TwistBlocks: Pluggable and Twistable Modular TUI for Armature Interaction in 3D Design

Meng Wang[1,2], Kehua Lei[1], Zhichun Li[3], Haipeng Mi[1], Yingqing Xu[1,2]
[1]Academy of Arts and Design, [2]The lab for Lifelong Learning, [3]Department of Automation
Tsinghua University, Beijing, China
{m-wang16, lkh16, zc-li15}@mails.tsinghua.edu.cn, {mhp, yqxu}@tsinghua.edu.cn

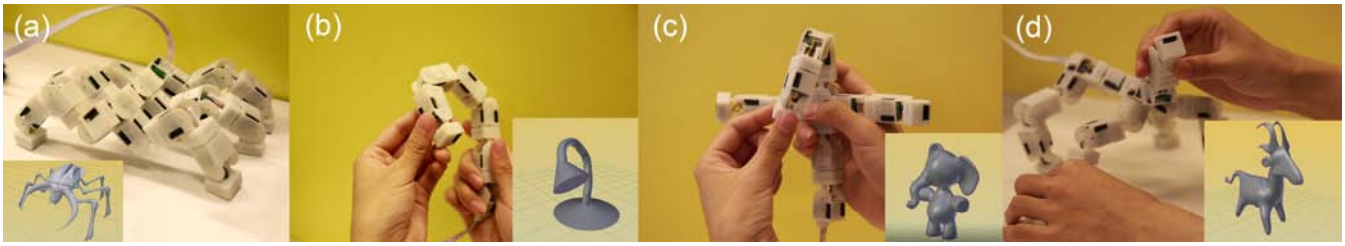

Figure 1. TwistBlocks as physical armatures.

ABSTRACT
The use of armatures is a convenient way of deforming and animating 3D digital models. However, interact with an armature is usually time-consuming, and often requires professional skills. Tangible interfaces, such as building blocks, while having improved the accessibility of digital construction, are still lacking in flexibility and present difficulties in dealing with curved armatures. This paper introduces TwistBlocks, a pluggable and twistable modular TUI that improves the accessibility of 3D modeling and animating by physical armature interaction. TwistBlocks is capable of creating complex armatures with dense branches, and supports a high DOF (Degree of Freedom) in physical manipulation. In addition, a set of software tools are provided for novice users to easily create, rig, and animate models. The global-posture sensing network sensing scheme can also measure the rotation and movement of the physical armature, and enables interaction between multiple models.

Author Keywords
Tangible Input; 3D Design; Skeletal Animation; Non-Rigid Interface; Modular Sensor Network

ACM Classification Keywords
• Human-centered computing~Interface design prototyping

TEI '18, March 18–21, 2018, Stockholm, Sweden

ACM ISBN 978-1-4503-5568-1/18/03…$15.00
https://doi.org/10.1145/3173225.3173231

INTRODUCTION
Creating and rigging digital 3D models is a common task in areas such as entertainment, education, and digital fabrication. However, novice users usually find doing this task challenging since it requires lots of experience, professional skill, patience, and time.

Tangible User Interfaces (TUIs) enable direct, hands-on interaction with physical objects. Tangible building blocks, which can be assembled into a complex structure, have been explored in creating both physical and digital models. Generally, 3D design often requires a high flexibility, which may not be easily supported by the traditional rigid TUIs. For example, the armature (aka skeleton) is widely used to provide deformation for a digital model. However, because it consists of many bones with a branched structure, it is not easy to use simple TUIs to control an armature. Recently, modular TUIs have been proposed to deal with the armature manipulation problem [1, 12]. Those methods adopt hierarchical local sensing techniques, limit such methods to have a wider applications.

In this paper, we present TwistBlocks, a novel modular TUI toolkit that improves the accessibility of 3D modeling and animating by physical armature interaction. Our contributions are summarized as follows:

- A pluggable and twistable modular TUI with a compact design that can be assembled into various complex physical armatures that can then be manipulated freely.
- A novel sensing scheme using a global-posture sensing network, which enables measurement of rotation and movement in a global coordinate system and the interaction between multiple armatures.
- A series of software tools that allows novice users to easily rig existing models or to create new models based on a physical armature.

RELATED WORK

Building Virtual Structures with Physical Blocks

Assembly based TUIs have been a hot topic in tangible interface research for a long time. Anderson et al. introduced a method to build virtual structures with LegoTM-like blocks [5] as early as 1999. Kitamura et al. presented ActiveCube [13], which allows for real-time 3D interaction with tangible cubes. Gupta et al. demonstrated the DuploTrack system [2] to track the assembly process of building blocks using computer vision.

Most assembly based TUIs employ rigid building blocks, which are capable for constructing rigid structures but lack flexibility to present curved shapes.

Curve Input

Curve input has been the focus of several research works. Grossman et al. introduced an input device for the manipulation of curves [14]. Rendl et al. presented FlexSense [4], a transparent self-sensing deformable surface. Bächer et al. introduced DefSense [11], an algorithm for the design and fabrication of customized, deformable input devices that are capable of continuously sensing their deformation. Nakagaki et al. demonstrated LineFORM [9], an actuated curve interface.

Sensing techniques have enabled new input methods supporting curved shape interaction in a tangible manner. However, most of them can only interact with a single curve string or an individual deformable surface.

Modular Sensing

Kahn et al. propose the concept of "Smart Dust" for mobile networking [8] in 1999, which is a prototype of modular sensing. Modular sensing is a novel way in HCI to perform a complicated sensing. Topobo [7], presents a 3D constructive assembly system embedded with kinetic memory. Siftables [6], applies wireless sensor networks to tangible user interfaces. SensorTape [3], presents a self-sensing tape and a novel way of detecting/obtaining curve input.

In these modular sensing networks, the sensor nodes are mostly in a static topology. A sensing network with dynamic topology for assembling requires more than one physical network layer in the design.

Armature Interaction

An armature is a very convenient tool for rigging a digital model. Weller et al. presents a computationally-enhanced hub-and-strut construction kit which can generate a simple digital character by physical construction for learning and play [10]. Yoshizaki et al. presented an actuated handheld puppet system for controlling the posture of a virtual character [15] in 2011. Jacobson and Glauser et al. developed a tangible input device as an armature to rig an animation [1, 12]. Their work applies a hierarchical local sensing scheme, which provides good performance in interacting with simple armatures. However, such method will encounter problems while handling a complex armature, and a local sensing system is not applicable for different characters in the same scene.

Based on previous work, we have extended the modular TUIs design for armature interactions in various aspects such as sensing, branching, rigging and modeling.

DESIGN

Considering the limitations of some related work, the following design requirements have been considered during the design process:

- **Global manipulation**. Global sensing enables physical rotation of the whole model, multiple models interaction and presents zero accumulated error in calculation.
- **Compact branching**. The modular block should serve as a polydirectional splitter itself. A compact design can better match more models, especially some with dense branches like the bug in Figure 1.a.
- **DOF of twists**. A digital armature joint has 3 DOFs for twisting. A 3-DOF physical joint (such as a ball and socket joint) can provide a better user experience for manipulation input.

Global Sensing System

The TwistBlocks sensing system consists of a set of assembling blocks and a data collector. The system architecture is shown in Figure 2.

Each block contains a 9-axis IMU (Inertial Measurement Unit), enabling them to measure the rotation in the global coordinate system. Compared with the relative rotation measuring techniques, the armature calculated using a global coordinate data can accurately reflect not only relative bending and twisting of each joint, but also a global manipulation in advance, enabling rotating and movement in one's hand. Additionally, a global coordinate sensing also removes the accumulated error that is always presented in the relative sensing systems.

The blocks, assembled together, collectively become an IMU sensing network. This sensing network consists of two physical layers: an IIC bus for data transmission, and a network between blocks for topology detection and reconstruction.

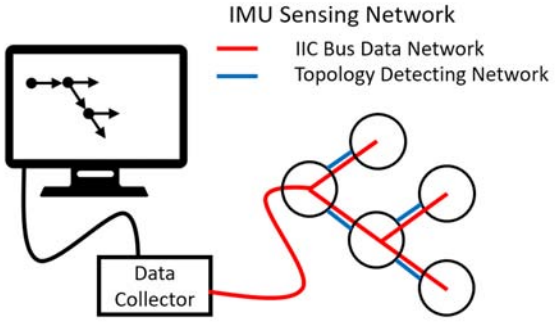

Figure 2. System architecture.

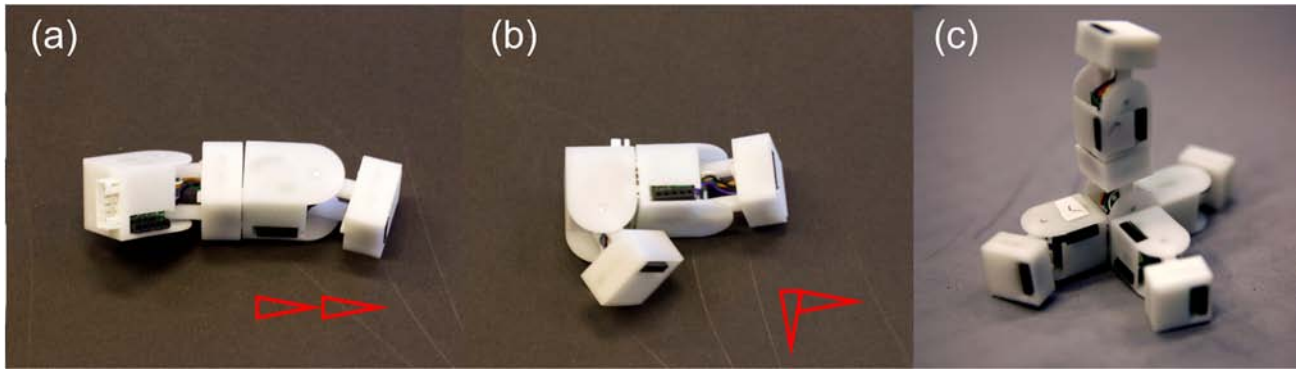

Figure 3. (a) A bottom connector plugged into a top connector, forming a chain connection; (b) A bottom connector plugged into a side connector, forming a side connection; (c) An example armature consists of one chain and three branches.

A data collector is used to consolidate the data and forward them to the computer. Sensors and the data collector simply composes a star-type communication network. Multiple data collectors can work in parallel, so that different sensing networks can work together to support multiple armatures.

Blocks

The design of TwistBlocks is based on building blocks, which has good accessibility for physical construction. Each block contains a twistable articulation and several connectors.

Figure 3 shows how the connectors work. There are three types of connectors in each block module. The *bottom connector* (white) can either match a *top connector* (black) to create a chain connection, or match a *side connector* (black) to create a branch connection. Each block module can have up to four side connectors, allowing other blocks to be connected to it simultaneously. In the latest design we replace the regular PCB connectors with a set of modular battery connectors (Figure 4.Bottom), which provides more robust electronic connection.

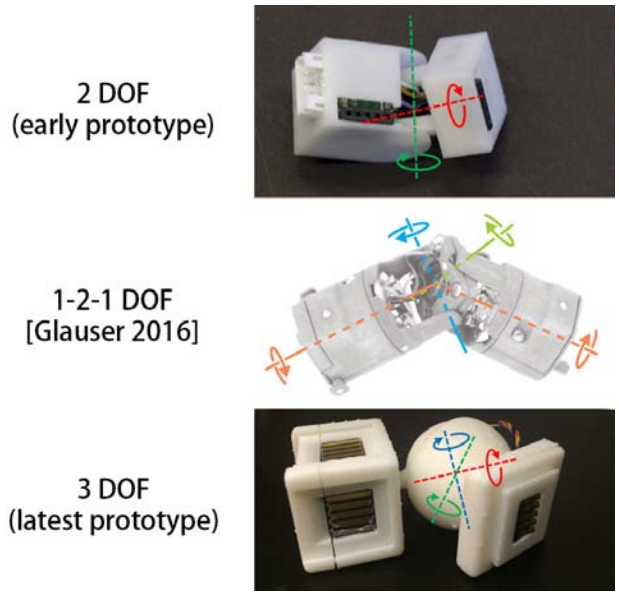

Figure 4. The articulation design of different DOF.

From 2-DOF to 3-DOF

In the first mockup of a modular block, we applied a gimbal structure that enabled two-DOF of rotation in every sensor block (Figure 4.Top). Glauser et al. presented a structure to to increase the DOF by adding two individual rotation DOFs at the end of each node (Figure 4.Middle) [12]. We then improved the joint design of modular blocks by using ball-socket joints (Figure 4.Bottom). This unique 3-DOF joint design can be implemented as the IMU sensing scheme does not need any sensing part within the articulation. With integrated 3-DOFs in one articulation, the ball-socket joints can provide better experiences than the separated 1-2-1 DOF structure.

Branching

An armature is structured as a typical tree network in topology. Each node with two or more child nodes will produce a physical branch in the network. In order to compose such a branch, a specially-designed splitter is usually required. Splitters lead to a greater *rigid area* and longer distances between branches (Figure 5). Having considered these limitations, we designed our modular blocks to be directly pluggable into the sides of one another. This splitter-free design allows for more flexibility in the physical armatures.

As is shown in Figure 5, each branch will inevitably produce a *rigid area*, in which the physical component is rigid and cannot be deformed. In order to achieve the splitter-free design, we integrated the rotation parts and branching connection parts into one block. The small size of the block is also important for hands-on interaction.

With this compact design scheme, we shrank the rigid area and reduced the minimum distance between branches without increasing the size of certain blocks. This improvement makes TwistBlocks more compact and flexible, allowing the built armatures to better match with the desired armatures, especially those with dense branches, such as insects. (Figure 1.a.)

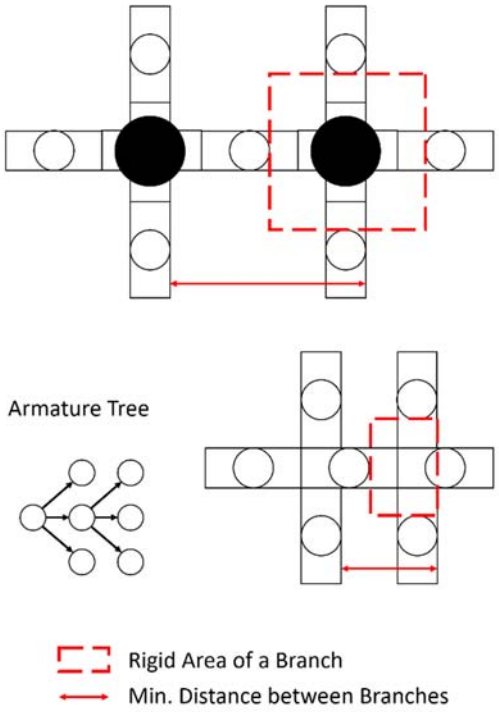

Figure 5. When matching the same armature tree, our design (bottom) shrinks the rigid area and reduces the minimum distance between branches in comparison with a splitter design (top). The two designs have the same block sizes in this figure.

Software

The data collector consolidates the data and forwards them to the computer. We used *Blender*, an open-sourced software, for creating and rigging 3D models. An add-on for *Blender* was developed, which allows the software to receive the sensing data while still maintaining compatibility with *Blender's* functions so professional users can still perform elaborate modifications on their models.

Apart from the generation of a digital armature based on the physical one, the add-on also provides many easy-to-use features, such as the relative deformation tool and the armature-based modeling tool. A detailed description is made in the following application section.

IMPLEMENTATION

Modular Block Hardware

A modular block consists of a controlling circuit board and a few connectors. Connectors are divided into two groups: chain connectors and side connectors. Each block has two chain connectors (one parent and one child), and a few side connectors (all children). Only a parent connector and a child connector can be connected. A compact design was implemented for the modular block resulting in a size of 18mm×18mm×36mm.

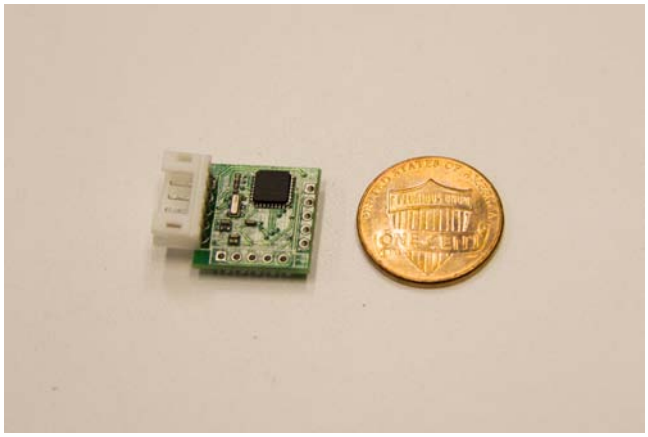

Figure 6. The PCB design.

The controlling circuit has a size of 15mm×15mm (Figure 6). An MCU (Micro Controller Unit, ATMEGA328P) and a 9-axis IMU (Inertial Measurement Unit, MPU9150) were integrated on the board. The IMU and the MCU are connected using a software IIC bus. The raw data from the IMU (including gyroscope, accelerometer, and magnetometer) is processed using an onboard DMP (Digital Motion Processor). Calibration is automatically done when the firmware runs for the first time. We fuse the gyroscope and compass data so that the sensor won't produce any obvious yaw drift (within 0.1°) even used continuously for hours. The AHRS (Attitude and Heading Reference System) data of each node is represented by a quaternion.

Due to the fact that the fusion calculation outputs a 3D rotation and that the accuracy/resolution differs when facing different directions, it is difficult to make a precise measurement directly. The compass sensitivity 0.3uT/LSB is roughly 0.3°, and the accelerometer sensitivity 16384LSB/g is roughly 0.003°. This basically provides similar accuracy as [1, 12]. However, in a local sensing system like [1, 12] the 0.5° angular error could produce a 5°angular accumulated error with a 10-node chain. In our global sensing system the accumulated error is always zero.

Topology Detection

In the TwistBlocks system, we defined two types of child branches in the topological network: chain and side. The topological network of an armature can be presented as a *tree network with chains* (Figure 7). Four parameters are used to locate each node: (M, S, C, N). M is a unique value that denotes the chain of the node. S denotes the index of the node in the chain. These two values describe the connection of each chain. If a chain's master node (S=0) is connected to another node (i.e. the start of a branch), C and N are used to record the value of M and S of the connected node. This describes the connection between chains so that the entire topology can be decoded and obtained from the original node data. Table 1 shows the specific description of the parameters.

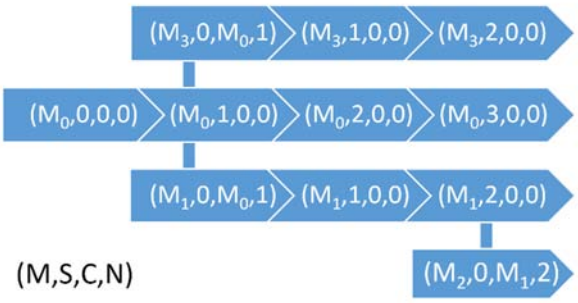

Figure 7. A typical chain tree network.

	Meaning	Value
M	Chain ID	The ID of the master node
S	Slave Index	Position in the chain
C	Connected Chain	M of the connected node
N	Connected Node	S of the connected node

Table 1. Description of the parameters.

A unique ID is burned into the firmware of each node which allows for the recognition in topological network and the avoidance of IIC address conflict. Unidirectional serial connections are used between nodes: the bottom connector contains a RX line, meanwhile the side and top connectors contain two different TX lines. Thus, a node can differentiate whether it is plugged into a side connector or plugged into a top connector. The node is able to interpret the topological parameters in the data received to know which node it is connected to.

Network Data Gathering

The connector between nodes integrates power supply, a hardware IIC bus, and a unidirectional serial interface described above. All the nodes act as IIC master devices and send data packets to a data collector which has a connection to a PC. We coded the data package as 15Bytes, so that a standard 400KHz IIC bus can in theory transmit 3000package/s. Therefore our data rate can reach 3000Hz for single node, or 300Hz for 10 nodes. In the current demo we control it at 20Hz to provide possibility for more nodes to work together at an acceptable rate.

Because we apply a multi-master IIC bus, the maximum of sensors is not restricted by the limitation of 7-bit address. However the actual number is affected by the parasitic capacitance of each device. We support at least 30 blocks under current circuit design, which is enough for most scenarios.

Armature Structure Generation

A data packet contains four topological parameters and a matching quaternion. The software will store or update the data once it receives a data packet from the data collector.

Calculations would be performed to ensure that every parent node is generated before its child node each time a new armature is generated. The armature bones can then be rebuilt one by one. Firstly, the parent bone is found using the topological relationship. Secondly, a new bone is extruded in accordance with the physical length set in the software. Thirdly, the new bone is rotated according to the AHRS data. This process is repeated until the whole armature is reconstructed.

APPLICATION

Armature Generation

Armature generation is the most fundamental function provided by the interface. The user starts by attaching the blocks together and twisting the joints to create a desired armature (Figure 8.a). Once the physical armature is created, the user can instruct the software to collect the sensing data of the armature and a digital *initial armature* will be generated automatically (Figure 8.b). This armature has the same structure, length and posture as the physical, tangible one. After the generation process, the user can still physically change the tangible armature (e.g. adding or removing a block) and the changes can be updated to the digital armature with another generation command.

Rigging Existing Models

Once the *initial armature* is generated, its posture can be changed according to the manipulations on the physical one. If an existing model is rigged with this *initial armature*, it will perform a deformation together with the armature (Figure 8.d-e).

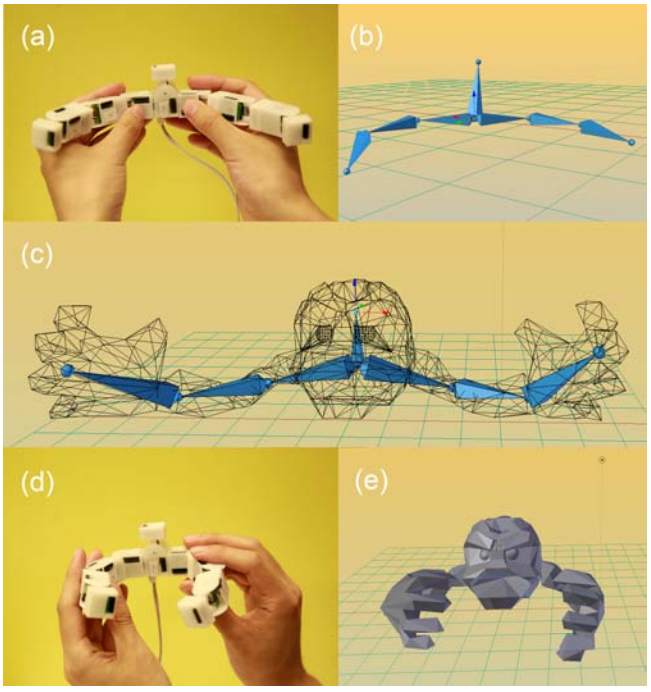

Figure 8. (a) Building a physical armature with TwistBlocks. (b) Generating a coressponding initial armature. (c) Adjusting the bones digitally to rig the model. (d)(e) Performing a relative deformation using physical manipulation.

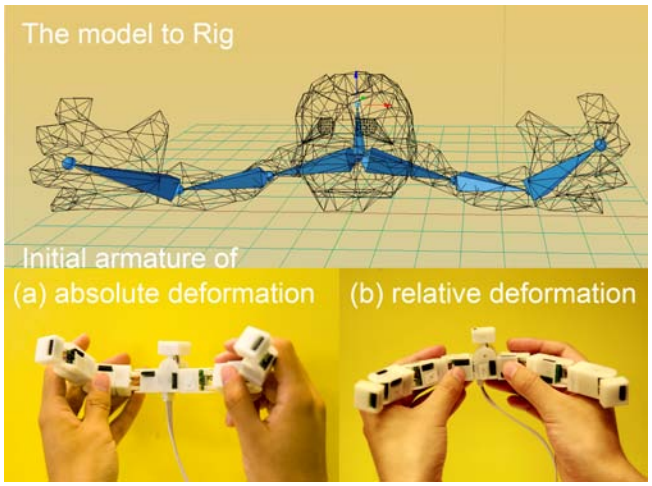

Figure 9. An initial armature with corresponding posture is required before absolute deformation. In contrast, relative deformation only needs an approximate structure.

TwistBlocks supports two types of deformations: *absolute deformation* and *relative deformation*.

Absolute deformation. With absolute deformation, the digital armature is always the same with the physical one. It provides direct imitation of the physical armature in the digital world.

Absolute deformation is widely used in most related work. However, it can still be improved if we pursue a better user experience. Physical manipulation is usually natural but not precise. Before the absolute deformation process, the user has to physically adjust the *initial armature* to a proper posture, and bind the armature to the existing model (Figure 9.a). Physical manipulation is not the best way to deal with this task, and if the user desires an armature with various bone lengths, digital modification is preferable. However, directly modifying the digital armature may lead to mismatches under absolute deformation. To solve this problem, we introduced relative deformation.

Relative deformation. With relative deformation enabled, the bones of the *initial armature* can be edited in the software (e.g. adjusting the length of certain bones), while still allowing the user to physically manipulate the modified armature with a corresponding mapping calculated by the change of physical armature posture.

Although absolute deformation is more precise, we noticed that users do not care about whether the digital armature is exactly the same as the physical one or not. Relative deformation provides faster and more flexible interaction for digital modification, and users will not feel an apparent mismatch when manipulating the model.

3D Design and Viewing

In our pilot study some novice users felt that they preferred animation rather than modeling, but it becomes an obstacle as models must be created before animations can be done on them using armatures. This indicates that an easy-to-use armature-based modeling method would be important and useful to novice users. Thus, we developed an armature-based fast modeling method consisting of the following procedures.

Initial armature. The user creates a physical armature with TwistBlocks and then an *initial armature* is generated. (Figure 10.a)

Straighten armature. Bones in the any chain can be automatically straightened (Figure 10.b), so that standard meshes (e.g. cylinders) can be easily applied to the armature.

Modeling. A function can help to generate a chosen type of mesh subdivided (producing a smooth effect) and rig it with the chosen chain (including length and position) automatically (Figure 10.c). The user can also add meshes by themselves.

Animating. All the models created can deform with the tangible armature automatically (Figure 10.d).

The TwistBlocks also allows for the design of various curved models (Figure 11). The global coordinate sensing network allows users to rotate the entire armature in their hands and view the digital model from multiple directions. A 3D curve can be better observed and adjusted in the physical space.

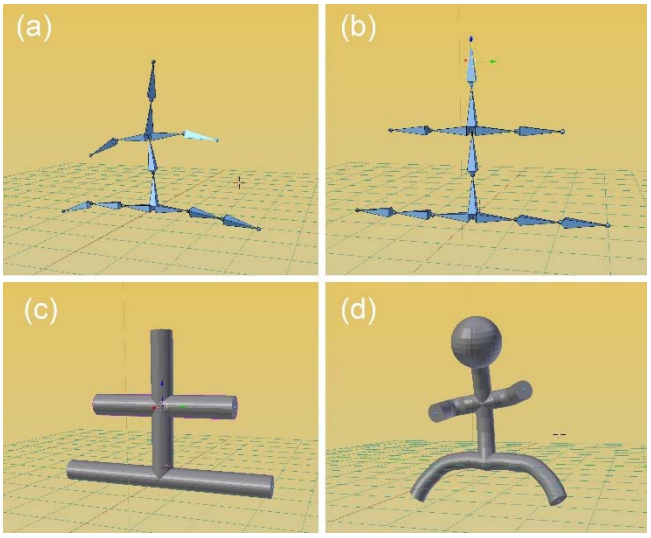

Figure 10. (a) Building with the blocks and generating an initial armature. (b) Straightening the armature. (c) Adding the cylinders automatically. (d) Adding a sphere and enabling deformation.

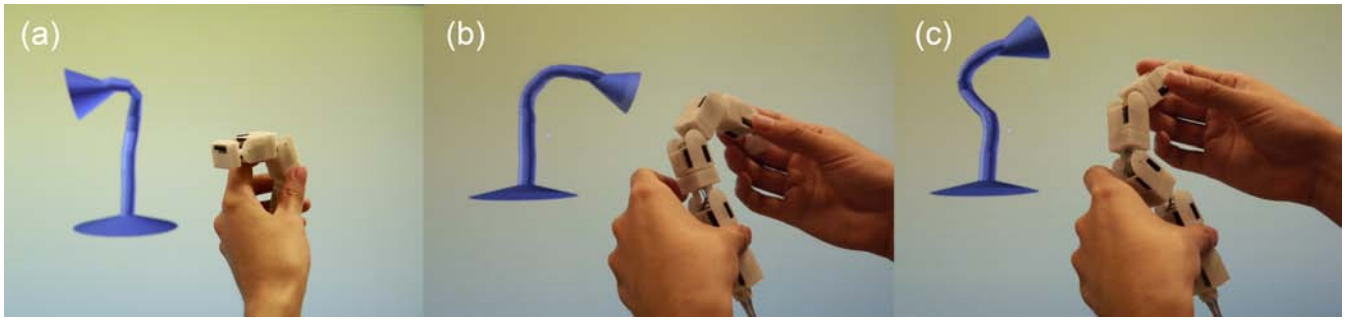

Figure 11. (a) Designing a lamp based on the physical armature. (b) Rotating the armature in order to observe the digital model from different directions. (c) Adjusting the lampstand using physical manipulation.

Interaction between Multiple Models

Interaction between multiple models brings possibilities to the creation of complex dynamic scenes such as interactive storytelling. The TwistBlocks system supports multiple physical armatures to be used simultaneously, enabling interactions between multiple digital models in real-time (Figure 12).

Since TwistBlocks uses a global coordinate system, all armatures share the same coordinate space. This means that users can easily perform face-to-face or back-to-back postures with two or more digital models. Traditional way of using a hierarchy of local sensors is not capable to sense any interaction between different models, while our sensing scheme enables the detection of global rotation, as well as global movement if we further apply a camera to calibrate the position.

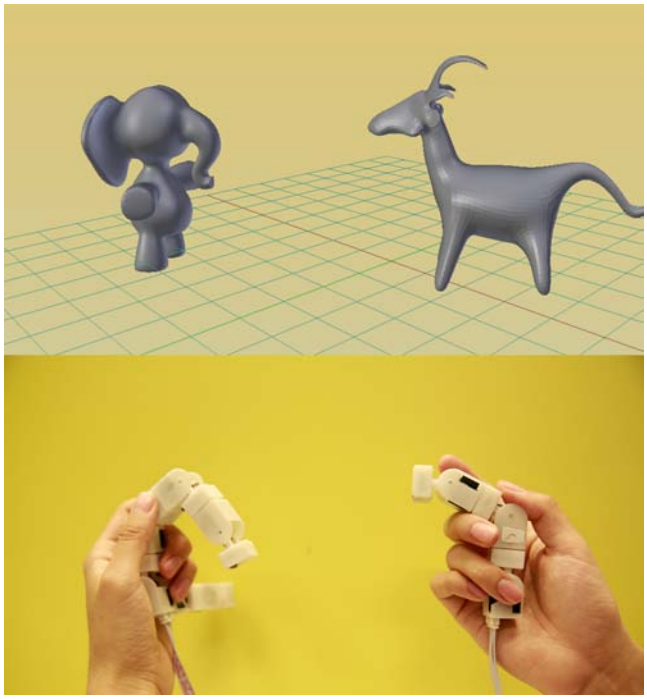

Figure 12. Multiple TwistBlocks systems can be used in the same scene for different characters.

CONCLUSION AND FUTURE WORK

We have demonstrated TwistBlocks, this easy-to-use TUI toolkit allows novice users to create digital 3D models and animate them in a simple manner. We have proposed a global coordinate sensing system, which enables physical manipulation of either a part or the whole model. A 3-DOF physical joint provides best user experience in manipulation, and interaction between multiple models brings possibilities to the creation of complex dynamic scenes such as interactive storytelling.

The design of the modular blocks can be further optimized. The armature structure is affected by the size of the block, so that a more compact design will provide more flexibility. In the next step, we are planning to optimize the demo with our latest designed mockups, and conduct a thorough user study. We are also considering interfaces for additional sensors and actuators to extend the applications.

ACKNOWLEDGEMENTS

Thank Piyush and Mr. Shaoen MA for proofreading, and Ms. Yuan YAO for producing the video. This work is supported by National Key Research & Development Plan of China under Grant No. 2016YFB1001402, and National Natural Science Foundation of China under Grant No. 61402250, 61232013.

REFERENCES

1. Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cedric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. 2014. Tangible and modular input device for character articulation. In *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology* (UIST'14 Adjunct), 45-46.
2. Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. 2012. DuploTrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12), 389-402.
3. 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), 649-658.

4. Christian Rendl, David Kim, Sean Fanello, Patrick Parzer, Christoph Rhemann, Jonathan Taylor, Martin Zirkl, Gregor Scheipl, Thomas Rothländer, Michael Haller, and Shahram Izadi. 2014. FlexSense: a transparent self-sensing deformable surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14), 129-138.
5. David Anderson, James L. Frankel, Joe Marks, Darren Leigh, Eddie Sullivan, Jonathan Yedidia, and Kathy Ryall. 1999. Building virtual structures with physical blocks. In *Proceedings of the 12th annual ACM symposium on User interface software and technology* (UIST '99), 71-72.
6. David Merrill, Jeevan Kalanithi, and Pattie Maes. 2007. Siftables: towards sensor network user interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07). ACM, New York, NY, USA, 75-78.
7. Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 647-654.
8. J. M. Kahn, R. H. Katz, and K. S. J. Pister. 1999. Next century challenges: mobile networking for "Smart Dust". In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (MobiCom '99). ACM, New York, NY, USA, 271-278.
9. Ken Nakagaki, Sean Follmer, and Hiroshi Ishii. 2015. LineFORM: Actuated Curve Interfaces for Display, Interaction, and Constraint. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15), 333-339.
10. Michael Philetus Weller, Ellen Yi-Luen Do, and Mark D Gross. 2008. Posey: instrumenting a poseable hub and strut construction toy. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (TEI '08). ACM, New York, NY, USA, 39-46.
11. Moritz Bächer, Benjamin Hepp, Fabrizio Pece, Paul G. Kry, Bernd Bickel, Bernhard Thomaszewski, and Otmar Hilliges. 2016. DefSense: Computational Design of Customized Deformable Input Devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16), 3806-3816.
12. Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. 2016. Rig animation with a tangible and modular input device. *ACM Trans. Graph*. 35, 4, Article 144 (July 2016), 11 pages.
13. Ryoichi Watanabe, Yuichi Itoh, Masatsugu Asai, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi. 2004. The soul of ActiveCube: implementing a flexible, multimodal, three-dimensional spatial tangible interface. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology* (ACE '04). ACM, New York, NY, USA, 173-180
14. Tovi Grossman, Ravin Balakrishnan, and Karan Singh. 2003. An interface for creating and manipulating curves using a high degree-of-freedom curve input device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03), 185-192.
15. Wataru Yoshizaki, Yuta Sugiura, Albert C. Chiou, Sunao Hashimoto, Masahiko Inami, Takeo Igarashi, Yoshiaki Akazawa, Katsuaki Kawachi, Satoshi Kagami, and Masaaki Mochimaru. 2011. An actuated physical puppet as an input device for controlling a digital manikin. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 637-646.