A Framework for Incorporating Serious Games into Learning Object Repositories through Experiential Learning

Abhishek Parakh
Cybersecurity
University of Nebraska at Omaha
aparakh@unomaha.edu

Mahadevan Subramaniam
Computer Science
University of Nebraska at Omaha
msubramaniam@unomaha.edu

Parvathi Chundi
Computer Science
University of Nebraska at Omaha
pchundi@unomaha.edu

Abstract

A learning environment Galore *seamlessly combining serious games with generative learning objects based on an experiential learning model is described. The learning object repository in* Galore *allows domain experts to elucidate knowledge concepts using parameterized learning objects in diverse formats such as text, visuals, interactive widgets, as well as embedded* gamelets *to provide a learning experience customized to student learning preferences. Reflective observation and abstract conceptualization zones are explicitly modeled to support the exploratory phases of the experiential learning model. The metadata associated with the knowledge components in the environment are used to develop algorithms that automatically retrieve and synthesize lesson plans for students to achieve a specified set of learning objectives. We illustrate* Galore *through a case study for secure quantum internet protocols using Python Jupyter notebooks. A total of 28 notebooks and 16 gamelets using* Galore *cover the foundations of secure quantum internet protocols.*

1. Introduction

Serious games and e-learning learning object repositories (LORs) have achieved impressive successes in enabling digital learning in diverse domains including information systems, management, medicine, and other STEM areas [1, 2, 3, 4, 5, 6]. By focusing on engagement through reflexive player interactions, serious games often manage to create a state of flow among its players providing them with concrete and active experimentation opportunities [6].

On the other hand, in LORs [2, 3, 7], the emphasis is less on engagement but more on elucidating concepts through a variety of learning objects represented using videos, texts, interactive widgets, codingbats, along with assessment instruments such as quizzes, tests and exercises. Typically, the learning objects in LORs can be customized to suit student learning preferences [8, 9, 10] and potentially studied through deliberate and intentional user interactions that enable students to contemplate about concepts, synthesize and interweave concept dependencies.

While incorporating deliberate and intentional user interactions in a game-based setting without disrupting a state of flow tends to be difficult, we can potentially extend LORs by incorporating serious games to combine the benefits of serious games with those of the LORs. Such extended e-learning platforms can also be a step towards integrating game-based learning into classrooms which has great potential in providing an engaging and comprehensive learning environment for students.

In this paper, we present a novel learning environment, (*Galore*), an extended e-learning platform which seamlessly combines serious games with e-learning LORs. The framework in *Galore* is based on the foundational principles underlying Kolb's four-phase model for experiential learning [11]. Serious games encapsulated as learning objects, called (*gamelets*), and a variety of regular e-learning objects are used in *Galore* to realize the concrete and active experimentation phases of Kolb's learning model. Reflective observation, and abstract conceptualization phases of the model are built on top of these learning objects to realize the comprehensive experiential learning model in *Galore*. The knowledge concepts in *Galore* are organized as a set of *units* each of which is a collection of *modules*, *micro-modules* and *nano-modules* based on how much time it takes to complete these. Groups of gamelets and regular e-learning objects form each module and micro-module.

All the components in *Galore* from the units down to the learning objects are parameterized entities similar to generative learning objects [12, 13, 14]. Parameters including anticipated student learning outcomes, supported learning preferences, pre-requisites, and other learning attributes such as levels of hints and scaffolding, etc. can be instantiated

URI: https://hdl.handle.net/10125/79978
978-0-9981331-5-7

using a range of permitted values to create a variety of learning experiences for students. A customized family of lesson plans is automatically synthesized corresponding to each assignment of parameter values. A student session in *Galore* typically starts by exploring a lesson plan chosen from the available family of plans and transitioning to other plans in the family based on their performance and learning preferences. A student learning experience with a family of lesson plans is completed when all the learning outcomes specified in the parameters are successfully satisfied. These can be achieved by students using one or more sessions.

Lesson plans are presented to students as a collection of Python Jupyter notebooks where each cell in each notebook corresponds to an instantiated learning object. In some cases, a collection of cells in the Jupyter notebook maybe necessary to fully instantiate a learning object. Cells in Jupyter notebooks are of two types - code cells and markdown cells. Code cells are used to embed Python code that, when run, results in an output, interactive terminal for the user, animation, interactive graphical elements, etc. in the cell following the code cell. The markdown cells are essentially webpapge elements and be used to embed other types of learning objects such as text, visuals, animations, videos, gamelets and so on.

Associated with each lesson plan are three graphs. Two of these graphs depict the concept and learning outcome dependencies among the learning objects. The third graph is the student interaction graph in which each node is a learning object, and the edges represent the next learning object in the lesson plan that the student may interact with after working with a learning object. Student traversals of the interaction graph are constrained by the concept and the learning outcome dependencies as well as by the performance of the student. *Galore* provides different views of the interaction graph to support goal-oriented and concept-oriented learning preferences. *Galore* will automatically choose another member from the family of lesson plans if the current selection does not result in an optimal learning experience for a student.

In this paper, we illustrate the *Galore* framework with a non-trivial application involving quantum secure internet protocols [15]. Quantum internet based on quantum key exchange protocols are an emerging technology that empower secure exchange of information against quantum powered adversaries. Secure quantum internet protocols require students to be proficient in several diverse and intricate concepts related to quantum computing principles, secure key exchange protocols, and routing approaches in computer networking. Learning such a diverse set of topics using traditional pedagogical methods is a challenge especially given the scarcity of textbooks covering such a wide array of emerging topics. In order to illustrate the potential of *Galore* in effectively addressing these challenges, we develop parameterized e-learning and gamelet learning objects to model the fundamental quantum computing principles including programming and measurement of qubits, qubit entanglement and qubit decoherence. A gamelet object is used to create a play scenario where students can establish an entangled pair of qubit states over two endpoints of a network modeled as small $N \times N$ square grid. The e-learning and gamelet objects are designed to support concrete and active experimentation phases of the experiential learning model. We also model reflective observation and abstract conceptualization phases of the model as dedicated zones and tightly integrate the four phases to illustrate how the Galore framework can provide a comprehensive and engaging experience for students to learn about design and development of quantum internet protocols.

To the best of our knowledge, *Galore* is the first extended e-learning framework that combines multi-modal, e-learning objects with gamelet learning objects based on experiential learning for quantum computing based security protocols and quantum secure internet. *Galore* is a general-purpose learning framework and can be adapted to create a customizable learning environment embedding serious games and multi-modal learning objects.

The rest of this paper is organized as follows. After a brief review of the related works, we describe the overall architecture of *Galore* in Section 2. Algorithms for synthesizing customized well-formed lesson plans, incorporating learner preferences, and those for performance-based exploration in student sessions are described in Section 3. Section 4 illustrates our approach using a case study involving quantum internet protocols. Section 5 presents conclusions and future work.

1.1. Related Works

E-learning frameworks using LORs have been extensively studied in the literature. Adaptive web-based lesson plan generation has a long and rich history in e-learning. Rule-based frameworks have been extensively investigated by several earlier works to automatically synthesize course books [10, 8] for a given set of learning concepts, scenarios, and pedagogical goals. Large learning object repositories parameterized by concepts and learner preferences and backgrounds have been developed by several works [2,

14, 3, 16] for computer science education. Informally, a generative learning object denotes a family of related learning object instances that can be automatically generated on demand by instantiating the associated parameters. While the learning object review instrument [17] to evaluate these repositories includes adaptation as one of its nine criteria, they provide limited support ([14] is an exception) for adaptivity as compared to the rule-based frameworks.

There has been extensive research on games and learning dates back several years [18, 19, 20, **?**], integrating games into formal education contexts such as classrooms, online classes, projects and assignments. There has been a recent resurgence in game-based teaching involving serious games[21, 22, 23, 24]. In [22], Ketamo et al argue that it is important to focus on teachers in game-based learning and investigate teacher models in addition to learner models that are commonly used to analyze the effectiveness of serious games. They consider several games such as the AnimalClass, Eedu Elements, Media Detective, and ALICE FireEvacuation supporting varied teacher models including those supported by flipped classrooms, learners as teachers and teachers as active rivals in the game to show that the incorporating teacher models into games leads to significant improvement in learning. In [23], Thorkild, develop a framework that establishes a correspondence between game scenarios and the traditional curricular and pedagogical practices and describe how the teachers can move between game-based and curricular interactions to incorporate games into traditional instruction. They define instructor, evaluator, playmaker, and guide as the four roles that the teachers may choose to move between their different modes of interactions. In [24], de Freitas et al investigate several empirical studies involving educational games. They conclude that to realize the full potential of game-based education, a tighter and fine-grained integration of pedagogical approaches employed in formal educational contexts with those underlying games, facilitated by various instructional roles is needed. Most of these approaches have focused on how games can be used in classrooms and/or the role of teachers in navigating the pedagogical models underlying traditional instruction with those underlying games. Surprisingly, scant attention has been devoted towards developing a comprehensive learning environment that integrates e-learning frameworks with serious games. In this sense, the work described in this paper is significantly different from all these earlier works. Our learning environment, *Galore*, combines e-learning platforms and serious games through generative learning objects to provide a customized learning experience while incorporating components that explicitly realize the experiential learning model. To the best of our knowledge *Galore* is the first such environment that has been applied for the instruction of secure quantum internet protocols.

2. Galore Architecture

The overall architecture of the Galore learning environment is depicted in the Figure 1. The *Galore* environment supports a three layered architecture whose bottom most layer consists of a parameterized, extended learning object repository consisting of regular e-learning and gamelet learning objects along with reflective observation and abstract conceptualization zones. The access to this repository is controlled by the mid-level layer consisting of retrieval, synthesis, and instrumentation engines. The top-level layer contains a Jupyter notebook-based UI along with user adaptivity engine. The adaptivity engine continually observes student interactions and performance and updates the notebooks on-the-fly so that the student can make progress towards the specified learning outcomes while having an engaged learning experience.

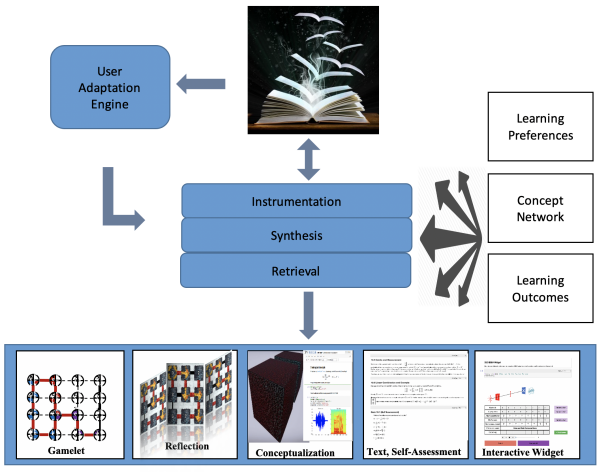


Figure 1: Galore Learning Environment Architecture

Each learning experience in *Galore* starts with a student logging into *Galore* and providing information about their background knowledge, and learning preferences. Based on a pre-assessment quiz, a set of learning objectives are presented to the student to choose from.

In the rest of this section, we describe extended e-learning repository of *Galore*. The algorithms for generating the family of lesson plans is described in the next section.

2.1. E-learning Objects

Informally, e-learning objects are like sections of a traditional textbook chapter that covers one or more knowledge concepts in detail. Analogous to a textbook, it is assumed that the students have the necessary prerequisites to learn the concepts covered in that particular learning object (section of the text) and also the next set of learning objects (sections of the text) that the student can attempt to learn after successfully completing a particular object. However, unlike a textbook section, each learning object is parameterized by several attributes. The *preference* parameter specifies the learning preferences supported in *Galore* for learning the associated concepts. Based on the preference parameter value, learning objects may elucidate the associated knowledge concepts in different ways to appeal to the learning preferences of the students. The preference parameter values supported by *Galore* include - *text, visuals (image), symbolic example, numeric example, widgets, simulation, code-IDE, code-IDE-wtests, quiz, final-quiz.* Setting the preference parameter to one of these values results in an instantiated learning object where the concepts are presented to a student as specified by the parameter value. For example, setting the preference parameter value to *text* will produce an instantiated learning object where the associated concepts are presented predominantly through textual descriptions, whereas setting the value, *visuals*, will present the concept using images to facilitate learning by visual learners.

Additional parameters for instantiating e-learning objects include prerequisites enforcement, completion, and list of learning outcomes. The prerequisite enforcement parameter values are represented by list of pairs whose first element is a knowledge concept and the second element defines the enforcement level. The prerequisite enforcement parameter is a categorical attribute whose values can be one of – *strict, conditional, or waivable*. The default prerequisite enforcement parameter value for a learning object is deemed to be *strict* meaning that the student must be proficient in the associated prerequisite concept needed to attempt that learning object. If the enforcement parameter value is *conditional* for a prerequisite concept in a learning object then the learning object can be attempted provided the student is proficient in all other non-waivable prerequisites. Prerequisites whose enforcement value is *waivable*, are optional.

The completion parameter values indicate the various assessment rubrics that determine the successful completion of a learning object. The parameter values could be *quiz, simulation, code-IDE, code-IDE-wtests, reflective observation, abstract conceptualization zones* and/or *final exam*. The completion of a learning object is assessed locally when the completion parameter value is *quiz* and or *simulation*. In all other cases, additional activities must be performed involving other learning objects to assess the completion of the concerned learning object. The assessments like a *code-IDE*, or c*ode-IDE-wtests* specified for completing a learning object are scheduled once the user is ready to be assessed for completion of that object. Assessments like *reflective observation* and *abstract conceptualization* and *final exams* are scheduled later in the student session after the student has gone through all related learning objects.

The list of learning outcomes specifies the proficiency attained by student on successful completion of the learning object. This is a non-configurable parameter and all instantiated learning objects inherit the same list of learning outcomes.

2.2. Gamelets

Each gamelet object is a miniature serious game (like snippets) consisting of fully interactive, 3D game scenario and incorporating all traditional features of a serious game. They are media rich environments supporting a variety of audio-visual elements such as narrations, and videos that could be delivered passively through the game elements in the scenarios or delivered actively through diverse avatars and oracles within a gamelet. Problems underlying the game scenarios are generated, instrumented to analyze player success and failure, and supported through various modes of hints, scaffolding, and adaptation [25, 26, 27]. Each has a clear *game play objective* (GPO) (which is different but related to student learning objective) and the progress of the player towards the GPO is measured through a scoring mechanism. Player(s) must achieve the GPO objective using the available resources whose utilization is measured in terms of the player health. Gamelets are distinct from e-learning objects. They are self-contained game scenarios designed using game engines such as Unity or Unreal (Galore gamelets are designed using Unreal) that can be launched from a Jupyter notebook cell. Gamelets conceptually differ from e-learning objects through the use of GPOs[1].

[1]Informally, gamification is a transformation that maps an SLO to a GPO and the achievement of an GPO coupled with abstraction leads to an SLO associated with any e-learning object. This mapping will be further discussed an expanded version of this paper.

2.3. Reflection Zones

Reflection zones are designed using game elements and each zone corresponds to exactly one gamelet. They are designed using game elements to mimic the associated gamelet in appearance, to stimulate reflection, but the user interaction supported are different from that in the gamelet. Student performance and utilization of resources are not quantitatively measured by scores, and health in these zones. In a reflection zone, students can repeatedly replay and observe their game play behavior in the associated gamelet. Students can also simulate the game play behavior in the associated gamelet by using a game-bot and observe its interactions that improve performance. The zone also supports student-initiated queries over the simulated game scenarios along with answers for a deeper comprehension and reflection of the concepts underlying the scenario in the gamelet. The two user interaction modes of replay and simulation are designed to stimulate student reflection triggering queries whose answers will aid in deeper conceptual understanding. Student learning in reflection zones can be captured in the *student knowledge cache* for future recall using a variety of audio-visual notes.

2.4. Conceptualization Zones

Conceptualization zones enable students to generalize and abstract the concepts learnt in the gamelets through gaming and reflection by fostering computational thinking. A conceptualization zone is usually mapped to one or more gamelets and the corresponding reflection zones. Abstraction and conceptualization are essential to apply the concepts learnt through gamelets and reflection zones to situations varying in scale, structure, and other attributes pertaining to the networks and quantum states. Conceptualization zones provide students to actively experiment with diverse applications involving the concepts learnt in the corresponding gamelets and reflection zones. A conceptualization zone can span multiple cells built using game elements and coding, and simulation visualization cells. The conceptualization cells with game elements present applications to the students as an interactive audio-visual game scenario and these cells also assist student queries with hints for solving the problem. The tasks in these zones are deliberately designed to be solved by coding and simulating code outputs using the coding and simulation visualization e-learning objects respectively.

3. Instrumentation, Retrieval and Synthesis

This section describes our procedure for generating a family of lesson plans using the learning object repositories for a given set of learning outcomes, student preferences, and backgrounds.

3.1. Instrumenting Learning Objects

To use the learning objects in *Galore*, students usually have to perform a keyword search to identify modules of interest and go through them in some chosen order. The *Galore* synthesis algorithm automatically retrieves the relevant learning objects from the repository and generates a family of lesson plans that the student can follow to achieve the desired learning outcomes. All the objects in the repository are instrumented with metadata obtained from domain experts describing the relations of each object to other objects, concepts, and outcomes along with the digital assets included in the object.

The metadata associated with each object consists of *local* and *global* data members in the object and those in the enclosing modules, respectively. The local data members include information such as the `object-id`, `object-concepts`, `object-outcomes`, and `object-prerequisites`, i.e., outcomes that must be met in order to use this learning object for study. The local data `object-type`, can be one of – *text, code, image, symbolic example, numeric example, video, widgets, quiz, auto-graded-simulation, auto-graded-quiz, code-IDE, code-IDE-wtests*. Users can write Python programs and test them using a given test suite in objects of type *code-IDE,* and *code-IDE-wtests*. The *numeric example, auto-graded-quiz, widget, code-IDE, code-IDE-wtests* objects allow users to interact by modifying the contents whereas *text, code, symbolic example, image, quiz*, and *video* cells allow more limited forms of user interaction. The Boolean valued local data `object-interactive` denotes whether an object is interactive or not. An object has to be interactive to be used to assess its learning outcome. The local data `object-alternates` provides a list of alternate objects that are semantically equivalent to the current object but with object-types different than the current object. Alternate objects are used to adapt to student preferences and for suggesting next objects based on student performance as described in the next section.

The global data members of an object include the containing `module-title`, `module-outcomes`,

and `module-prerequisites` i.e., other modules whose outcomes must be achieved in order to achieve the outcomes of the current module.

3.2. Generating Lesson Plans

The main steps to generate a family of lesson plans using learning objects are the following.

- **Pre-process the Instrumentation**: First, the consistency of the instrumentation is verified using properties such as: a) all global data in all objects within one module are identical, b) outcomes of all the objects of a module belong to the outcomes of that module, c) type checking of objects along with their alternates. Next, module and outcome dependency graphs are built from the object data. Lastly, these graphs are verified using properties such as: a) absence of dangling edges and cycles, b) module dependencies are a projection of outcome dependencies and that c) there exist one or more auto-graded final-quiz cells in each module that interactively evaluate all the outcomes of that module. The semantically verified modules and the graphs are then used to generate lesson plans. Note that the verification is a one time activity that needs to be performed at the creation and/or revision of the repository.

- **Retrieve Learning Objects/Modules**: Galore can either retrieve modules or individual objects from the repository. This choice is specified by the retrieval mode[2]. Given a set of outcomes, a retrieval mode, and a ranked list (of student preference parameter values), a family of relevant learning objects are identified. To build a lesson plan, modules whose outcomes include a given input outcome are identified. A reverse topological sort of the dependency graph starting with each of these modules (with ties among the modules being broken arbitrarily) produces a linear chain of modules which are then merged while obeying module dependencies to create a single chain of modules.

- **Synthesizing Lesson Plans**: The linear chain of modules obtained from the previous step are then organized into a hierarchy of nano-, micro- modules, units, and courses based on the estimated times to output a lesson plan whose successful completion is guaranteed to satisfy the input outcomes. In order to incorporate the student preferences in a lesson plan, for each module in the linear chain, the module is retained if its objects match the highest ranked input student preferences; otherwise, we replace it by a module with better matching, alternate objects, if they exist. Certain modules and objects within modules are pruned from a lesson plan based on the background knowledge of the student.

  We can use a similar procedure to generate a lesson plan based on individual objects instead of modules. In this case, the outcome dependency graph is used instead of the module dependency graph. This often, leads to more focused lesson plans with lesser number of objects taking lesser instruction time. Student background is used to prune objects and modules in this case as well.

[2]Galore supports additional retrieval modes in terms of downward completeness of prerequisites, assessment bundling and more.

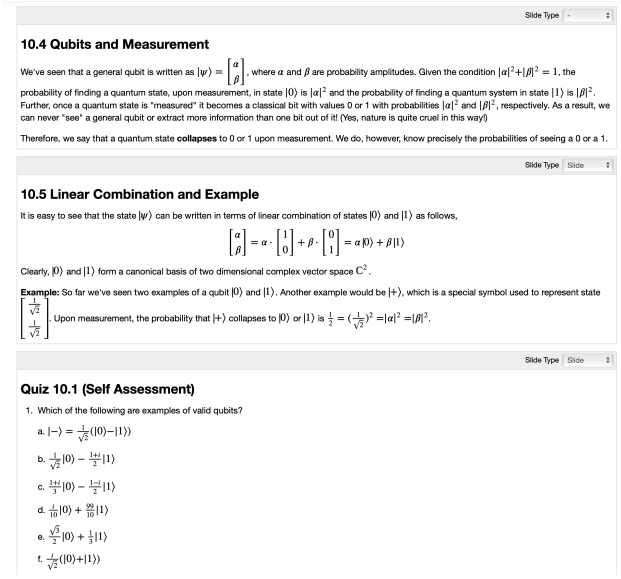
4. Quantum Internet Protocols

In this section, we describe a case study involving secure quantum internet protocols using the *Galore* environment. The e-learning and gamelet object repositories describing fundamental quantum computing and quantum cryptography principles are described. Reflective observation and abstract conceptualization zones are developed for quantum key distribution over a square quantum internet grid. Then, we illustrate how the algorithms described in the previous section can be used to synthesize and explore family of lesson plans.

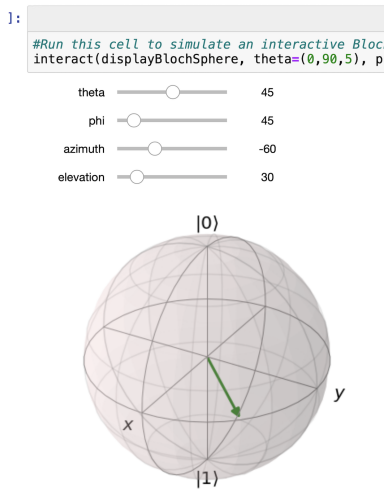
4.1. E-learning Objects for Quantum Internet

The e-learning objects of *Galore* quantum computing and cryptography are hosted on Clark, a digital library of learning objects developed by the authors through federal grants. The Clark library consists of learning objects contributed from over fifty universities across the United States on various topics in cybersecurity. The *Galore* collection of e-learning objects in Clark are available for free download.

The *Galore* e-learning objects are presented to students in the form of Jupyter notebooks. Jupyter notebooks have become the defacto standard for quantum programming as they allow for an easy integration of mathematics, code as well as visual objects that can then interface to external APIs (such as the IBM simulator or quantum computer) with minimal effort. Python was the chosen language for the development of these notebooks since Python is a popular language for quantum programming and well as the cybersecurity domain. While Jupyter notebooks provide two types of cells - *markdown and* code,



(a) Static cells with definitions, examples and self-assessment quizzes.



(b) Interactive cell with simulation of Bloch sphere.

Figure 2: Galore Learning Objects modeled as Jupyter notebook cells

we further logically organize these cells into various categories as mentioned in section 3.

Each *Galore* e-learning object maps to one or more cells in a notebook and each notebook represents a collection of learning objects. The learning outcomes of a notebook are a union of those of its constituent learning objects. Each notebook is mapped to a final quiz that tests the concepts covered by the learning objects in that notebook. In order to facilitate goal-oriented exploration of lesson plans, it is mandatory for each notebook in *Galore* to map to a final quiz. Figure 2 depicts some of the *Galore* notebook cells on Clark. Figure 2a depicts cells corresponding to the instantiated e-learning objects with preference parameter values text, example, and quiz respectively. These instantiated objects are generated by setting the user interaction parameter to the value static indicating that these cells do not generate responses to user interactions. On the other hand, figure (2b) corresponds to the instantiated learning object obtained by setting the preference parameter to the value widget and the interaction parameter to the value interactive. It depicts a Bloch sphere *widget* cell where users can modify values for parameters `theta, phi, azimuth,` and `elevation` using sliding scales to generate different qubit states visualized in the sphere.

The figure 3 depicts an *auto-graded-simulation* cell of the BB84 quantum key exchange protocol where users can modify qubit values and orthonormal bases to orient, measure qubits and enter the resulting secret key answer to be checked for correctness.

In total, there are currently 28 notebooks that cover concepts from linear algebra, quantum computing and quantum cryptography for secure quantum internet protocols organized into three overall units.

4.2. Gamelets

One of the first modules in *Galore* introduces the concept of qubit (quantum analog for a classical bit) programming through polarization of photons. This module includes four gamelet objects with three basic scenarios and one composite scenario where each scenario comprises of three challenges. In each challenge, the student must activate one type of qubit receptor – SameAngle (photon orientation must match that of the receptor for activation), Orthogonal (orientation must be orthogonal to that of the receptor for activation), and OppositeQuadrant (orientation equals an angle that is in the opposite quadrant of that of the receptor), by programming the qubit appropriately. The qubits must be programmed using the matrix, ket, or linear combination of the vector's representations of

photons in the three basic exercises. A combination of these representations must be consistent to activate the receptors in the composite scenario. Figure 4 depicts a gamelet object where a student has successfully programmed a qubit at 255 degrees using the matrix notation (seen on the workspace on the left of Figure 4) to activate an orthogonal receptor accepting photons at 345 degrees. As customary in serious games, the performance of the student and the qubits remaining are measured using a numeric score and health respectively as shown on the right hand top corner of Figure 4.

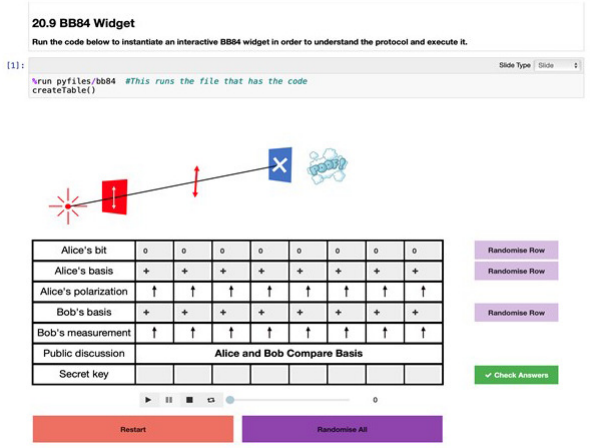


Figure 3: Interactive simulation of BB84 protocol.

4.3. Reflective Observation Zones

Suppose that a student must transfer 1-bit quantum information over a given 4×4 fiber network (QN) using the E91 QKD protocol [15]. Two gamelets are depicted in figure 5 (A and C) in order to model the game scenarios for this task. In the first gamelet in figure 5 (A), a student repeatedly selects pairs of adjacent network nodes to share an entangled qubit pair in the Bell state $|\psi_+\rangle$ such that one of the pairs remains in one of the nodes and the other is transmitted through the fiber link (black edges in the figure). Initially, all network edges are black. If two nodes connected by an edge succeed in sharing an entangled pair, then the corresponding edge turns red. Scoring in the gamelet is proportional to the number of distinct paths that a player establishes between Alice and Bob over the QN nodes.

After finding at least one red path (say, A-1-6-10-14-B) from Alice to Bob, a student proceeds to the reflection zone to observe a summarized gamified replay of their performance as well as that executed by a game-bot. These game simulations are aimed to trigger the process of self-discovery of the fundamental

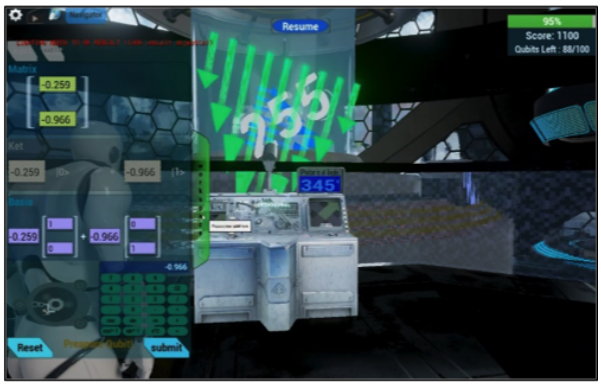


Figure 4: Gamelet for Qubit Programming

properties of QN (hints are used to aid discovery as well). For instance, by contrasting the nodes selected in two game simulations above, a student may discover the dashed edges inhibit qubit sharing and that common sub-paths can be used in the paths between Alice and Bob, and this can maximize scores. Further, a student may click and query specific QN edges to learn about link attributes such as fiber attenuation, loss and noise values of the links and hypothesize that their values negatively influence sharing of a qubit and eventually obtain the equation $p = 10^{\frac{-\alpha L}{10}}$, where p denotes the probability of exchange and α is the fiber attenuation coefficient and L us the loss value. The gamelet in figure 5(A) and related reflection zone can be repeated multiple times on a newly generated network each time.

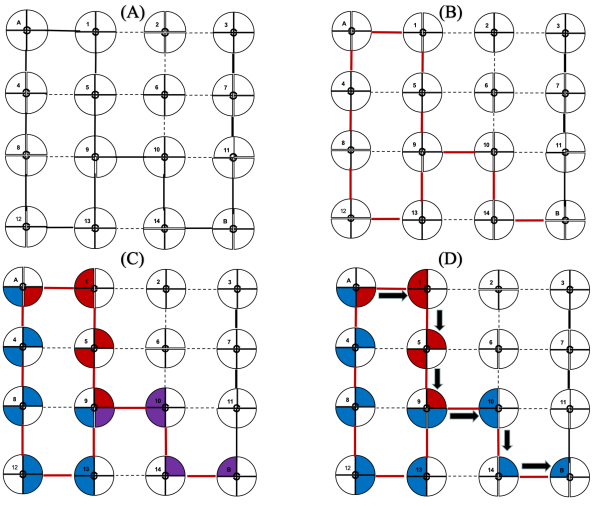


Figure 5: Network Example

The second gamelet, in figure 5(C) depicts the network where certain adjacent nodes have successfully shared qubits. A student must select some of

these nodes, swap qubits in these nodes to form an entanglement chain from Alice to Bob. The student score depends on the number of chains built by the student weighted inversely by the chain length. Student health is monitored in the same way as done in the first gamelet. The gamified reflection zone associated with this gamelet assists the user to observe the play simulations to discover the node and link properties leading to successful entanglement chains. These zones also support pre-programmed student queries to trigger the process of self-discovery of global and local routing heuristics for entanglement chaining from Alice to Bob.

4.4. Abstract Conceptualization Zones

The conceptualization zone of our running example involves a student programmatically transmitting qubits from Alice to Bob using a scaffolding model to generate boiler plates using pre-defined quantum computing Python libraries (such as QISKIT from IBM). Properties of the network including their scale will be used to foster generalization and abstraction. For instance, a student may be asked to exchange qubits over large networks as in figure 6 where manually choosing nodes to form chains is impractical.

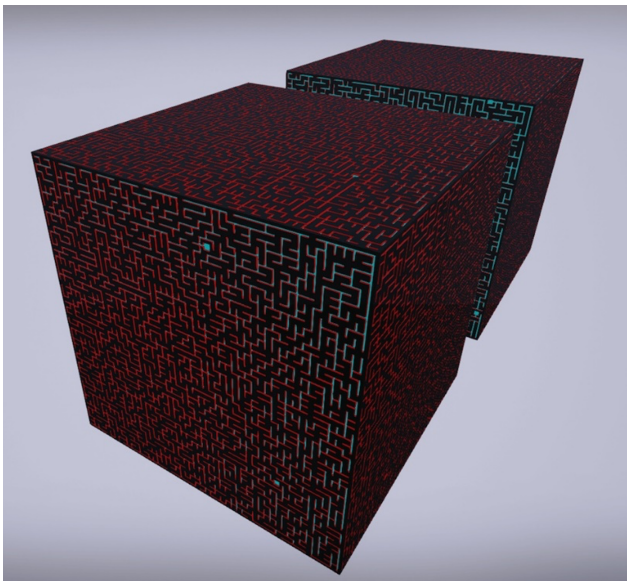


Figure 6: A 512 x 512 quantum grid with subnets

Qiskit provides several programming interfaces to manipulate quantum bits, measure and observe the results of manipulations. The qubit manipulations are done with the help of quantum gates such as the Hadamard gate, CNOT gate, Pauli gates, Toffoli and Clifford gates, etc. If a student wants to generate entangled qubits to simulate the implementation of the E91 protocol, the student can make use of two qubits starting in the ground state, then apply the Hadamard gate to one of the qubits to put it into superposition. Next task is to apply the CNOT gate to both the qubits entangling the qubits. Transmission of qubits can be simulated using classical-quantum programming interface and calculating the final state of the qubit. Once the final state of the qubit is computed, an appropriate custom gate can be instantiated with the IBM Qiskit framework to simulate the evolution of the qubit from the sender to the receiver. When applied this gate will result in the exact state that Bob would receive after being affected by noise during transit.

Every such transmission of qubit between various nodes in a network can be precisely controlled and simulated. Different network devices such as trusted nodes and repeaters differ mainly in storage capacity and whether they make qubit measurements or not. Repeaters have limited storage capacities whereas trusted nodes will make measurements. A student wanting to simulate trusted nodes within the Qiskit framework will make use of various measurement gates and classical registers to store the results. Repeaters, with limited memory capacity, on the other hand can be simulated using delay lines of high fidelity.

Qiskit allows for the implementation of quantum error correction codes using entangled qubits that Alice and Bob will need to use inorder to distill a high-fidelity encryption key. Quantum noise models can be simulated using several Kraus operators provided in Qiskit. Students may apply the common depolarizing error as well as amplitude damping error or choose from several other error models such as Pauli errors, thermal relaxation error and the phase damping error to closely simulate the quantum channel. Furthermore, Qiskit also provides mixed unitary matrices to simulate the probabilistic transition of a pure state to a mixed state.

Figure 7 shows code snippets that a student can implement to instantiate quantum bits, channel errors and transmission of qubits.

5. Conclusions and Future Work

Integrating serious games into instruction models based on traditional pedagogical practices is a challenging problem. This paper takes a step in addressing this problem by proposing a novel learning environment *Galore* that serious games into e-learning frameworks based on generative learning objects. Serious games encapsulated as generative learning objects called gamelets are embedded into a repository of parameterized e-learning objects and instantiated to produce customized lesson plans. The lesson plans

```
# Construct a 1-qubit bit-flip and phase-flip errors
p_error = 0.05
bit_flip = pauli_error([('X', p_error), ('I', 1 - p_error)])
phase_flip = pauli_error([('Z', p_error), ('I', 1 - p_error)])
print(bit_flip)
print(phase_flip)
```

(a) Code snippet showing the application of Pauli X and Z gates corresponding to bit flip and phase flip errors during transmission. Both errors are applied with probability p_error.

```
# Create an empty noise model
noise_model = NoiseModel()

# Add depolarizing error on qubit 2 forall single qubit u1, u2, u3 gates on qubit 0
error = depolarizing_error(0.05, 1)
noise_model.add_nonlocal_quantum_error(error, ['u1', 'u2', 'u3'], [0], [2])
```

(b) Code snippet showing the instantiation and application of depolarizing error using noisy gate implementations on a single qubit.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(3, 'q')
creg_mes = ClassicalRegister(2, 'mes')
creg_res = ClassicalRegister(1, 'res')
circuit = QuantumCircuit(qreg_q, creg_mes, creg_res)

circuit.x(qreg_q[0])
circuit.h(qreg_q[1])
circuit.h(qreg_q[0])
circuit.cx(qreg_q[1], qreg_q[2])
circuit.cx(qreg_q[0], qreg_q[1])
circuit.h(qreg_q[0])
circuit.measure(qreg_q[0], creg_mes[1])
circuit.measure(qreg_q[1], creg_mes[0])
circuit.id(qreg_q[2]).c_if(creg_mes, 0)
circuit.x(qreg_q[2]).c_if(creg_mes, 1)
circuit.z(qreg_q[2]).c_if(creg_mes, 2)
circuit.x(qreg_q[2]).c_if(creg_mes, 3)
circuit.z(qreg_q[2]).c_if(creg_mes, 3)
circuit.h(qreg_q[2])
circuit.measure(qreg_q[2], creg_res[0])
```

(c) Code snippet showing quantum teleportation application.

Figure 7: Code snippets showing instantiation of qubits, error models and transmission of qubits.

are presented as interactive Python Jupyter notebooks to students. Serious game-based reflective observation and abstract conceptualization zones enable *Galore* to combine serious games with learning objects based on the experiential learning model, which has proved to be an effective pedagogical approach across several domains. The paper also described a case study involving secure quantum internet protocols. A total of 28 notebooks and 16 gamelets were developed as part of the case study along with reflective observation and abstract conceptualization zones for establishing a quantum entanglement based key exchange over a square grid-based quantum network.

To the best of our knowledge *Galore* is the first learning environment that encapsulates serious games as learning objects and embeds them in generative learning object repositories and explicitly support game-based reflective observation and abstract conceptualization zones to provide a customized learning experience to students for quantum internet protocols. We believe the framework underlying *Galore* can be easily adapted to other domains. We plan to work with engineering and medical domain experts to develop learning modules with serious games in these areas.

Acknowledgement

A. Parakh's work on this article has been partially supported by a grant from University of Nebraska at Omaha under the U.S. Department of State-supported initiative Partnership 2020: US-India Higher Education Cooperation.

References

[1] S. Egenfeldt-Nielsen, J. H. Smith, and S. P. Tosca, *Understanding Video Games: The Essential Introduction*. New York, NY, 10001: Routledge, 2nd ed., 2012.

[2] "Cybersecurity library." `https://www.clark.center/home`.

[3] "MERLOT." `https://www.merlot.org/merlot/`.

[4] S. Arnab, T. Lim, M. B. Carvalho, F. Bellotti, S. de Freitas, S. Louchart, N. Suttie, R. Berta, and A. De Gloria, "Mapping learning and game mechanics for serious games analysis," *British Journal of Educational Technology*, vol. 46, no. 2, pp. 391–411, 2015.

[5] A. Parakh, M. Subramaniam, and E. Ostler, "Quasim: A virtual quantum cryptography educator," in *2017 IEEE International Conference on Electro Information Technology (EIT)*, pp. 600–605, May 2017.

[6] J. L. Plass, B. D. Homer, and C. K. Kinzer, "Foundations of game-based learning," *Educational Psychologist*, vol. 50(4), pp. 258–283, 2015.

[7] "About Learning Objects." https://www.wisc-online.com/about-learning-objects.

[8] H. Al-Chalabi and A. HUSSEIN, "Ontologies and personalization parameters in adaptive e-learning systems: Review," *Journal of Applied Computer Science Mathematics*, vol. 14, pp. 14–19, 01 2020.

[9] G. Weber and P. Brusilovsky, "Elm-art: An adaptive versatile system for web-based instruction," *International Journal of Artificial Intelligence in Education*, vol. 12, 01 2001.

[10] E. Melis, E. Andres, J. Budenbender, A. Frischauf, G. Goduadze, P. Libbrecht, M. Pollet, and C. Ullrich, "ActiveMath: A Generic and Adaptive Web-Based Learning Environment," *International Journal of Artificial Intelligence in Education (IJAIED)*, vol. 12, pp. 385–407, 2001.

[11] D. Kolb, "Experiential learning: Experience as the source of learning and development (vol.1)," 1984.

[12] R. Burbaite, K. Bespalova, R. Damasevicius, and V. Stuikys, "Context-aware generative learning objects for teaching computer science," *International Journal of Engineering Education*, vol. 30, pp. 929–936, 01 2014.

[13] F. Costea, C. Chirila, and V. Creţu, "Designing e-learning content using aglos," *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 685–690, 2019.

[14] "Model-driven processes and tools to design robot-based generative learning objects for computer science education," *Science of Computer Programming*, vol. 129, pp. 48 – 71, 2016. Special issue on eLearning Software Architectures.

[15] O. Amer, W. O. Krawec, and B. Wang, "Efficient routing for quantum key distribution networks," *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 137–147, 2020.

[16] V. Rossano, M. Joy, T. Roselli, and E. Sutinen, "A taxonomy for definitions and applications of los: A meta-analysis of icalt papers," *Educational Technology Society*, vol. 8, pp. 148–160, 01 2005.

[17] J. Vargo, J. Nesbit, K. Belfer, and A. Archambault, "Learning object evaluation: Computer-mediated collaboration and inter-rater reliability," *International Journal of Computers and Applications*, vol. 25, no. 3, pp. 198–205, 2003.

[18] M. Dondlinger, "Educational video game design: A review of the literature," *Journal of Applied Education Technology*, vol. 4(1), pp. 21 – 31, 2007.

[19] E. Simpson and S. Stansberry, "A guide to integrating cots games into your classroom, in handbook of research on effective electronic gaming in education," *Games: Purpose and Potential Education*, pp. 163 – 184, 2008.

[20] R. Van Eck, "A guide to integrating cots games into your classroom, in handbook of research on effective electronic gaming in education," *Information Science*.

[21] M. K.L., A. Orr, P. Frey, D. R.P., V. V., and A. McVay, "A literature review of gaming in education," *Gaming in Education*, 2012.

[22] M. Dondlinger, "Flow," *Journal of Applied Education Technology*, vol. 4(1), pp. 21 – 31, 2013.

[23] H. Thorkild, "Game-based teaching: Practices, roles and pedagogies," *New Pedagogical approaches in Game Enhanced Learning: Curriculum Integration*, pp. 81 – 101, 2013.

[24] S. de Freitas, M. Ott, M. Popescu, and I. Stanescu, "New pedagogical approaches in game enhanced learning: Curriculum integration," 2012.

[25] V. Bommanapally, M. Subramaniam, P. Chundi, and A. Parakh, "Navigation hints in serious games," in *Immersive Learning Research Network*, June 2018.

[26] D. Abeyrathna, S. Vadla, V. Bommanapally, M. Subramaniam, P. Chundi, and A. Parakh, "Analyzing and predicting player performance in a quantum cryptography serious game," in *Games and Learning Alliance* (M. Gentile, M. Allegra, and H. Söbke, eds.), (Cham), pp. 267–276, Springer International Publishing, 2019.

[27] A. Parakh, M. Subramaniam, P. Chundi, and E. Ostler, "A novel approach for embedding and traversing problems in serious games," in *Proceedings of the 21st Annual Conference on Information Technology Education*, SIGITE '20, (New York, NY, USA), p. 229–235, Association for Computing Machinery, 2020.