Open Research Tools for the Learning Technologies Innovation Lab

https://doi.org/10.3991/ijoe.v18i14.35077

Matthias Ehlenz(✉), Birte Heinemann, Ulrik Schroeder
Learning Technologies Research Group, RWTH Aachen University, Aachen, Germany
ehlenz@cs.rwth-aaachen.de

Abstract—Orchestrating scientific work in educational research laboratories is demanding, especially when many interdisciplinary perspectives are involved. A monolithic approach does not suffice here. This paper describes an open-source architecture for an educational research laboratory. The presented system assists interdisciplinary scientists in implementing prototypes, evaluating new didactical approaches, researching collaborative learning processes, collecting learning analytics data, abstracting and automating experimental procedures, and securely monitoring progress within the lab, even in hybrid or remote setups. Due to both asynchronous and synchronous capabilities, the presented components make it easy to virtualize experiments, incorporate them in courses and lectures, and empower self-regulated but still monitorable learning processes. The ecosystem described originated in the context of collaborative educational Serious Games for interactive table-top displays but is implemented modular and scales well regarding concurrent experiments, connected clients, and other use cases like VR and smart environments.

Keywords—lab-based learning, learning analytics, educational research, serious games, collaborative learning, orchestration, research ecosystem

1 Introduction & motivation

In advance of learning technologies, lab-based research has a long tradition, predating the work presented here. Skinner [1], Pawlow [2], and many other big names from educational research tried concepts like programmed learning [3] and researched the behavior of their subjects in controlled environments. But since those days, the nature of lab-based teaching has changed. Individual learners using devices interconnected by digital systems drastically changed the variables for learning and teaching.

Tablet computers, VR and AR devices, and even large interactive table-top displays make it possible to simultaneously interact with one device as a whole group without binding all attention to the medium. However, this new form of self-regulated, collaborative learning with systems capable of automated feedback comes with new challenges. It requires psychologists, educators, and human-computer interaction researchers to revisit some of their foundations. Furthermore, those settings lead to a revival of lab culture within the learning technologies research community, demanding the implementation of new tools, methods, and prototypes.

Our earlier research focused on proprietary technologies like the smart table [4]. While these approaches produced some promising results, they hibernated with the hardware being declared end of life by the manufacturer. With the rise of capacitive technology, the research efforts on educational capabilities resurfaced again. Starting with 27” displays in late 2015, the previous conceptual designs were taken into active development, now based on an open technology stack like described in [5]. One of the first such devices of a size suitable for collaboration use-cases was the Microsoft Surface Hub 84”, which was released in June 2015. Shortly after, in May 2016, the TABULA project started to enhance tangible learning on such devices, as presented in [6]. One sub-goal was the development of two research prototypes. Following the *Do not Repeat Yourself* principle, a generic approach was developed. It started with collaborative learning with interactive (multi-touch) table-top systems but has grown into a multi-purpose ecosystem [7] for lab-based research on user behavior.

This article describes this multi-purpose ecosystem, presents requirements, and shows solutions. While some of the components described in the upcoming section might be specific to multi-touch devices, the focus of this article -the research toolkit- is intended to be used for all kinds of research in lab-based learning, including features for learning analytics [8].

At the time of writing, five undergraduate software labs have built upon this technology. Students worked on various seminar topics: multi-touch applications, collaboration, and gamification. Additionally, nearly 30 bachelor’s and master’s theses are developed within that MTLG ecosystem (MTLG – Multi-Touch Learning Game). Much progress has been made due to the combined effort of many people involved, including students and student employees, as well as the team members.

2 A structured approach to a framework

Integrating multi-touch technology within teaching heterogeneous students and the project context requires a structured approach to developing learning applications [9].

Implementing MTLG as a framework instead of a library has been deliberate. There are varying definitions, and one broadly acknowledged is that libraries are usually way more specific and assist in completing certain tasks. Thus, it might have been feasible to implement most functionality of MTLG in a library. The significant difference between framework and library is in the control of flow. With a library, the developer must manage the control of the application; a framework inverts the control and takes charge of the life cycle. The framework approach has been chosen to assist beginners as much as possible and to make the products more maintainable. Nevertheless, MTLG was implemented to provide a high degree of freedom. There is a minimal set of conventions to be adhered to, and the many configuration options are populated with suitable default values. As a result, a minimal working learning game can be obtained within a few minutes, even without deep knowledge of web technologies.

The components forming the MTLG ecosystem can be divided roughly into three perspectives: Developers of research prototypes use MTLGcore and MTLGmodules, the dev-ops perspective is covered by MTLGtoolchain and MTLGinfrastructure, and MTLGresearch describes a toolset for scientific usage, which matches the definition of [7] describing an ecosystem as "a collection of software projects which are developed and co-evolve in the same environment". Following modular design principles comparable to the MAP's in engineering [10], adjustment, stacking, component sharing, sectional and component swapping are important basics of the design approach.

2.1 Choice of technology

There have been various explicit and implicit attempts to create development as well as research frameworks for interactive table-top displays [4], [6], [11]–[16]. Those function as the foundation for the presented ecosystem. The problems of the former solutions fall into at least one of the following categories:

Single purpose creation: Having just one learning game in mind easily limits the focus of development of the framework itself to the capabilities to fulfill the intended purpose [15]. There is little to no abstraction, and interfaces remain undocumented.

Technological constraints: As well as the focus on a single application, focusing on a specific technology will most likely attach a kind of expiry date to software and frameworks [4]. This is especially the case if proprietary hardware is involved, probably accompanied by specialized software development kits.

Projects decaying beyond funding periods: There are promising examples of research-driven software frameworks in scientific literature. However, looking deeper into the matter often yields links to partial, no longer maintained repositories and announcements of discontinuation after public funding was over [14] [10].

"Betting on the wrong horse": There is a multitude of examples where technologies and software development approaches have been considered "the next big thing". One example is Adobe Flash, for a long time a feasible solution for interactive web content, now discontinued. A project might have delivered a sufficiently generalized framework, but even if the repositories are still accessible years after the project wrapped up, but the version and the accompanying Development Kits might have moved on one or two major releases. Building on this would require a porting effort.

The decision for HTML5 and JavaScript solves many of the above problems. Early on, backwards compatibility has been a core issue of web development. Though there have also been approaches here that did not reach longevity, like Flash or in-browser Java plugins, but regarding the core functionalities of HTML, CSS, and JavaScript, websites from the early days of the internet still have a high probability of being rendered (at least close the intended representation) in a browser.

Full control over every bit of code is tempting, but the arguments for integrating external components are overwhelming. Thus, the MTLG ecosystem does not strive to provide own tools, i.e., for drawing, rendering, or handling assets. Instead, existing open-source solutions are integrated and interfaced, always aiming for a loose coupling between those components to ease future replacements (MAP principle [10]).

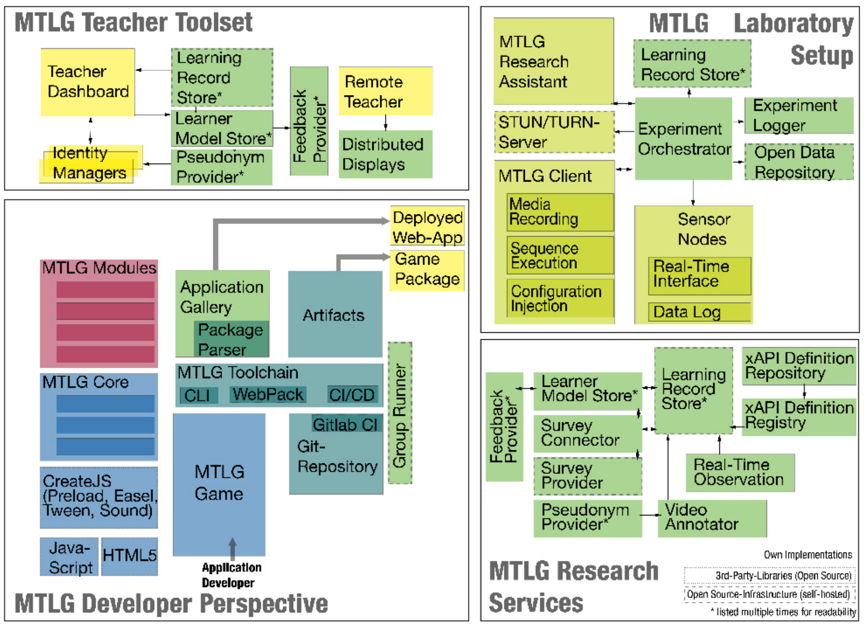


Fig. 1. Schematic overview of the presented ecosystem

In the described state of MTLG, the development framework parts ship with the CreateJS suite consisting of PreloadJS for (asynchronous) asset management, TweetJS for animations, and EasleJS for convenient drawing on the HTML5 canvas element. Maintaining a loose integration ensures alternative approaches in the development of learning applications, like using WebGL for graphical representation of content. The development focused mainly on Chromium-based browsers, as those provide the most reliable support of multi-touch input, but interfaces have been implemented to overcome limitations and provide hypothetical support for other browsers. A native component implemented in any language can intercept the touch device's interactions and provide access to those via a local WebSocket connection.

Considering the growing market share of multi-touch capable end-user devices, it can be expected that those interfaces can be considered as a safe fallback solution, as support of multi-touch interaction might improve in upcoming browser generations.

2.2 Prototype development: core building blocks

The MTLGcore is the basic building block of any learning application. It orchestrates all components, is responsible for loading modules, menus, and levels, and provides basic functionalities. In addition, it is responsible for life cycle management, loading specific settings, and configuring used models.

The first and foremost requirement of the MTLGcore module is ease of use. The loading of supplementary modules is intended to be as quiet as possible, as well as the processing of configuration.

Lifecycle management is often encountered as soon as learning games defer from a straightforward linear storyline. While most games follow a traditional approach of growing complexity, learning games could provide possibilities for an adaptive approach that allows repetition or options to skip levels or personalization [17]. This adaptiveness could be implemented in the background without the knowledge of the individual learner. The lifecycle management is implemented to facilitate all approaches in a common, easy-to-implement fashion.

First experiences showed that a crucial component for developers is the proficient handling of assets. **Asset management** is a non-trivial task. The task is usually not obvious to the (novice) developers during the implementation process, as all resources are stored locally during the development. But assets must be fetched remotely in most deployed contexts. This leads to a subpar user experience, as sounds are played with delay and images are not properly displayed.

Fortunately, solutions already exist, like the PreloadJS library in the CreateJS bundle. Still, to maintain a rather loose coupling, improve compatibility for future changes, and to keep the learning curve flatter, the MTLGcore tools abstract that part of development. Thus, the core module provides interfaces to use preloaded assets through a certain grade of abstraction, while automating i.e., the creation of manifest files listing the integrated assets, which is required for the PreloadJS library to function properly. Thus, all essential assets are fetched into memory before the application starts, showing a splash screen with a progress bar.

The **Internationalization** core module has been implemented to abstract the usage of translations. There are multiple libraries available for that task and this module currently integrates i18n to read language maps and make use of those functionalities within the learning application.

The **User Management** component allows the integration of various login procedures, to match various contexts with different requirements. While collaboration research is often done in a lab setup with injected pseudonyms, usage in schools aims for ease of use for teachers and pupils. For pre-school children, a "login" procedure with symbols is used. As GDPR strictly regulates data collection and classrooms are rightfully considered protected environments, additional measures must be taken to collect interaction data into a provisioned data collection infrastructure while still allowing the teaching personnel to take advantage of the progress monitoring capabilities of learning analytics. Thus, all information leaving the school network is pseudonymized, with the only information linking datasets to students always remaining on the teacher's device.

To allow pupils to continue their personal progress at home there must be ways to allow this while maintaining the learners' privacy. The User Management component frees the developer of learning games and researchers from those considerations and just hooks in the respective log-in mechanisms. The usernames and pseudonyms are provided via simple getter methods to use the former for displaying messages in the learning application and the latter for storage of progress and learning analytics data.

Beyond those there are many more modules beyond the core modules, i.e., to incorporate tangible devices, integrate tablets as "private display space," adapt feedback or provide tools for classroom management. Still, those are not within the scope here.

2.3 Deploying the prototype: toolchain & infrastructure

The MTLGtoolchain provides some convenience tools directed at developers implementing lab-based learning applications. Mainly two challenges have been discovered: bootstrapping and deployment. A separate repository is used for modularization.

The MTLGtoolchain consists of two major parts: A command-line interface (CLI) and the pipeline components for continuous integration & deployment. The CLI provides a set of tools which developers can use to create boilerplate code for new applications, to build them for debugging or production, to generate example code for the different modules, spin up a local webserver for testing or for bundling up the application. Those bundles can then be used in the research client software described later and be conveniently archived along with the research data for future reusability.

3 The researchers toolset

The previously described parts form the foundation towards a holistic research framework by providing the necessary tools to implement high-quality learning applications for collaborative learning. Complementary, three software components have been developed to support researchers in creating methodologically sound case studies in a transparent and reproducible way. Before discussing them in detail, a set of five possible application use cases is shown. The use cases have been selected as edge cases. Most case studies in the field of web-based interaction research should be locatable somewhere in the space spanned by these edge cases:

Collection in the wild: Most applications are serious games or contain game elements, e.g. [18]. For these applications, learners could decide to continue beyond school or lab sessions since the games built upon web technologies do not (technically) rely on the presence of multiple players or touch interfaces[1]. The data could be collected and provide further insights into in-game usage and self-regulated learning processes. A dedicated client as an alternative to using the games in-browser might be considered an easy opt-in opportunity for pseudonymized data collection. Additionally, this leads to an encapsulated, tested environment eliminating the need for any browser configuration.

Usage in schools: The usage of collaborative Serious Games in schools brings other constraints: Teachers might allow free choice of the game but would probably prefer to enforce the one fitting into the current lesson. Furthermore, the benefits of *Learning Analytics* might provide some use to educators. But to diagnose the individual progress, the learning analytics data collection must allow the identification of a particular student. In contrast, most schools will not be able to provide the required infrastructure, such as a *Learning Record Store*. Thus, it is required to combine centralized logging of pseudonymized interaction data in combination with local depseudonymization with a dashboard showing progress as close to real-time as possible. Additionally, this setup might profit from a feature enabling the teacher to remotely check student screens and supervise their activity.

[1] Usually mouse input will suffice for interaction

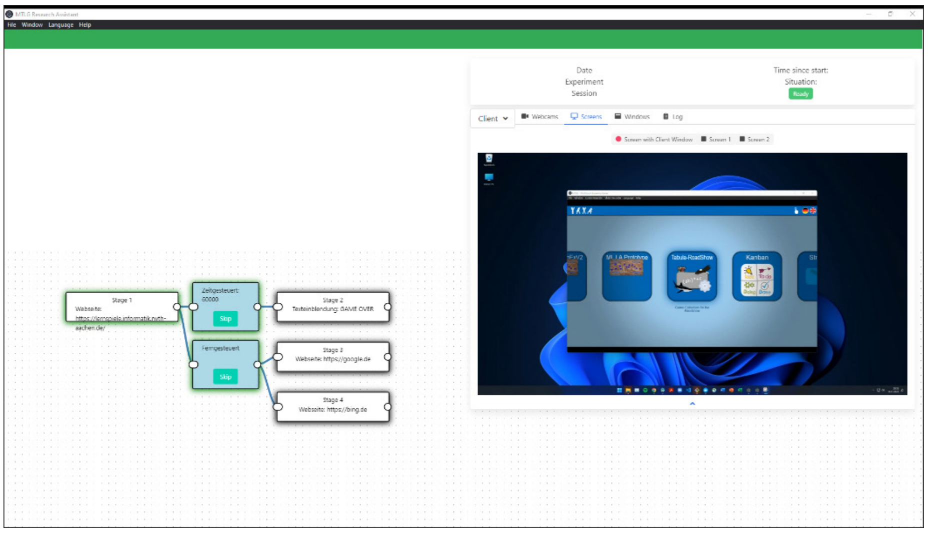


Fig. 2. (Remote) Supervising of experiments in the research assistant

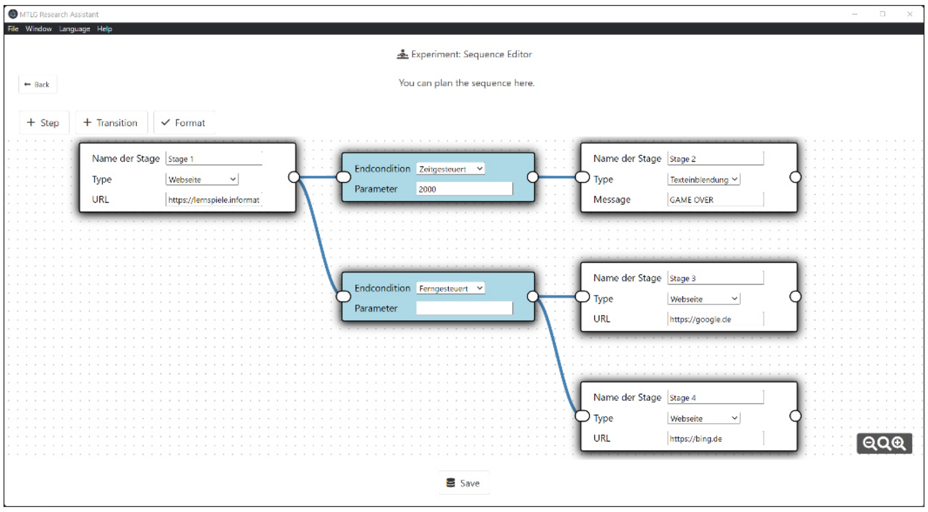
Full-Scale Lab Experiment (synchronous on-site): A controlled user study most likely takes place in a controlled environment like a university learning lab, usually centered around collaborative hardware like an interactive table-top system. The experiment follows a predefined protocol and is conducted by an experimental supervisor. This protocol might include analog activities, e.g., perceptional speed tests with pen and paper. We have also considered: a mixed setup including individual client devices, e.g., for surveys in pre-and post-tests, a multi-modal lab environment with additional sensors and cameras synchronized with the protocol, efficient switches in both sequence and parameters for test and control groups, both predefined and supervised on-site. And lastly, we considered adaptive learning based on learner models.

Individual, BYOD, and online (synchronous off-site, "remote/virtual lab"): In cases where hardware and specific devices do not matter too much, a bring-your-own-device approach could generate more participants. It might even be possible to conduct an impromptu experiment in a crowded lecture hall. One requirement for this case is providing a preconfigured package with a low threshold to participate: No installation required, no URLs to type, and little room for mistakes. An enforced sequence would ensure participation in a pre-test and might even provide tokens as the last part of the sequence as proof of participation without any conflicts in pseudonymity. All data is collected online automatically.

Individual at home on provided device (asynchronous off-site, "on-demand remote lab"): For this the experiment is bundled up, all data is collected on the device, and post-processing is done as soon as the devices are returned (or online again).

Based on those edge cases and the previously discussed criteria of good scientific practice and FAIR data, the following requirements are derived and assigned to a phase of a single research iteration: preparation, conduction, or post-processing.

1. Experiment Preparation: The preparation of an experiment consists of four different steps. The first is the creation of the experiment itself and the acquisition of metadata. Required are title, description, and contact info. Optional are research background, the context of the experiment, and other. The second step defines scope and scale: Is this a collaboration study or an individual? Online or offline? Is the number of sessions assessable? Is there a control group in addition to the test group? Shall the system randomize the allocation of sessions to the control and test groups or is it done manually? The third step depends on the defined scope of step 2 and gathers information on the (physical) composition of the lab environment: Which clients are used? Which data sources are available? Is screen content and camera feed to be recorded? The fourth step composes the content of the survey. Sequences are to be created and assigned to the client devices. There might be multiple sequences differing in order or parameters for the test and control groups, see Figure 3 as example.
2. Experiment Conduction: The MTLGresearch toolset must provide solutions to orchestrate synchronous and asynchronous experimental setups. In synchronous conditions, like a supervised lab experiment, the conductor should be able to initiate the experiment, (noninvasively) supervise progress, and trigger stage transitions manually, i.e., after giving instructions. Nevertheless, the experiment should be automated as far as possible and should be able to trigger conditionally without intervention. Recording should be orchestrated as well as the logging of all events and collecting information to synchronize all data streams in post-processing. Both loading and parameterization/configuration of online and offline content should be automated without intervention to avoid human error. Integration of an annotation tool for supervisor observations or diversions from the protocol is mandatory.



Experiment: Sequence Editor

You can plan the sequence here.

Name der Stage: Stage 1
Type: Webseite
URL: https://lernspiele.informat

Endcondition: Zeitgesteuert
Parameter: 2000

Name der Stage: Stage 2
Type: Texteinblendung
Message: GAME OVER

Endcondition: Ferngesteuert
Parameter:

Name der Stage: Stage 3
Type: Webseite
URL: https://google.de

Name der Stage: Stage 4
Type: Webseite
URL: https://bing.de

Fig. 3. The planning of a non-linear experiment sequence

3. Post-Processing: The first and foremost task of the post-processing stage is to get all recorded data from the various sources involved delivered to the responsible

researcher in a usable format. This includes screencasts, video recordings, audio recordings, interaction data, and event logs, along with available metadata and a protocol of the session. A modular approach is taken for further integration of steps like manual analysis of recordings, i.e., to generate additional interaction data beyond automation. The final step of the post-processing is assistance for open data publication. This helps the experimenter to upload all aggregated data as well as the metadata from step 1 along with these sequences and tools to a connected, open data portal.

Three components are implemented to cover all required steps and use cases: A dedicated client as an interface to the participant, a research assistant toolkit for planning and supervision, and the orchestrator, connecting all of those.

3.1 The MTLGclient

Officially, the Touch Events API is supported by all major browsers[2] as shown on the Browser Compatibility Page from the Mozilla Developer Network website [19]. Nevertheless, usage is not as easy as it might be expected, i.e., Chrome does not support touch events out of the box. From version 71 on, touch events had to be enabled by setting a flag inside Chrome's settings page, from version 78 (on Windows), this option disappeared. Instead, it was required to start Chrome with an additional command-line argument "--touch-events" to forward touch events to the web application[3].

This is an inconvenience for developers but might be a major obstacle for other users. In school environments, there is often a strict role management system in place, forbidding students (and teachers) to alter such settings or modify links to applications. Furthermore, it would be a serious threshold for teachers to use such web applications if they must set up all individual student systems in advance. Researchers with a lower affinity for technology intending to use multi-touch learning applications for their scientific projects could be repelled by requiring such "hacky" modifications.

Those arguments should already suffice for putting this issue on the agenda for developing a sustainable ecosystem for real-world usage as well as interdisciplinary research, but there are other shortcomings that could be addressed by a possible solution as well. In principle, it is possible to access screen content and recording equipment from within a website, but it requires additional user interaction and might provide different problems if restrictions in browser settings are in place.

The best potential solution is an application that can be bundled up, already set up with all required parameters, that can be executed stand-alone without previous installation. Fortunately, there is an application development toolset providing those capabilities. The Electron Project is a framework that bundles up a chromium core with NodeJS within a native application for all desktop operating systems.

Multi-touch support within the Electron application is straightforward, which also bundles a specific version of chromium and NodeJS. This ensures backward

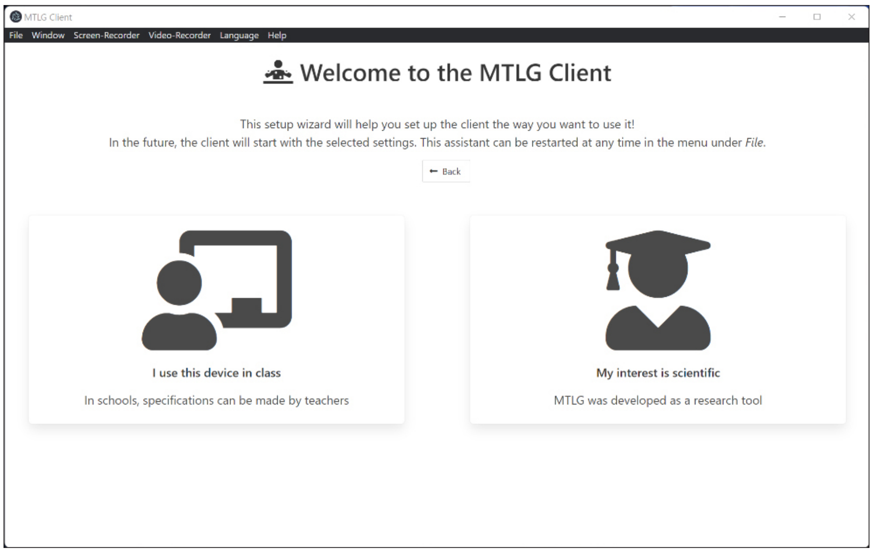
[2] Safari excluded. MacOS has no native touch support at the time of writing

[3] https://support.google.com/chrome/thread/23399752/what-happened-to-touch-events-api-under-chrome-flags?hl=en

compatibility, so even if future versions move the flag again, it is still possible to compile and get a working version as long as the packages remain in NPM. Furthermore, the compiled application works stand-alone and does not require installation, but it is also possible to build an installable package if desired. For a research tool, those advantages outweigh the criticism of a large and resource-intensive bundle. Furthermore, including possibly outdated browser components requires a certain degree of responsibility and consideration of the combined bundles, sequences, and targeted web resources.

While the tool might still suffer backlashes by changes in package availability in the future, the actual research prototype, i.e., the learning application, is completely decoupled from the MTLGclient application. So even if a future change might possibly hinder compiling, any provided compiled version will still support the development of new learning applications for interactive table-top displays in the future. It will also ensure the ability to reproduce any existing experimental setups. This ensures sustainability, as games and executables can be bundled with research data. The implemented application, called the MTLGclient, has access to screens, windows, and cameras through the WebAPI and is able, due to the integrated NodeJS core, to access the host machine's file system. This allows the synchronized recording for later post-processing.

Additionally, the application includes a setup wizard for easy configuration, as shown in Figure 4. The configurable features include an automated startup in full-screen mode as well as locking the settings, the import of packaged games for local execution as well as the displaying of remote content. Furthermore, the MTLGclient offers the definition of content sequences and injection of parameters, which will both be discussed in detail later. Through the combination of those features, it is possible to define several use cases for this application in accordance with the formerly described settings. On the first start, a wizard guides the user through the selection and configuration process for the six possible modes: "Just Playing", "School Usage", "Lab-based collaboration research", "Additional Lab Device", "Sync remote", and "Async Remote".


MTLG Client
File Window Screen-Recorder Video-Recorder Language Help
Back
I use this device in class
My interest is scientific

Welcome to the MTLG Client

This setup wizard will help you set up the client the way you want to use it!
In the future, the client will start with the selected settings. This assistant can be restarted at any time in the menu under *File*.

In schools, specifications can be made by teachers

MTLG was developed as a research tool

Fig. 4. Setup wizard for the client software

The most relevant modes for lab-based research are the third and fourth. Those modes are intended for controlled experiments in a research lab setup. Those experiments often require more than just the learning game running. Some post-processing like gesture or posture detection can benefit from calibration patterns to locate the screen within the camera footage, sometimes it is necessary to provide instruction prior to the game. To enable the controlled processing of such a sequence, the client is not to be configured with a game package or URI, but with the address of an additional component of the MTLGresearch toolset, the MTLGorchestrator. Additionally, it takes an argument to identify itself to the orchestrator, i.e., the room number, the device type, or both, for example, "InteractiveTableTop-LTIlab".

The additional device mode is integrated to enhance the setups described in the third mode. It does not differ much in contrast to that mode, as it also connects to an orchestrator instance and is capable to receive and execute sequences. A possible application scenario is a study that relies on the participants taking part in individual surveys like pre- and post-tests. Through the integration within the orchestrator and the usage of components for user modeling, it is possible to generate learner models, which will be made available instantly to the application. The orchestrator allows the automated injection of pseudonyms both in the surveys as well as into the multi-touch application, improving the user experience in post-processing and opening new possibilities for adaptive collaborative learning. The configuration interface provides an additional input field in comparison to mode 3, asking for a group name to orchestrate individual steps in the experimental sequence on all clients of a group at once. Another possible application would be an observer group, devices connected to recording equipment like cameras to take that resource load from the central device.

A major requirement for controlled experiments is the execution of a predefined protocol. In the MTLGclient, this was considered by the implementation of sequences. Sequences consist of stages, which differ from each other in two major aspects: content and end condition. The four **stage content types** currently included are *message*, *package*, *URL*, and *file*. *Message* conveys simple information like "Listen carefully to the instructions provided by the supervisor" to the users. This enables researchers to structure sequences without writing HTML code. The *package* type names a package previously imported into the client, and *URL* points to an online resource. *File* allows the integration of local resources provided with the client. Currently, there are a checkerboard pattern and a ChArUco pattern[4], the most common calibration pattern for OpenCV applications. Both are included within the predefined sequences for reference.

The next aspect of sequencing is the **transition** between stages. The experiences from case studies, student projects, and literature reviews showed four trigger mechanics for transitions: *Timing*, *content-trigger*, *event-trigger*, and *server-messages*. The *time* trigger might be used to limit the time spent within the stage/learning application. Another likely usage is shortly displaying the calibration patterns mentioned above before automatically proceeding to the next stage. The *content* trigger serves the purpose of integrating external tools with limited access to their functionality. For example, it is possible to integrate any survey tool within a sequence and display the text "Thank you for participating, please wait for further instructions" on the last page.

[4] https://docs.opencv.org/master/df/d4a/tutorial_charuco_detection.html

Set on *content-trigger*, the software parses the document object model within the web view for the string provided and will then proceed and report progress to the orchestrator. Consequently, the orchestrator can detect the completion of a pre-test by the group, proceed with the next stage, and notify the instructor. *Event-trigger* aims at applications developed specifically for the usage with the MTLGclient. The client injects a callable function into any website loaded. By calling mtlgClientBridge.stageFinished(), the contained website notifies the client to proceed, i.e., if all levels of a learning game are completed. The last transition mechanics are server messages, which wait for an event triggered by the orchestrator before proceeding with the next stage. This can be either initiated by the last participant completing a pre-test or by a supervisor's click after giving all required instructions.

An essential aspect of the MTLGclient capabilities is the injection of parameters in the sequence data structure. While this has little usage for external content, the MTLGcore will look for any configuration stored within the mtlgClient object. If present, those settings will override online configuration (if hosted publicly) or package configuration files (if a package is provided) and the default values.

This feature opens a few advanced possibilities: Learning games can be hosted online for public usage, avoiding many different circulating versions, and still set up, i.e., a specific learning record store or session ids, or user identification strategies while still having a central instance for bug fixes or extensions, thus delivering public updates instantly while still maintaining private configuration for the local setup. Furthermore, this opens new possibilities for comparative studies: Sequences can not only differ in their order for test and control groups. One could define sequences with different parameter sets for both groups, i.e., enabling or disabling adaptive feedback. The supervisor must just load the correct experiment file to send the proper sequence.

Research assistant. The Research Assistant tool is a component providing support to experiment planners and supervisors and offers the possibility to add post-processing capabilities via additional modules in the future. It is built on the same software stack as the client to increase maintainability. It assists in planning, conduction, and post-processing.

During the **planning stage**, the Research Assistant manages experiments in files. Those files include the specification for the experimental setup as a nested data structure in the JSON format. This approach was chosen over an integrated database for portability reasons: The experiment can be planned and prepared on the researcher's desktop PC and later transferred to the supervisor's machine or distributed by mail, and most important stored along with the research data as metadata as it includes valuable information on context and parameters.

The JSON format also offers a high degree of flexibility: Later extensions of additional metadata are easy and a non-breaking change. While currently there is just a basic set of meta-information included, the file is intended to store, i.e., contact information for responsible scientists, supervisors, or additional context data. Since this project provides a sustainable, extensible ecosystem, there is no need for completeness in the provided software; instead, the focus of implementation was to keep the specification as open as possible and avoid API changes, as those can lead to ripple effects [20].

As such, it is currently possible to set up the experimental mode and modify the sequence, i.e., in order to create experiment definition files for test and control groups as a forking, non-linear sequence in the editor shown in Figure 3. Here it is possible to adjust the order of the sequence or modify the individual stages regarding the aspects discussed in the Client section. If a valid orchestrator URL and sequence are given, the experiment can be started.

The **supervision stage** begins as soon as the *Start Experiment* button is clicked. As the first step, the connection to the orchestrator is to be initialized by the click of a button. After the login, an initial fetch of connected clients is performed and can be repeated if the required clients are not connected yet. A dropdown selection menu in the configuration tile allows the selection of the main device to run the experiment sequence on. Succeeding the confirmation, the sequence is parsed by the stage, so the supervisor is provided with a comprehensive overview of the experimental sequence. On the left two-thirds of the screen is the detailed view of the sequence. Each step is represented with a comprehensive stage card, including the description coded in the sequence along with further information on the transition conditions. If the stage is to be concluded by the supervisor, a button is displayed to confirm that the required instruction is given or that different requirements for progress are met.

The right side of the screen is again divided into two different parts: On the upper part, a summary of the current state is given. Below are additional tools that are especially useful for setting up the experiment or supervising remotely. After connecting with the orchestrator, a list of connected clients is fetched and included here. Upon selection of a client, all available data sources are requested. Currently, those are clustered in three tabs: Cameras, Screens, and Windows. A click on any of these sources will request the client to start a peer-to-peer stream to the Research Assistant. This helps to align cameras in advance to fit the required perspective or check the screen content remotely to supervise experiments unobtrusively. The list always includes all available cameras, the screens connected to the device, and all available windows. For ease of use, the software always includes the options *screen with client window* and *client window* at the beginning of the respective list for convenience. This "sneak peek" into the media streams is also available during recording without interfering with the process like shown in Figure 2. Due to the usage of WebRTC as the base technology for that feature, the internet uplink is (in most standard scenarios) not occupied by this. Experience shows that the LRS connection alone can require bandwidth up to 5 MBit/s[5]. To avoid this bottleneck and still provide those convenient features, peer-to-peer connections seemed the obvious choice, since most often Client and Research Assistant share the same network.

The **post-processing stage** is still conceptual as it refers to future extensions of the toolset beyond the MTLG ecosystem. The current handling of research data in prototypes includes the automated direction to sophisticated tools and open-source systems, like full-featured learning record stores (i.e., Learning Locker) or survey systems (i.e., LimeSurvey). A proof-of-concept tool for automated extraction and upload to open data portals like CKAN has been implemented and tested. Beyond the data aggregation and transfer, additional stages can be included as well, like verification of interactions or videographic post-processing of the recorded streams.

[5] Measured during a roadshow event as the demonstrator showed serious performance issues due to a congested uplink

Orchestration. A common approach in university laboratories is the installation of a local network infrastructure, to some degree isolated and thus self-reliant. While this eliminates the "single point of failure" of a server-centric infrastructure, it still comes with serious downsides: It is hard to manage, add new devices, communicate with external services, and makes remote maintenance and supervision difficult. A further issue often arises when mobile devices are added: Eduroam, the WiFi service of most European universities, has client isolation enabled and thus prohibits direct communication between devices. The provision of own wireless network opens potential security risks and is often explicitly forbidden.

A small, robust central component can circumvent most of those issues. Beyond signaling between research assistant and client, such a persistent service makes infrastructure discovery and device handshakes simpler, offers central logging capabilities, and enables remote access to the experimental setup in all stages. To avoid a potential bottleneck, this component also manages the signaling for the WebRTC protocol, thus making use of (local) peer-to-peer media stream and file transfer in on-site, synchronous setups, making use of an automatic remote-controlled separate STUN/TURN server on a different machine as an integrated fallback. Along with the log, the orchestrator monitors the connection quality and system time differences between connected computers, thus allowing for precise post-factum synchronization of events. Furthermore, the orchestrator is, beyond logging on a per-session basis, a stateless, signaling-only component with very light traffic requirements, thus it can serve many concurrent experiments. To ensure proper isolation, in that case, a rudimentary authentication system based on virtual rooms and authorization tokens is integrated.

4 Open tools for open science

All described tools are still work in progress and probably will be forever. As they are released open source, there will always be new features to implement and new use cases to cover. Still, in the spirit of good scientific practice, the tools can be considered sufficiently complete to use. The idea is to provide a set of tools without complicated external dependencies, so they can be archived alongside the data recorded with them.

5 Conclusion & outlook

The ecosystem presented in this paper was originally intended to serve one purpose: Researching collaborative learning with interactive table-top displays. It has since then grown far beyond that: Due to its modular nature, its FAIR & Open Data-first approach, and the use of sustainable, interoperable technologies, it can now be used as a general-purpose framework for the effective research of lab-based learning. While the focus lies on the modern, interconnected on-site lab setup, it also serves hybrid and remote configurations as well, both in synchronous and asynchronous procedures. It already incorporates multi-modal sources and synchronizes event streams, and thus enables new insights into collaborative learning processes.

Beyond the current use cases, future extensions are already in planning: Complex, non-linear experiments and learning applications, integration of new digital media like VR headsets or wearables by providing (user-)interfaces through their APIs, and the extension of the learning analytics concept by utilizing state of the art machine-learning and computer vision technologies. And since all is developed open source, there is always the possibility to contribute, in case there are missing features for a promising new lab-based scenario.

6 Acknowledgement

The authors gratefully acknowledge the funding by the Federal Ministry of Education and Research (BMBF) for the project LeBiAC2 (grant no. 01JA1813). The responsibility for the content of this publication lies with the authors.

7 References

[1] B. F. Skinner, *About behaviorism*, 1st ed. New York: Knopf and [distributed by Random House], 1974.
[2] R. M. Yerkes, and S. Morgulis, "The method of pawlow in animal psychology," *Psychological Bulletin*, vol. 6, no. 8, pp. 257–273, 1909. https://doi.org/10.1037/h0070886
[3] Š. Hošková-Mayerová, and Z. Rosická, "Programmed learning," *Procedia– Social and Behavioral Sciences*, vol. 31, pp. 782–787, Jan. 2012. https://doi.org/10.1016/j.sbspro.2011.12.141
[4] T. Warnecke, P. Dohrmann, A. Jürgens, A. Rausch, and N. Pinkwart, "Collaborative learning through cooperative design using a multitouch table," in *Cooperative Design, Visualization, and Engineering*, vol. 6874, Y. Luo, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 22–29, 2011. https://doi.org/10.1007/978-3-642-23734-8_4
[5] N. Bergner, M. Ehlenz, and U. Schroeder, "Kooperatives e-learning am multitouch-display für grundschulkinder," presented at the 7. Münsteraner Workshop zur Schulinformatik, Münster, May. 2016.
[6] S. Voelker *et al.*, "PERCs: Persistently trackable tangibles on capacitive multi-touch displays," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, Charlotte NC USA, pp. 351–356, Nov. 2015. https://doi.org/10.1145/2807442.2807466
[7] M. Lungu, "Reverse engineering software ecosystems," PhD Thesis, Faculty of Informatics, University of Lugano, Switzerland, 2009.
[8] W. Greller, M. Ebner, and M. Schön, "Learning analytics: From theory to practice—data support for learning and teaching," in *Computer Assisted Assessment. Research into E-Assessment*, Cham, pp. 79–87, 2014. https://doi.org/10.1007/978-3-319-08657-6_8
[9] M. Ehlenz, N. Bergner, and U. Schroeder, "Synergieeffekte zwischen Fach- und Lehramtsstudierenden in Softwarepraktika," in *Hochschuldidaktik der Informatik : HDI2016; Universität Potsdam / Andreas Schwill, Ulrike Lucke (Hrsg.)*, Potsdam, vol. 10, pp. 99–102, 2016.
[10] J. A. Mesa, I. Esparragoza, and H. Maury, "Modular architecture principles – MAPs: A key factor in the development of sustainable open architecture products," *International Journal of Sustainable Engineering*, vol. 13, no. 2, pp. 108–122, 2020. https://doi.org/10.1080/19397038.2019.1634157

[11] T. Richardson, L. Burd, and S. Smith, "Guidelines for supporting real-time multi-touch applications," *Software: Practice and Experience*, vol. 44, no. 8, pp. 931–949, 2014. https://doi.org/10.1002/spe.2183
[12] J. Derboven, D. De Roeck, and M. Verstraete, "Semiotic analysis of multi-touch interface design: The MuTable case study," *International Journal of Human-Computer Studies*, vol. 70, no. 10, pp. 714–728, Oct. 2012. https://doi.org/10.1016/j.ijhcs.2012.05.005
[13] E. Mercier, "Device ecologies to support collaborative learning in classrooms," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*, pp. 217–218, May. 2014. https://doi.org/10.1109/CTS.2014.6867567
[14] R. Martínez, A. Collins, J. Kay, and K. Yacef, "Who did what? Who said that?: Collaid: An environment for capturing traces of collaborative learning at the tabletop," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, New York, NY, USA, pp. 172–181, 2011. https://doi.org/10.1145/2076354.2076387
[15] S. Berndt, "Konzeption und implementierung generischer spiele-frameworks für multi-touch-systeme," Diplomarbeit, Universität Potsdam, Potsdam, Germany, 2015.
[16] A. Hassan, and N. Pinkwart, "On the adaptability and applicability of multi-touch user interfaces addressing behavioral interventions for children with autism," *IETE Technical Review*, 2019. https://doi.org/10.1080/02564602.2019.1590164
[17] T. Zarraonandía, P. Díaz, and I. Aedo, "Modeling games for adaptive and personalized learning," in *The Future of Ubiquitous Learning: Learning Designs for Emerging Pedagogies*, B. Gros, Kinshuk, and M. Maina, Eds. Berlin, Heidelberg: Springer, pp. 217–239, 2016. https://doi.org/10.1007/978-3-662-47724-3_12
[18] M. Ehlenz, T. Leonhardt, C. Cherek, W. S. Wilkowska, and U. Schroeder, *The Lone Wolf Dies, the Pack Survives?: Analyzing a Computer Science Learning Application on a Multitouch-Tabletop*. New York, NY, USA: ACM, 2018. https://doi.org/10.1145/3279720.3279724
[19] Mozilla Developer Network, "Touch events API," *MDN Web Docs*, 2021. https://developer.mozilla.org/en-US/docs/Web/API/Touch_events (accessed Sep. 13, 2021).
[20] R. Robbes, M. Lungu, and D. Röthlisberger, "How do developers react to API deprecation? the case of a smalltalk ecosystem," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, New York, NY, USA, pp. 1–11, Nov. 2012. https://doi.org/10.1145/2393596.2393662

8 Authors

Matthias Ehlenz is a Ph.D. candidate at the RWTH Aachen University. He graduated with a computer science and chemistry teaching degree, followed by a researcher position in the Learning Technologies Research Group (LuFG i9). He is currently working at the Teacher Training Center of RWTH Aachen University, coordinating the recently established MediaLab for digital education. His research interests focus on the practical effects of innovative media on education, especially collaborative learning with interactive table-top displays.

Birte Heinemann studied Computer Science with a minor in psychology at Paderborn University and stayed for three years in the Computer Science Education Department before she joined the Learning Technologies Research Group at the RWTH Aachen University as a Ph.D. student. She loves developing and researching learning technologies

and teaching computer science. Her actual research focuses on infrastructures for multi-modal Learning Analytics and Virtual Reality in interdisciplinary projects.

Ulrik Schroeder studied Computer Science (CS) at TU Darmstadt, where he received his doctorate in software engineering. After postdoctoral research at the Center for Lifelong Learning and Design (L^{3}D) at the University of Colorado at Boulder and a substitute professorship at TU Darmstadt in 1999, he was appointed to professor for CS and its didactics at the PH Ludwigsburg in 2000. Since 2002, he has been head of the teaching and research group for learning technologies and didactics of computer science at RWTH Aachen University. His research focuses on learning technologies and learning analytics. He is a member of numerous academic committees, a reviewer and program committee member for various conferences and journals, and the director of the InfoSphere student lab.

Article submitted 2022-09-01. Resubmitted 2022-10-09. Final acceptance 2022-10-15. Final version published as submitted by the authors.