. ACM Press, 253. DOI: http://dx.doi.org/10.1145/1152399.1152451

[19] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 36:1–36:17. DOI: http://dx.doi.org/10.1145/3173574.3173610

[20] Pedro Lopes, Sijing You, Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. 2017. Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1471–1482. DOI: http://dx.doi.org/10.1145/3025453.3025600

[21] Thomas H. Massie and J. K. Salisbury. 1994a. The PHANToM haptic interface: A device for probing virtual objects. In *Proceedings of the ASME Dynamic Systems and Control Division*. 295–301.

[22] Thomas H. Massie and J. K. Salisbury. 1994b. The PHANToM haptic interface: A device for probing virtual objects. (1994), 295–301.

[23] John C. McClelland, Robert J. Teather, and Audrey Girouard. 2017. Haptobend: shape-changing passive haptic feedback in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction*. ACM Press, 82–90. DOI: http://dx.doi.org/10.1145/3131277.3132179

[24] W. A. McNeely. 1993. Robotic graphics: a new approach to force feedback for virtual reality. In *Proceedings of IEEE Virtual Reality Annual International Symposium*. 336–341. DOI: http://dx.doi.org/10.1109/VRAIS.1993.380761

[25] Jun Murayama, Laroussi Bougrila, Yanlinluo Katsuhito Akahane, Shoichi Hasegawa, Béat Hirsbrunner, and Makoto Sato. 2004. SPIDAR G & G: A two-handed haptic interface for bimanual VR interaction. In *Proceedings of EuroHaptics 2004*. 138–146.

[26] Ken Nakagaki, Artem Dementyev, Sean Follmer, Joseph A. Paradiso, and Hiroshi Ishii. 2016. ChainFORM: A Linear Integrated Modular Hardware System for Shape Changing Interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 87–96. DOI: http://dx.doi.org/10.1145/2984511.2984587

[27] Alexa F. Siu, Eric J. Gonzalez, Shenli Yuan, Jason Ginsberg, Allen Zhao, and Sean Follmer. 2017. shapeShift: A Mobile Tabletop Shape Display for Tangible and Haptic Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM Press, 77–79. DOI: http://dx.doi.org/10.1145/3131785.3131792

[28] Haruhisa Kawasaki Takahiro Endo and Yasutoshi Doi Tetsuya Mouri. Five-fingered haptic interface robot: HIRO III. In *World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. https://ieeexplore.ieee.org/document/4810812/

[29] Ryoichi Watanabe, Yuichi Itoh, Masatsugu Asai, Yoshifumi Kitamura, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi. 2004. The Soul of ActiveCube: Implementing a Flexible, Multimodal, Three-dimensional Spatial Tangible Interface. *Comput. Entertain.* 2, 4 (Oct. 2004), 15–15. DOI:http://dx.doi.org/10.1145/1037851.1037874

[30] Eric Whitmire, Hrvoje Benko, Christian Holz, Eyal Ofek, and Mike Sinclair. 2018. Haptic Revolver: Touch, Shear, Texture, and Shape Rendering on a Reconfigurable Virtual Reality Controller. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18).* ACM Press, 1–12. DOI:http://dx.doi.org/10.1145/3173574.3173660

[31] Nobuhisa Hanamitsu Yukari Konishi, Ayahiko Sato Kouta Minamizawa, and Benjamin Outram Tetsuya Mizuguchi. 2011. Synesthesia Suit : the full body immersive experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).* https://dl.acm.org/citation.cfm?id=2932629

[32] Andre Zenner and Antonio Kruger. 2017. Shifty: A Weight-Shifting Dynamic Passive Haptic Proxy to Enhance Object Perception in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 23, 4 (April 2017), 1285–1294. DOI:http://dx.doi.org/10.1109/TVCG.2017.2656978

[33] Yiwei Zhao, Lawrence H. Kim, Ye Wang, Mathieu Le Goc, and Sean Follmer. 2017. Robotic Assembly of Haptic Proxy Objects for Tangible Interaction and Virtual Reality. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17).* ACM, New York, NY, USA, 82–91. DOI:http://dx.doi.org/10.1145/3132272.3134143