EduTalk: An IoT Environment for Learning Computer Programming and Physics

Yi-Bing Lin, Fellow, IEEE, Min-Zheng Shieh, Ming-Feng Shih, and Chang-Chieh Cheng

Abstract—This article proposes EduTalk, an out-of-the-box Internet of Things (IoT)-based smart learning environment for programming education. In particular, EduTalk enables the students to write VPython programs that render 3-D animations in the browser without installing extra software or using any specific (and typically expensive) hardware. EduTalk takes the user's smartphone as a controller for cyber–physical interaction, which nicely integrates with learning of other core courses such as physics and mathematics. EduTalk allows building science exhibition projects by writing VPython programs to show 3-D animation, where the cost for EduTalk’s cyber–physical interaction is low and is almost maintenance free. The major contribution of this article is the IoT-based EduTalk proposal that subtly utilizes an IoT platform IoTtalk to conveniently generate cyber–physical interaction for learning how to program as well as learning core courses such as physics. The programming exercises can be easily extended to science exhibition projects and then the development of digital twin applications. A mechanism is provided to easily integrate GlowScript animation demos with EduTalk, which significantly simplifies the effort for teachers to prepare the lecturers. Finally, we show how data collected from EduTalk can be analyzed to improve learning design for cyber–physical interactive animation.

Index Terms—Cyber–physical interaction, GlowScript, Internet of Things (IoT), scratch, smart learning environment (SLE).

I. INTRODUCTION

COMPUTER programming has an integral role in advancing the world’s technologies, and it is our belief that programming is an essential ability for 21st-century learners. Technology is part of our life, and coding brings it closer to us.

Learning the fundamentals of programming helps one innovate and create solutions for global challenges. Specifically, coding enhances creativity, enables people to cooperate across geographical boundaries, and allows communication using a universal language (e.g., Python).

Today, programming is a key component of many college curriculums. Several high schools also provide coding training [1]. In a traditional programming education environment, the students use computers with the preinstalled software in the classroom. To inspire student creativity, some teachers also purchase specific development kits that contain microcontroller boards (MCUs) and some sensors for the students to create the MCU-based programs [2]–[6]. However, establishing such a programming education environment is expensive since besides extra hardware cost, it is typically a tedious task to set up both software and hardware of the learning environment.

To resolve these issues, we propose EduTalk, an out-of-the-box Internet of Things (IoT)-based smart learning environment (SLE) for programming education. In an SLE, the students exercise learning tasks with the support provided by Information and Communication Technology (ICT) tools. The SLE supports physical or virtual learning context and produces educational big data for learning analytics. The data generated by the SLE are manipulated by using various computational and visualization techniques to produce information for both learners and teachers to improve the learning process.

In this article, we propose EduTalk, an online programming education platform which utilizes smartphones as controllers for cyber–physical interaction. EduTalk allows learning both programming and other core courses such as physics by writing VPython programs for physics experiments. Our experiment approach renders 3-D animation in the browser without installing software or purchasing any hardware. Any VPython-based animation of physics experiments can be almost automatically accommodated in EduTalk to provide vivid interactive scientific experiments, such as precession, planet orbiting, and snake pendulum. By scanning a QR code using a smartphone, EduTalk automatically retrieves, for example, the acceleration sensor of the smartphone to drive the snake pendulum animation. When the student shakes the smartphone, the acceleration value is sent to the animation program to change the mass and the gravity values, which in turn affects the behavior of the snake pendulum. In this way, the student gains a deeper impression on the effects of mass and gravity on snake swing, and the student’s learning experience on core subjects is further enhanced. This article is organized
as follows. Section II overviews the related work; Section III describes cyber–physical interaction in EduTalk; Section IV proposes the EduTalk architecture; Section V shows how to create and manage an EduTalk lecture; Section VI elaborates on how the students learn with the EduTalk; Section VII proposes a mechanism to integrate GlowScript demos with the EduTalk; Section VIII demonstrates several science exhibition projects using EduTalk; Section IX describes how data collected from EduTalk can be analyzed to improve the design for cyber–physical interactive animation.

II. RELATED WORK

Several SLE technologies help to achieve successful learning goals. For example, dashboards with analytic tools [7], [8] effectively support the self-regulation of the learning processes. Another example is real-time classroom orchestration [9] that allows a teacher to monitor the detailed learning progress of the class. In [10], a systematic review of empirical evidence on learning analytics is provided for learning design, which portrays the design patterns and learning occurrences imposed on learning technologies.

In [11], the state-of-the-art of workplace learning analytics were reviewed to investigate the researcher conceptions of learning and their influence on learning technology choices. The study in [12] incorporates dispositional dimensions, such as emotions and self-regulation into conventional learning analytics models to explain student behaviors. This study indicates that dispositional learning analytics effectively bridges learning analytics and educational intervention.

Based on a player model, a process for adapting gaming features was proposed [13]. This approach developed five different gaming features in a learning environment. The study indicated that with adapted gaming features, the learners enjoyed more in the learning environment.

Different from the studies surveyed above, this article elaborates on EduTalk, an IoT environment for online programming education with a focus on cyber–physical interaction. There are several online programming platforms based on open source. For example, Scratch [14] is an online community designed primarily for children, where the users create online projects using a block-like interface. Scratch allows the teachers to organize student projects into studios. Community statistics on the Scratch official website show that more than 50 million Scratch projects are shared by over 50 million users. Additional features like voice and user’s movement is provided by Scratch with specific invention kits to conduct cyber and physical interaction. The cyber–physical interaction features of EduTalk can be integrated with Scratch through an extension block. Our experience with Scratch is good except that it is not very convenient to make complex combinations of logic conditions and is not easy to debug complicated programming logic. To our knowledge, Scratch cannot be used to develop commercial ICT software products.

GlowScript [15] is a platform for writing programs in JavaScript or VPython to generate real-time 3-D animations in the browser. It also provides a basic error message to help debugging. GlowScript does not support course management with suitable tools for teaching and assignment. The method of interacting with animation is limited to keyboard and mouse in GlowScript. In Section VII, we will show how to integrate EduTalk with GlowScript to provide a better SLE.

From the above discussion, there are no simple ways to provide nontrivial cyber and physical interaction in programming education. Some solutions [2]–[6] utilized specific development kits, including MCU boards and sensors that can be controlled by programs for cyber–physical interaction. To accommodate such solutions, the students need to perform specific and tedious programming tasks to drive the MCUs, and such programming efforts for hardware integration with the SLE is not essential for programming education. In fact, our experience indicates that such tedious programming may discourage students. To resolve this problem, EduTalk provides an automatic approach to integrate a smartphone with VPython animations, which eliminates the need of extra costs for purchasing specific hardware and the nonessential programming efforts.

III. CYBER–PHYSICAL INTERACTION IN EDU TALK

EduTalk allows students to learn the programming skills with a basic core course (such as physics, chemistry, biology, and math). Without loss of generality, the remainder of this article will use physics as a core course example.

EduTalk utilizes the IoT technology to create connections between the user’s smartphone and the animations for physics experiments. A smartphone is considered as a collection of sensors and controls, and the 3-D animation program is considered as a collection of actuators. We denote sensors, controls, and actuators as “features.” EduTalk automatically connects the smartphone features to the animation features (the input variables of a VPython program), and creates a unique QR code for each lecture. A student scans the QR code with an arbitrary smartphone to control the 3-D animation using the smartphone’s sensors. In this way, EduTalk supports cyber–physical interaction to conduct physics experiments. In this article, the input variables of a VPython program are called “cyber features,” and the sensors/controls that manipulate the VPython program are called “physical features.” EduTalk provides cyber–physical interaction by linking the physical features to the cyber features.

An EduTalk user is either a teacher or a student. EduTalk allows the teachers to create and manage lectures. The students remotely learn the EduTalk lecturers through their own laptops or smartphones without special setups, and the learning experience is extended with cyber–physical interaction. To support the IoT features, EduTalk is developed on top of IoTtalk [16], an IoT development system supporting quick creation of applications to drive the sensors and the actuators. EduTalk automatically generates the IoTtalk applications for physics experiments.

In the EduTalk, the teacher prepares a default VPython program (a 3-D animation) for each lecture. The details will be given in the next section. An example of 3-D animation is the planet orbiting simulation illustrated in Fig. 1(1). The VPython program is automatically transformed into an IoT actuator in IoTtalk, which can be accessed by scanning a QR code.
[Fig. 1(2)] provided by EduTalk. When an EduTalk user scans the QR code with a smartphone [Fig. 1(3)], the smartphone can control the created 3-D animation to exercise physics experiments. Besides the real-time smartphone sensors data, the user can also control the animation with simulated data generated from a Web-based graphical user interface (GUI) (a range slider, an input box, and so on) or a special sensor such as MorSensor that can be easily attached to the smartphone [Fig. 1(4)] [17]. With MorSensor, the smartphone is extended with various sensing features (such as humidity, UV, and alcoholic detectors) displayed on the screen [Fig. 1(5)]. In the planet orbiting simulation, when the user turns the smartphone to some degree, EduTalk maps the $(\alpha, \beta, \gamma)$ values of the orientation sensor to the gravity of the earth and the speed of the moon. Then, EduTalk instructs IoTtalk to use these two features to animate the Moon orbiting the Earth.

As another example, in the parabolic motion simulation, the student makes a throwing ball gesture using a smartphone [Fig. 2(1)]. The acceleration sensor of the smartphone controls the initial speed and the gyroscope sensor controls the angle of the ball. Then, the ball-throw animation [Fig. 2(2)] shows the parabolic trajectory. The trajectory data are displayed in EduTalk dashboards [7], [8], and are recorded for future analysis.

IV. EduTalk Architecture

EduTalk consists of a Web-based GUI, the EduTalk engine, and the Cyber–Physical Interaction System. The GUI [Fig. 3(1)] allows the teachers/students to create and exercise the lectures through the EduTalk engine [Fig. 3(2)]. The details will be given in the next section. The Cyber–Physical Interaction System [Fig. 3(3)] is responsible for bridging the EduTalk engine with the IoTtalk engine [Fig. 3(4)].

The IoTtalk Engine [16], [18] receives the instructions from EduTalk to provide interaction of the IoT devices. Therefore, the cyber-world VPython animations programmed by the students can be controlled by the smartphones or other sensors/actuators in the physical world.

The EduTalk engine consists of three parts: 1) the event handler; 2) the management procedures; and 3) the database. The EduTalk Event Handler [Fig. 3(5)] receives the events sent from the EduTalk GUI, dispatches the events to execute the corresponding management procedures [Fig. 3(6)], and store all data in the EduTalk Database [Fig. 3(7)]. There are three types of EduTalk Management Procedures. The user management procedures [Fig. 3(8)] are responsible for the management of user accounts. The user management also handles learning analytics focusing on the collection, analysis, and reporting of data about students and contexts in which learning occurs. The Lecture Management Procedures [Fig. 3(9)] support the CRUD (create, read, update, and delete) operations on lectures performed by the teachers. The CRUD operations allow the teacher to perform classroom orchestration by modifying the lecture materials to monitor the learning progress and decide when to stop the activity to clarify misunderstandings [9].

The Lecture Management Procedures manipulate a lecture with the following attributes: the lecture name attribute uniquely identifies a lecture; the material uniform resource locator (URL) attribute links EduTalk to the teaching materials; and the program name attribute uniquely identifies the animation to be manipulated. The details will be given in the next section.

To facilitate analysis of a student’s data stored in the EduTalk Database, the DataBank Management Procedures [Fig. 3(10)] collects the student’s historical activity data, which are stored in Microsoft Excel files, and are manipulated by the user management procedures to interact with the dashboards.

Like the EduTalk engine, the Cyber–Physical Interaction System includes three parts. The Event Handler [Fig. 3(11)] is responsible for interaction between the EduTalk engine and the IoTtalk engine by executing the management procedures [Fig. 3(12)]. When the EduTalk engine invokes the requests to
the IoTtalk engine, the HTTP services procedures [Fig. 3(13)] encapsulate the requests into HTTPS messages. The message delivery path is (1)–(5)–(11)–(13)–(11)–(4) in Fig. 3. The Cyber–Physical Interaction Event Handler initializes the above message path and the URLs of the EduTalk services stored in the EduTalk URL Configuration File [Fig. 3(14)]. It also invokes the procedures to create and remove the control links of the IoT devices [Fig. 3(15) and (16)], and generate the applications (e.g., physics experiments) to be executed by the IoTtalk engine.

In the physics programming course, every lecture is associated with one or more cyber and physical IoT devices for the experiments. When a student accesses a lecture through the GUI, the EduTalk Event Handler invokes the Cyber–Physical Interaction System to set up the experiment configuration and include the corresponding IoT devices through the path (1)–(5)–(7)–(5)–(11)–(12)–(11)–(4) in Fig. 3. For the parabolic trajectory example in Fig. 2, the (11)–(4) action creates a parabolic motion project in IoTtalk called “Ball Throw,” and the graphical representation of the project is automatically created by IoTtalk as illustrated in Fig. 4(a). In this figure, the smartphone and the ball throw animation are represented by rectangular icons (1) and (2) in Fig. 4(a). The Join 1 connection links the acceleration sensor of the smartphone to the speed variable of the ball throw animation, and Join 2 links the gyroscope to the angle. The graphical IoTtalk project allows the teacher to double check if the cyber–physical connections are correct. The details of the IoTtalk project are out of the scope of this article, and the reader is referred to [16].

It is important to guarantee that the data are quickly delivered through Joins 1 and 2 [Fig. 4(a)] so that the animation can be played smoothly. We have measured the data delivery delays of 1000 transmissions over WiFi and another 1000 transmissions over LTE communications. The delay histograms are shown in Fig. 4(b). The average data delivery delays over WiFi and over LTE are 54.239 and 81.177 ms, respectively. In both scenarios, the data delivered through Join paths 1 and 2 experience short time delays.

V. LECTURE DEVELOPMENT AND MANAGEMENT

In the EduTalk Web site, the teacher creates a lecture in the lecture page [Fig. 5(a)]. This page is partitioned into the editing area for teachers [Fig. 5(b)] and the viewing area for both teachers and students [Fig. 5(e)]. The teacher fills the name for the lecture [Fig. 5(c)], the URL field linking to the teaching materials [Fig. 5(d)] and the name for the VPython program [Fig. 5(f)]. Then the teacher provides the default code [Fig. 6(a)] for the VPython program from scratch if “New” is selected [Fig. 6(b)]. Alternatively, the teacher may create the default code by modifying an existing program [for example, “BallFreeFall”; see Fig. 6(c)].

The teacher uses the Input field [Fig. 6(d)] to specify the cyber features (the input variables of the VPython program). For the parabolic motion experiment in Fig. 2, “Speed” and “Angle” are selected. The cyber feature names should be the same as the input variable names in the VPython program. If no such names exist in EduTalk, then the teacher simply adds the input variable names as the cyber features [Fig. 6(e)]. When an input name is selected, a pull-down list pops up to show the available physical features (sensors and controls) that can be connected to the inputs of the VPython program. These physical features are classified in three groups: 1) the smartphone sensors for acceleration; 2) gyroscope; and 3) orientation [Fig. 6(f)]; the sensor devices (for example, MorSensor) for magnetometer, humidity, UV, and alcohol detection [Fig. 6(g)]; and the browser controls for range slider and input box [Fig. 6(h)]. Examples of range slider and input box are given in Fig. 10(a) and (c). For the example in Fig. 2, the acceleration feature is selected to map to Speed in Fig. 6(i), and IoTtalk automatically creates the Join 1 link in Fig. 4(a).

The teacher may delete the lecture through the “Delete Lecture” button [Fig. 5(g)]. When the “View/Edit” toggle button [Fig. 5(h)] is set in the “Edit” mode, the lecture editing
area is activated, where the teacher can edit the lecture. If the “View” mode is selected, the lecture editing area is hidden, and the user can view the lecture note through the viewing area [Fig. 5(e)].

VI. LEARNING WITH EduTALK

When a student accesses a lecture, the lecture page shows the viewing area [Figs. 5(e) or 7(a)]. The student cannot add a new lecture or modify the lecture materials. After reading the lecture note, the student may click the “Program” button [Fig. 7(b)] to start programming exercise.

When a user (a teacher or a student) first accesses the program page, the code window (Fig. 8) lists the default code created by the teacher [Fig. 8(a)], and the students are allowed to modify the VPython programs in their code windows [Fig. 8(b)]. Note that the students have their own copies of the program and therefore they will not interfere with each other. The “Save” button [Fig. 8(c) and (d)] is clicked to save the modified code into the EduTalk database. The program syntax errors are automatically detected and shown on the error message box [Fig. 8(e)]. In this example, the teacher purposely writes “variable=,” and the students are expected to correct the mistake. When the “Reset” button [Fig. 8(f) and (g)] is clicked, the default code of the lecture is restored. If the teacher clicks the “Set as Default” button [Fig. 8(h)], the modified code is released as the default program of this lecture.

When the user clicks the “Animation” button [Fig. 8(i) and (j)], the IoTtalk engine executes the program, and the animation window pops up to show the animation effect [Fig. 9(a)]. If the scanning icon [Fig. 9(b)] is clicked, the window pops up a QR code that can be scanned to connect a smartphone [like the one shown in Fig. 1(2)].

One can use the QR code scanner of a smartphone to access the remote-control page (Fig. 10). If multiple participants scan the QR code, the animation is controlled by the last person. The remote-control page provides the controls and sensors to manipulate the animation, which are generated based on the teacher’s setting when the lecture is created. Fig. 10(a) is the range slider that controls the gravity [Fig. 10(b)]. To input the radius data, the user types a number in the input box [Fig. 10(c)]. If a sensor is selected [the button is turned on; e.g., Fig. 10(f)], it automatically sends the real-time measured data [Fig. 10(e)] to drive the animation.
Possible lecturer design for physics experiments and the corresponding Python programming education includes 1-D constant speed motion (where the students learn basic Python structure and while loop), 1-D constant speed motion (to learn Python if statement), 3-D motion, circular motion, Hooker’s Law and the simple harmonic movement (to learn Python gcurve), synthesis of force vectors (to learn Python vector, list and for loop), momentum (to learn Python dictionary), elastic collision (to learn Python tuple and function), planetary revolution (to learn Python class), and so on.

VII. INTEGRATION OF EDUTALK AND GLOWSRCRIPT

GlowScript provides many VPython demos, which becomes a rich source of “cyber IoT devices.” These VPython demos can be semi-automatically integrated with EduTalk. In lecture creation, we can include a GlowScript animation program through code selection [Fig. 6(d)]. Then, EduTalk connects the input variables of the GlowScript animation program with the physical features in two steps. The teacher first selects the input variables of the GlowScript animation program [Fig. 6(d) or (e)]. Then the teacher selects the physical feature [Fig. 6(f), (g), or (h)] for every GlowScript input variable to be controlled.

In EduTalk, the values of the physical features change in real time. Unfortunately, most GlowScript inputs (cyber features) are designed as fixed values. Therefore, when the physical features change, the animation needs to be redrawn according to the new values. Consider the Bounce-VPython demo as an example (Fig. 11).

The VPython program consists of three parts. The first part builds the red and blue walls [Fig. 11(1)]:

The second part creates the green ball [Fig. 11(2)] and its trajectory [tail; Fig. 11(3)]:

The third part creates the animation:
In this program the mass of the ball is a fixed value; that is, ball.mass = 1.0 (line 21). Suppose that we want to use a range slider to control the mass of the ball, then we use the input field [Fig. 6(d)] to create a cyber feature named “mass” and link it to a range slider [Fig. 6(h)] in the lecture creation page. After the lecture is created, EduTalk automatically inserts one line of code “ball.mass = mass” between lines 28 and 29. The inserted line binds the mass of the ball to the latest value of mass because the mass value is automatically updated upon every change of the corresponding range slider. Based on the modified mass of the ball, the program recalculates the position of the ball in line 29 and draws the next frame of the animation.

VIII. EduTalk-Based Projects for Science Exhibitions

EduTalk is an excellent tool for students’ science exhibitions. Through EduTalk, students’ creative designs can be easily implemented as Python programs that automatically interact with the smartphones and other IoT hardware devices. In 2019, two high school sophomores used what they learned from EduTalk physics to design a virtual shooting machine, a darts machine and a doll-clamping machine, in which a player could interact with these games from anywhere with any smartphone. This work participated in the 2019 Mobileheroes Competition of Ministry of Economic Affairs (MOEA, Taiwan) competing with the commercial products of universities and start-ups, and was the only high school team to make the cut after three rounds of elimination among 167 teams, and was finally ranked top 30. In summary, EduTalk incorporates computer programing education into physics teaching, which can be easily extended to construct science exhibitions. This section gives examples of EduTalk-based science exhibition projects at the university, the high school and the primary school levels.

Like GlowScript, Scratch can also be integrated with EduTalk through an EduTalk extension block. Fig. 12 illustrates a Scratch-based EduTalk project created by primary school students. In this cat-catching-mouse project, the player uses an arbitrary smartphone [Fig. 12(1)] to scan the project QR code, and then the smart phone’s orientation sensor is used to control the cat movement in the Scratch animation [Fig. 12(2)]. The Scratch code [Fig. 12(3)] involves an extension we built for EduTalk [Fig. 12(4)].

To promote smart agriculture education, EduTalk was used to develop a tree animation for a senior high school science exhibition project. In this project, a micro weather station [Fig. 13(1)] includes the sensors for humidity, temperature, CO2, luminance, and so on [Fig. 13(2)]. EduTalk automatically connects this animation to the micro weather station [Fig. 13(2) and (3)], and the tree grows according to the weather conditions. For example, when the humidity increases, the tree grows the leaves [Fig. 13(4) and (5)]. When the luminance increases, the sky turns blue [Fig. 13(5) and (6)]. The light intensity and humidity readings of the weather station are shown in Fig. 13(7).

In a high school science exhibition project for the pendulum experiment, we built a physical pendulum based on MorSensor [Fig. 14(a)], where a weighted bob [Fig. 14(1)] connects to a pivot [Fig. 14(2)] through a weightless cord that can swing freely.

When the bob is placed sideways from its resting equilibrium position indicated by a measured board [Fig. 14(3)], a restoring force will accelerate it back toward the equilibrium position [Fig. 14(4)]. The bob is a cube-shape MorSensor [Fig. 14(5)] that measures the acceleration data and sends them to the EduTalk server through Bluetooth [17]. The physical acceleration feature of the MorSensor is mapped to the cyber feature of the Python pendulum animation displayed in a big screen [Fig. 14(6)]. The measured data are also illustrated on the screen [Fig. 14(7)]. With this animation, the students can observe the pendulum phenomenon by replaying the data stored in the EduTalk database.

In Fig. 14(a), extra costs are required to use specific hardware (MorSensor), 3-D-printing bob holder and the string. In

1 https://www.youtube.com/watch?v=pokylIFldYM
2 https://www.youtube.com/watch?v=bGb-pf3ra4s
3 https://www.youtube.com/watch?v=s1UaA-Xunq4
Fig. 14. EduTalk pendulum experiment. (a) MorSensor version. (b) Smartphone version.

In Fig. 14(b), we designed a physical pendulum based on a smartphone without purchasing extra hardware. In this experiment, a smartphone [Fig. 14(8)] is used as the bob, and an iron ruler is used as the pendulum string ([Fig. 14(9)].

The acceleration data [Fig. 14(10)] are sent to the VPython program for animation [Fig. 14(11)].

We also design the circular motion experiment using a smartphone [Fig. 15(a)]. We put the smartphone [Fig. 15(1)] on the armrest of a swivel chair [Fig. 15(2)]. Then we rotate the chair to collect the data to animate the VPython program [Fig. 15(3)], and the data are plotted in a graph where the measured data [the blue dots; Fig. 15(4)] are validated against the theoretical curve [the green curve; Fig. 15(5)]. In a smartphone-based EduTalk simple harmonic motion experiment [Fig. 15(b)], a smartphone [Fig. 15(6)] is attached to a spring [Fig. 15(7)] to exercise the simple harmonic motion. The data are collected by EduTalk to animate the VPython program [Fig. 15(8)], and are plotted in a graph [Fig. 15(9)].

EduTalk has been used to develop interactive art animation for undergraduate projects in the colleges. An example is cyber–physical interaction of skeleton ceiling light that can change shape and light color during the night to reflect the shadow of different geometric shapes on the floor. The original skeleton light won the silver award of Salon International Des Invention, Geneva, Switzerland [Fig. 16(1)]. We later developed a simplified skeleton [Fig. 16(2)]. With a color lightbulb inside the skeleton body, the physical skeleton is an interactive ceiling lamp where the shape change is achieved through compression of the skeleton stalks with various angles and sizes. Through growing and compressing, Skeleton’s shadow in the floor shows beautiful geometric patterns [Fig. 16(3)]. Through IoTtalk, the physical skeletons can be directly controlled by smartphones with three features, including size, angle, and color. Through EduTalk, the students built a skeleton animation [Fig. 16(4)], where the cyber features (size, angle, and color) can be mapped to the physical skeletons or the smartphones.

---

4https://www.youtube.com/watch?v=WHn7260OGXg
5https://www.youtube.com/watch?v=cmGMqQdYzmM
6https://www.youtube.com/watch?v=vnK4LEWdDHm
For a graduate-level project in a university, we have developed a glove puppetry application [20]. We built a simplified puppet robot by removing two fingers of an off-the-shelf hand robot with three joint features for control [Fig. 17(1)]. We have also built an enhanced robot with 14 joint features [Fig. 17(2)]. Through IoTtalk, these puppet robots can be controlled by smartphones or smart gloves. Through EduTalk, we built robot animation [Fig. 17(3)] with cyber features mapped to 14 corresponding joint features of the enhanced robot.7

The cyber–physical interaction projects described in this section can be easily extended to the development of digital twin applications.

IX. ENHANCING EDU TALK USER EXPERIENCE

Empirical analytics for learning design was elaborated in [7], [8], [10], and [11]. In this section, we describe how data collected from EduTalk can be used to investigate student learning and to improve learning design for EduTalk animation.

A. Learning Statuses of the Students

When a student starts to learn a lecture by pressing the “Lecture” button [Fig. 5(a)], to program by pressing the “Program” button [Fig. 7(b)], to view the resulting animation by pressing the “Animation” button [Fig. 8(j)], and when the teacher interacts with the student for this lecture, the events are captured by EduTalk’s DataBank Management Procedures [Fig. 3(10)]. Then, we can conduct a primary observation on student learning through a simple finite state machine (FSM) illustrated in Fig. 18(a). In EduTalk, a student is associated with a learning FSM. When the student clicks “Lecture,” the FSM moves from state S (start) to state 1 (blue). When the student clicks “Program,” the FSM moves from state 1 to state 2 (red). When the student clicks “Animation,” the FSM moves from state 2 to state 3 (yellow). When the teacher evaluates the program, the FSM moves from state 3 to state 4 (green). If the teacher decides that the student should go through the learning process again, the FSM moves from state 4 to state 1. If the teacher is satisfied with the student work, the FSM moves from state 4 to state E (end).

Fig. 18(b) illustrates three pie charts, each of them indicates the percentages of time that a student spends in the states of his/her FSM. Pie charts A and B show that students A and B spend more than 50% of their time in learning the lecture. Student C spends more than 70% of his/her time in programming. The teacher spends a lot of effort in guiding student A to repeat the learning process. The EduTalk learning FSM allows a primary observation on student learning, which addresses RQ3 of [12]: We can profile students into clusters based on their online activities. Following this primary observation, one future direction for EduTalk is to conduct more in-depth secondary investigation to answer RQ1 [12] (To what extent academic performance can be predicted by online activity trace data) and RQ2 (To what extent can the use of the e-tutorials be predicted by disposition data obtained by self-reports).

B. Improving Design for Animation

The data collected from EduTalk can also be used to improve learning design for EduTalk animation. This subsection uses the planet orbiting simulation as an example to show how user experience can be enhanced through the collected data. This animation has the problem that if a participant (a player) changes the demo parameters (i.e., the speed or the gravity) before the Moon circles the Earth for a complete round, the incomplete animation will confuse the audiences. The problem is resolved as follows. Denote such

7https://www.youtube.com/watch?v=oyay_uvIm0g
delay \( t_0 \) as the “complete animation time.” In Fig. 19, suppose that the demo starts the animation at time \( t_0 \), and the animation completes one round at time \( t_0^\ast \), then the complete animation time is \( t_0^\ast = t_0 + t_0 \). Assume that after \( t_0 \), the participants request to change the demo parameters at times \( t_1 < t_2 < \cdots < t_{m-1} < t_m \), where \( t_i = t_0 - t_{i-1} \) for \( i \geq 1 \). Note that both \( t_0 \) and \( t_i \) can be recorded in the EduTalk database, and then we can plot their histograms just like what we measured the data transmission delays in Fig. 4(b). These histograms can be approximated by some probability density functions, and we assume that both \( t_0 \) and \( t_i \) are random variables. If we approximate the \( t_i \) histogram by the Erlang density function \( g(t_i) \) with the shape parameter \( n \) and the scale parameter \( \lambda \), then its density function and the cumulative distribution function are expressed as

\[
g(t_0) = \frac{\lambda^n t_0^{n-1} e^{-\lambda t_0}}{(n-1)!} \quad \text{and} \quad \int_{t_0}^{t_0^\ast} g(t_0) = 1 - \sum_{j=0}^{n-1} \frac{\lambda^j t_0^j e^{-\lambda t_0}}{j!}.
\]

The Erlang random variable is considered to model \( t_0 \) because the Erlang distribution or a mixture of Erlang distributions are typically used to model computation times in telecommunications [21], [22]. Similarly, we assume that \( t_i \) are i.i.d. random variables with the density function \( f(t_i) \) for \( i \geq 1 \). Let \( f^*(s) = \int f(t_i) e^{-s t_i} dt_i \) be the Laplace transform of \( f(t_i) \). Let \( T_i = t_0 - t_0 = \sum_{j=1}^{i-1} t_i \) have the density function \( f_i(T_i) \). From the convolution rule, the Laplace transform \( f_i^*(s) \) of \( f_i(T_i) \) is expressed as

\[
f_i^*(s) = [f^*(s)]^i.
\]

If more than \( m \) consecutive requests occur during a complete round of animation, then \( t_m = t_0^\ast \), and its probability is

\[
\Pr[T_m < t_0] = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m.
\]

From frequency-domain derivative of the Laplace transform, (2) is rewritten as

\[
\Pr[T_m < t_0] = \int_{t_0}^{t_0^\ast} \int_{t_0}^{t_0^\ast} g(t_0) dt_0 dT_m.
\]

Substitute (1) into (3) to yield

\[
\Pr[T_m < t_0] = \sum_{j=0}^{n-1} \frac{(-\lambda)^j}{j!} \left[ \frac{[f^*(s)]^n(s)}{s^{j+1}} \right]_{s=\lambda}.
\]

If \( j = 0 \) then

\[
\Pr[T_m < t_0] = \left[ \frac{(-\lambda)^j}{j!} \left[ \frac{[f^*(s)]^m(s)}{s^{j+1}} \right] \right]_{s=\lambda}.
\]

If \( m \geq j > 0 \) then

\[
\Pr[T_m < t_0] = \left[ \frac{(-\lambda)^j}{j!} \left[ \frac{[f^*(s)]^m(s)}{s^{j+1}} \right] \right]_{s=\lambda}.
\]

If \( j > m > 0 \) then

\[
\Pr[T_m < t_0] = \left[ \frac{(-\lambda)^j}{j!} \left[ \frac{[f^*(s)]^m(s)}{s^{j+1}} \right] \right]_{s=\lambda}.
\]

From (5)–(7), we have

\[
\Pr[T_m < t_0] = \left[ \frac{[f^*(s)]^m(s)}{s} \right]_{s=\lambda} + \left[ \frac{[f^*(s)]^n(s)}{s} \right]_{s=\lambda}.
\]

If we approximate the \( t_i \) histogram by the Gamma distribution (a generalization of the Erlang distribution) with the shape parameter \( \alpha \) and the scale parameter \( \beta \), then the Laplace transform for \( f(t_i) \) is

\[
f^*(s) = \frac{\beta^\alpha}{(s+\beta)^{\alpha}} \quad \text{and} \quad \left[ \frac{f^*(s)}{s^m} \right]_{s=\lambda} = \frac{(-1)^m \Gamma(\alpha+j) \beta^\alpha}{\Gamma(\alpha)(\lambda+\beta)^{\alpha+j}}.
\]

For the mean value analysis of \( t_0 \) [23], we set \( n = 1 \) [i.e., \( j = 0 \) in (5) or (8)], then from (5) and (9), we have

\[
\Pr[T_m < t_0] = \left( \frac{\beta}{\lambda + \beta} \right)^m.
\]

Since \( \lambda = (1/E[t_0]), \alpha = (1/C[t]), \) and \( \beta = (1/(C[t]E[t])), \) (10) is rewritten as

\[
\Pr[T_m < t_0] = \left( \frac{E[t_0]}{E[t_0] + C[t]E[t]} \right)^m.
\]

Based on (11), Fig. 20 plots \( \Pr[T_m < t_0] \) against \( m, E[t_0], E[t], \) and \( C[t] \). The figure indicates that \( \Pr[T_m < t_0] \) increases as \( E[t_0]/E[t] \) and \( C[t] \) increase or \( m \) decreases.

If we conduct the mean value analysis for \( t_i \); i.e., \( \alpha = 1 \), then (11) is simplified as

\[
\Pr[T_m < t_0] = \left( \frac{E[t_0]}{E[t_0] + E[t]} \right)^m.
\]
Fig. 20. \( \Pr[T_m < t_0] \) against \( E[t_0], E[t_1], m, \) and \( C[t_1] \).

If we attempt to limit \( \Pr[T_1 < t_0] \) to be less than \( \delta \), then from (12), we should set

\[
E[t_1] \geq \left( \frac{1 - \delta}{\delta} \right) E[t_0].
\]

For example, if \( E[t_0] \) is 5 s, and \( \delta = 0.1 \), then we should select \( E[t_1] > 45 \) s. In the above analysis, the complete animation time \( t_0 \) can be obtained based on the user experience questionnaire using the situational motivation scale (SIMS) approach [13] with a feedback mechanism. The interarrival times of user request \( t_i \) are automatically measured by the EduTalk engine. Equation (13) is used as a quick check to see if we need to adjust \( t_0 \) or \( t_1 \). From the curves in Fig. 20, if \( \Pr[T_m < t_0] \) is approximately equal to 0 for small \( m \) values, then we may design a blocking mechanism such that EduTalk does not accept the \( m \)th user request if \( T_m < t_0 \) (such situation does not occur often anyway). On the other hand, if \( \Pr[T_m < t_0] \) is large, then there are three solutions. First, the teacher/student may redesign the animation to reduce \( t_0 \). Second, the participated players are advised to wait (to increase \( t_1 \)), and do not make requests before \( t_0 \). In the third solution, the user requests are queued in the EduTalk engine, and are executed until the previous round of animation is complete. In the third solution, the participants will experience lagged animation.

X. Conclusion

Based on an IoT technology called IoTalk, we developed EduTalk, a programming education platform whose usage has been outreach from universities to high schools. EduTalk allows a student to write Python code to implement a 3-D animation program controlled by a smartphone. Besides learning programming skills, EduTalk is also integrated with learning of other core courses such as physics. In this way, the student gains a deeper impression on physics phenomena during writing VPython programs, which further enhances the student’s learning on core subjects.

The major contribution of this article is to develop the EduTalk that subtly utilizes the IoTalk platform to conveniently generate cyber–physical interaction for learning how to program as well as learning core courses such as physics.

The results of programming can be easily extended to create science exhibition projects and then the development of digital twin applications. A mechanism is provided to integrate GlowScript and Scratch animation demos with EduTalk, which significantly simplifies the effort for teachers to prepare the default animation programs of the lecturers. Finally, we showed how data collected from EduTalk can be analyzed to improve the learning design for cyber–physical interactive animation.

In the future, students’ motivation to use the EduTalk will be evaluated using the SIMS approach. We will conduct enjoyment survey of EduTalk features (lecture access, program creation, cyber–physical feature binding, and so on) through a questionnaire. We will also use the EduTalk database to collect all actions and outcomes during student learning sessions.

REFERENCES


Yi-Bing Lin (Fellow, IEEE) received the Ph.D. degree from the University of Washington, Seattle, WA, USA, in 1990.

He is a Chair Professor of the following universities: National Yang Ming Chiao Tung University, Hsinchu, Taiwan; China Medical University, Taichung, Taiwan; National Cheng Kung University, Tainan, Taiwan; and Asia University, Taichung; and an Adjunct Research Fellow of Academia Sinica, Taipei, Taiwan. From 1990 to 1995, he was a Research Scientist with Bellcore, Piscataway, NJ, USA. He then joined National Chiao Tung University (NCTU), and became a Senior Vice President of NCTU in 2011. From 2014 to 2016, he was a Deputy Minister of Ministry of Science and Technology, Taiwan. He has coauthored the books *Wireless and Mobile Network Architectures* (Wiley, 2001), *Wireless and Mobile All-IP Networks* (John Wiley, 2005), and *Charging for Mobile All-IP Telecommunications* (Wiley, 2008).

Dr. Lin is an AAAS Fellow, an ACM Fellow, and an IET Fellow.

Min-Zheng Shieh received the B.S. and M.S. degrees in computer science and information engineering and the Ph.D. degree in computer science and engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2003, 2004, and 2011, respectively.

From 2012 to 2016, he served as an Assistant Research Fellow with the Information and Communication Technology Laboratories, National Chiao Tung University. Since 2016, he has been an Assistant Professor with the Information Technology Service Center, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. His main research interests include computational complexity, algorithms, coding theory, and Internet of Things.

Ming-Feng Shih received the bachelor’s degree from National Chiao-Tung University, Hsinchu, Taiwan, in 1991, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 1998.

In 1998, he joined Physics Department, National Taiwan University, Taipei, Taiwan, as an Assistant Professor. He became a Full Professor there since 2005.

Prof. Shih is a member of the Optical Society of America.

Chang-Chieh Cheng received the B.S. degree in information management from the National Yunlin University of Science and Technology, Douliu, Taiwan, in 2003, and the M.S. and Ph.D. degrees in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2007 and 2014, respectively.

He is currently an Assistant Professor with the Information Technology Service Center, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. His primary research projects focus on image processing, computer vision, and computational imaging. He is also interested in IoT applications and artificial intelligence.