ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΚΠΑΙΔΕΥΤΙΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΦΥΣΙΚΩΝ ΕΠΙΔΡΑΣΕΩΝ ΣΕ ΕΙΚΟΝΙΚΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ

ΑΝΑΣΤΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ
ΒΙΤΤΕ ΣΤΕΦΑΝΟΣ-ΑΘΑΝΑΣΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος
Αναπληρωτής Καθηγητής

Λαμία .... έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΚΠΑΙΔΕΥΤΙΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΦΥΣΙΚΩΝ ΕΠΙΔΡΑΣΕΩΝ ΣΕ ΕΙΚΟΝΙΚΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ

ΑΝΑΣΤΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ
ΒΙΤΤΕ ΣΤΕΦΑΝΟΣ-ΑΘΑΝΑΣΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος
Αναπληρωτής Καθηγητής

Λαμία .... έτος 2025

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

EDUCATIONAL SIMULATION OF PHYSICAL EFFECTS IN VIRTUAL REALITY

ANASTOPOULOS PANAGIOTIS
WITTE STEFANOS-ATHANASIOS

FINAL THESIS

ADVISOR

Kolomvatsos Konstantinos
Associate Professor

Lamia .... year 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*
2. Δέχομαι ότι η αυτολεξεί ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*
3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια
4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/.…./20……

Ο – Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Το έργο «Προσομοίωση Φυσικών Φαινομένων» αναπτύσσει μια διαδραστική διδακτική εφαρμογή, συναρμολογημένη στο Unity, με στόχο να καταστήσει την οπτική πιο προσιτή μέσω προσομοιώσεων σε πραγματικό χρόνο της διάθλασης, διασποράς και ολικής εσωτερικής ανάκλασης. Αξιοποιώντας τους αλγορίθμους 3-Δ διανυσματικού raycasting του Unity, το σύστημα παρέχει δύο παιδαγωγικές επιδείξεις: έναν κόσμο διασποράς με τριγωνικό πρίσμα και έναν κόσμο διάθλασης αντικειμένου σε νερό, όπου ο χρήστης μπορεί να ρυθμίζει γωνίες, δείκτες διάθλασης και γεωμετρικές ρυθμίσεις και να παρατηρεί τόσο τα οπτικά αποτελέσματα όσο και τον αριθμητικό έλεγχο. Ένας δάσκαλος τεχνητής νοημοσύνης (Hugging Face Llama-3.1-8B-Instruct) παρέχει επεξηγήσεις κατάλληλες για το πλαίσιο των ρυθμίσεων της προσομοίωσης, υποστηρίζοντας τη μάθηση βασισμένη στην ανακάλυψη. Το έργο διατίθεται ως δύο συνταγμένες εφαρμογές που συνοδεύουν την παρούσα διπλωματική -μια έκδοση για υπολογιστή (πληκτρολόγιο & ποντίκι) και μια έκδοση VR για τρισδιάστατες ολογραφικές οθόνες -και οι δύο συναρμολογημένες από τον ίδιο βασικό πυρήνα προσομοίωσης και αριθμητικής μηχανής. Η έκδοση VR προσθέτει στερεοσκοπικές διεπαφές απόδοσης και αλληλεπίδραση μέσω χειριστηρίων κατά προεπιλογή, καθώς και διεπαφή χρήστη στον χώρο του κόσμου, διατηρώντας την ίδια αριθμητική σημασιολογία. Οι μελέτες περίπτωσης επικυρώνουν την φυσική ακρίβεια των προσομοιώσεων (συμφωνία στο επίπεδο των 0,1° ή καλύτερα σύμφωνα με τον νόμο του Snell) και δείχνουν σημάδια παιδαγωγικής αποτελεσματικότητας μέσω εργασιών βασισμένων στη μοντελοποίηση και τη λογική επεξεργασία. Με την ενσωμάτωση της VR λειτουργίας, τα επόμενα βήματα περιλαμβάνουν επεκτάσεις της συσκευής στη κυματική οπτική, ελεγχόμενες μελέτες με ανθρώπινα δείγματα κατά

λειτουργία για σύγκριση των μαθησιακών αποτελεσμάτων, καθώς και βελτιστοποιήσεις επιδόσεων και χρηστικότητας για αναπτύξεις σε τρισδιάστατες ολογραφικές οθόνες.

ABSTRACT

The Physical Phenomena Simulation project builds an interactive teaching device assembled in Unity in order to make optics more accessible by real-time simulations of refraction, dispersion, and total internal reflection. Availing the 3-D vector-based raycasting algorithms in Unity, the system provides two pedagogic demonstrations: a triangular-prism dispersion world and an object-in-water refraction world in which the user modulates angles, refractive indices, and geometrical setups and observes both visual outcomes and numerical verification. An artificial-intelligence tutor (Hugging Face Llama-3.1-8B-Instruct) provides context-appropriate explanations of simulation settings for supporting inquiry-based learning. The project is released as two compiled applications accompanying the current thesis -a desktop (keyboard & mouse) build and a VR build for volumetric displays -both assembled from the same base simulation and numerical engine; the VR build adds stereoscopic render passes and interaction by controllers by default and world-space UI while having the same numerical semantics. The case studies validate the physical fidelity of the simulations (agreement at the 0.1° level or better by Snell's law) and reveal signs of pedagogic efficacy by tasks of model-based reasoning. With the VR modality in place, the next steps are wave-optics extensions of the package, controlled studies of human subjects by modality for comparisons of the learning outcomes, and more performance and usability optimizations for volumetric display deployments.

Table of Contents

Chapter 1: Introduction

Studies of light and of interactions of light and matter are a core emphasis in the classic and in the current literature of physics. Familiar demonstrations one finds in the teaching environment-like an object appearing warped upon submersion in a liquid, refraction at flat surfaces, or a beam of white light split by a prism-serve as powerful representatives of the inherent first principles at work; they also indicate a chronic difficulty in teaching: algebraic proofs and line drawings often prove unsuccessful in conveying how repeatedly fluctuating variables (incident angle, refracting index, wavelength, and geometrical considerations) relate in order to produce the phenomena the student is asked to predict and explain. To achieve this shortage, the Simulation of Physical Phenomena offers a highly interactive and dynamic real-time environment in which the students can carry out virtual experiments and visualize instant outcomes while qualitatively exploring measurements in alignment with the fundamental ideals of optics. The system offers the same controllable variables as textbooks in the experiment mode-students manipulate the angles and the indices of the materials and the locations of objects and instantaneously see ray visualizations and numerical outputs-while an embedded artificial intelligence tutor provides contextually relevant explanations for facilitation of inquiry and reflection.

Notably, the project is offered in two modal forms extracted from a common codebase: a standard desktop version (using mouse and keyboard) and a special-purpose VR version for stand-alone headsets. The VR version includes stereoscopic perception of depth, world-space menus, and interaction by controllers, and they in turn encourage embodied exploration and development of spatial reasoning, while the desktop version provides access in standard classrooms. Both versions share the same simulation and numerical engine, so comparisons across modalities are meaningful; in the end, the project aims at shifting the learner from the receptive mode of passive acceptance to the active one of formulation of hypotheses, measurement, and conceptual understanding by way of immersive virtual experiments.

1.1: Background and Motivation

More recent work in the physical sciences teaching and learning community points up active engagement and participation, use of multiple representations, and routine formative assessment as the elements indispensable for effective conceptual change. Simulations that are interactive integrate visual, numerical, and manipulative representations in a unified framework in which the mental models of the learner can be constructed and tested. Applets and highly investigated collections such as the PhET simulations show how individually designed virtual experiments can enhance conceptual understanding, especially when embedded in planned activities or inquiry projects. Geometric relationships in the subject of optics are characterized by their spatial and directional nature; the fact of the addition of immersion and the possibility of real-time manipulation of angles, indices, and positions adds a powerful stimulus for the learning of correct intuitions.

Recent advancements in low-cost, real-time, three-dimensional engines-driven by Unity-along with ubiquitous access to managed natural-language inference application programming interface (API) packages, has lowered the barrier associated with the development of context-sensitive and highly interactive educational software. This work capitalizes upon such advancements in presenting two basic optical phenomena: a dispersion exercise utilizing a prism of triangles and a refraction exercise utilizing an underwater object facilitated by a conversational artificial intelligence (AI) assistant providing voice for the current simulation state in educationally desirable terms. The assistant is accompanied by a managed inference API (Hugging Face employing the Llama-3.1-8B-Instruct model) providing real-time explanations in contextually helpful terms triggered by a collection of simulation parameters spanning the likes of angles and refractive indices and points of intersection. Combinations of high-fidelity geometric simulations, discrete numeric overlays, and a contextualizing AI assistant should enhance self-paced learning and also accommodate instructor-led laboratory work.

1.1.a: Overview of Physical Phenomena in Optics Education

The current project studies three core phenomena:

- **Refraction.** The light is deflected while passing through the interaction surface of two media having different refractive indices according to the law of Snell. The law can also be represented in a scalar way by the equation $n_1 sini_1 = n_2 sinr_1$, or better in a vector system by the vector refraction equation used in this work. The refraction is responsible for the easily observable phenomena such as the water depth and forms the basis of many optical devices.

- **Dispersion.** The refractive indices for the majority of the transparent materials are wavelength-dependent. In passing through a prism the white light therefore gets dispersed in a spectrum: short waves (violet) being refracted more intensively than longer waves (red). An attempt at explaining dispersion in a teaching context is a balance of physical fidelity and teaching efficacy and a discrete sampling method by wavelength is typically the best understandable and computationally effective approach.

- **Total internal reflection (TIR).** This is the phenomenon in which light attempts to move from a denser medium to a less dense one at angles exceeding some critical value for which refraction is not feasible and total internal reflection of the incident light occurs. TIR is very sensitive to the changes in the refractive index ratio and the angle of the incident medium and hence forms a perfect candidate for an interactive demonstration indicating how qualitative gross changes in the behavior can emerge due to small variations in the parameters.

Particularly, the use of the prism and object experiments is a deliberate pedagogic method: both procedures are highly pictorial, briefly encapsulate concepts, and enable the use of instant numerical verification in suitable effective illustration accompanied by graphics. Thus, this selection enables the students to measure and check the same parameters and hence immensely enhance their model-based reasoning.

1.2: Project Objectives and Scope

The primary objective here is the development of a tutorial simulation software package such that (a) the package combines effectively elementary aspects of geometric optics for the demonstration of refraction and dispersion, (b) the package contains adjustable parameters by which cause-and-effect relationships can be explored by the user, and (c) the package provides quantitative diagnostic outputs along with short explanations useful for the interpretation of the associated equations and their computations.
Specifically, the scope of the project comprises:

- A **Prism dispersion** simulation by per-wavelength ray tracing of a triangle prism, interacting with a discretely sampled visual spectrum. This simulation outlines the entry and exit positions, normals, angles, and refractive indices, all of which are obtained from a simple linear dispersion mapping. Additionally, the simulation depicts the emergent rays by color-coded LineRenderer tracing and identifies the bins subject to complete internal reflection (TIR).

- A **water-and-object** experiment describing plane refraction and apparent displacement. This exercise, furthermore, introduces a simple and steady buoyancy model where interactive experiments on light (throwing small rigid bodies and studying their settling and damping behaviors) may be conducted.

- A **User interface**, **Camera** functionality such as a main menu, a pause, the ability to choose and manipulate things, and a cross-hair mode improve usability in situations where instructors are in control or a lone viewer.

- An **AI assistant** interface is intended to receive a brief representation of the simulation state (such as refractive indices, angles, and intersection points) and, in turn, presents contextual explanations via a real-time connection with a Hugging Face inference API. This assistant uses a structured prompt to ensure that the explanations are clear and instructional, especially focusing on optics principles. Responses are further delivered with a typewriter effect for better readability. The interface accepts user input via a text input field and a submit button, with a chat history retention ensuring contextual relevance.

- Three **Custom Shaders -** prism, water, and glass cup effects - have been created in an effort to increase the visual realism of the simulations. These shaders increase visual realism and facilitate educational discourse by the inclusion of colored refraction impacts within the prism, significant refractive distortions along the surface of the water, and real attributes of the glass forming the cup in a manner controlled by appropriate calculations carefully embedded in the simulation's coding. The shader for the glass cup might instead be used onto the one for the prism and hence display the internally refracted rays and provide an alternative perception regarding the light's behavior.

The design constraints were purposefully pedagogically motivated: the simulator enforces determinism, numerical fidelity, and control repeatability at the expense of photo-realistic light transport. It makes numerical overlays reliable for the right kind of measurement during interaction by the students and activities in the class.

1.2.a: What We Aim to Achieve – Practical Learning Goals

The simulation is designed to enable the following specific learning outcomes:

1. **Understand and apply Snell's law.** The students should be able to estimate refracted angles for given incidence angles and refractive indices and verify such estimates numerically by the use of the overlay.

2. **Observe and explain dispersion qualitatively and quantitatively.** Students should see that different wavelengths exit at different angles, understand the index-wavelength relationship implemented, and use per-bin critical angles to explain why some colors escape while others are trapped by TIR.

3. **Anticipate and determine total internal reflection.** The students will experimentally identify under what circumstances total internal reflection takes place for individual wavelengths by altering the tilt of the prism and the amount of dispersion.

4. **Connect geometric optics with everyday perceptual experiences.** The object experiment links perceived position of objects and geometric refraction and allows the calculation of apparent displacement for a wide range of viewing angles.

5. **Practice measurement and scientific reporting.** Laboratory standard worksheets assist in recording numerical results obtained from overlays, enabling comparisons of the results with calculations from analyses, and prompting reflection upon the range of variations seen (e.g., error in measurement, rounding, and approximations in modeling).

The specificity of such objectives has been designed carefully so that the teachers and the students can gauge proficiency in terms of short reusable tasks in a simulation.

1.2.b: Motivation for Implementation – Pedagogical and technical rationale

The decision regarding the use of the Unity platform was motivated by both teaching objectives and pragmatic considerations: Unity provides a user-friendly editing interface, allows for deployment across a wide variety of platforms, provides the fundamental physics primitives necessary in water simulation, and possesses a programmable render pipe which allows for the possibility of adding in shaders, and these factors enhance understanding. Ray sampling, binned by wavelength, strikes a balance between giving a qualitative account of dispersion and giving efficient, manageable computation on standard student equipment.

The chat assistant is designed to mimic the facilitating structure normally provided by the instructor in the laboratory setting; the assistant may direct the students to the correct numerical checks, explain the phenomenon of color extinction (Total Internal Reflection), or make slight adjustments. What is important here is the point the assistant functions as a learning tool and not a definitive testing tool: the hints and the solutions are accompanied by a notation of the numerical diagnostics provided and can be checked by the students.

Lastly, considerable emphasis has been placed in deterministic and precisely quantized actions like clearly defined overlays, predictable pseudo-random seeds operating upon stochastic elements, and constant buoyancy evaluations. This deliberate emphasis facilitates ease in classroom demonstrations and evaluation strategies. Particular care has been taken in eliminating complex volume calculations like precise displacement-volume integrals relating to buoyancy, thereby keeping computations simple and ensuring constant, realistic interactions for educational scenarios. In this way, facilitated by scripts like WaterPhysics.cs, which employs a linear buoyancy model, and PrismRaycaster cs, employing discrete ray sampling, ease of access and reproducing using common equipment is preferred. This allows instructors, therefore, to concentrate upon the teaching of the fundamentals of optics rather than having to contend with complex simulations.

1.3: Thesis Structure

The dissertation is organized to first situate the research within relevant literature, second, explain system design and its mathematical foundations, a discussion of implementation details and empirical verification, and finally, combine findings and recommend directions of future research.

- **Chapter 2 - Bibliographic Overview.** Literature survey across educational simulations, the use of game engines and XR teaching physics and prior work in the application of AI in educational software.

- **Chapter 3 - System Design and Methodology.** Descriptions of the three-scene architecture, the mathematical models (vector refraction, TIR criterion, discrete dispersion mapping and simple buoyancy), implementation decisions and the three custom shaders.

- **Chapter 4 - Implementation and Case Studies.** The chapter contains in-depth scene descriptions, details of the principal script and shader functions, and two case studies (prism total internal reflection and refraction by objects) thorough numerical verification and recommended class activities.

- **Chapter 5 - Conclusions and Future Directions.** Comprehensive summary of results, discussion of the limitations, and suggestions for the future breakthroughs (potential applications of XR, studies of more phenomena, development of complex wave-optics devices, and systematic assessment of teaching techniques).

Chapter 2 Bibliographic Overview

2.1: Literature Review on Educational Simulations

Educational simulations have been thoroughly investigated and optimized within the science educational arena because they can make abstract phenomena specific, allow variables testing in a risk-free setting, and yield expedient feedback mechanisms allowing for inquiry-based learning. The prime example for such a trend is the University of Colorado Boulder's PhET initiative which in the course of development took a research-informed approach towards design by utilizing iterative student interviews and classroom testing in feeding the interface and instruction design choices; papers and studies developed by PhET record that appropriately designed interactive simulations have the potential for improving students' conceptual knowledge and motivation towards the discipline of physics (1, 2).

Traditionally, physics instruction was based upon static visual images, analytical equations, and a restricted set of lab experiments. But in the past thirty years, improved computational technology, web access, and mobile phone apps have enabled more sophisticated simulation techniques to emerge: pioneering applets have given way to research-proven, pedagogically informed frameworks offering several related representations (graphic, numeric, and symbolic). Systematic reviews and meta-analyses indicate the educational efficacy of simulations is maximized if the activity possesses clear and specified learning targets, concise but focused guidance, and the possibility of measurement and testing of hypotheses-design parameters having significant implications for interface design and debugging overlays in this project (3, 4).

2.1.a: Historical Development of Physics Simulations for Teaching

The transition from classic graphical presentations through sophisticated, research-based simulations came with the availability of inexpensive multimedia and scripting tools during the 1990s and early 2000s; PhET emerged as a National Science Foundation- and university-sponsored project established specifically for developing research-based, publicly available simulations suitable for K–12 and post-secondary education (5). Follow-up studies on related topics entailed empirical validations-measuring outcomes from computer-based simulations against those obtained from classic instruction or experiential laboratory work-finding that high-quality, virtual laboratories reliably match or excel outcomes from conventional laboratory work involving narrowly defined conceptual measures, thus presenting secondary benefits in terms of cost-efficiency, safety, and reproduction (4). Such findings complement the project's focus on numerical overlays, test log reproducibility, and structured activity, allowing pupils to see and verify the equations they meet during their course of learning.

2.1.b: Role of Game Engines like Unity in Educational Tools

Over the past decade, general-purpose game engines, and in particular the Unity platform, have also been widely adopted for the development of interactive educational experiences. Unity also offers several important advantages in the form of a mature editor and a tried-and-true rendering pipe-line along with integrated physics and strong support for desktop and XR platforms and consequently lowers the cost of engineering and the iteration cycles for developers and instructors equally. Commercial virtual lab vendors and numerous academic programs also prefer the use of Unity for prototype and production deployments due to the ease of rapidly generating the contents and the reuse of the assets and providing stable performance across a wide variety of platforms; the current work employs Unity for the same reasons along with taking advantage of the rendering and the facilities for script programming for real-time ray-based renderings of the optical phenomena (6).

2.2: Augmented and Virtual Reality in Physics Education

Virtual and augmented reality technologies enhance desktop simulation capabilities by providing heightened immersion, possibilities for stereoscopic perception of depth, and natural variations of scale and perspective-the latter especially useful in subject areas in which a strong geometrical component exists, i.e., optics. Thorough reviews of the use of post-secondary university or college studies in immersive virtual reality consistently show promising outcomes in student engagement and understanding of the situation and are contingent upon the design of the components of the experience of VR in individual instructional outcomes and pedagogic context (7, 8). From the reviews, the use of extended reality (XR) is not feasible across the board, but XR can very effectively support activity in spatial reasoning and permit the student more natural interaction in a directional mode-this element being important in design for the use of XR in the current project.

2.2.a: Applications of AR/VR for Visualizing Optical Phenomena

The tailored XR implementations deliver sets of optical visualizations representing measurable physical interrelations. For example, femtoPro offers a training system in virtual reality on ultra-fast optics, utilizing Gaussian-beam propagation and ultrashort-pulse dynamics in a virtual laboratory environment. The system offers precise measurements and alignment, which are difficult or impossible in conventional training laboratories (9). Likewise, VisionaryVR offers a platform for performing vision-science simulations and optical experiments in a virtual reality environment, utilizing simulations of different defocus and aberrations both for research and education (10). Such projects demonstrate the possibility of blending ray-based and wave-informed simulations into immersive presentations, which not only deliver educational benefits but define design guidelines providing numerical accuracy in converting desktop worlds into XR applications.

2.2.b: Case Studies of AR/VR-Based Simulations in Refractive and Dispersive Optics

Recurring case studies of XR optics tools consistently reveal three general results having straightforward design implications for this project. First, quantitative fidelity emerges as a necessity: students can make XR measurements corresponding to analytical predictions under the assumption that the simulation respects fundamental principles (e.g., the law of Snell and relationships for the critical angle). Secondly, interactivity is shown to be essential: the permission for students themselves to set incidence angles, refractive index, and geometrical parameters enhances conceptual understanding by forging causal relationships. Thirdly, the value for learning outcomes of scaffolding is manifest: tutored tasks and in-built tests (e.g., numeric overlays, displays of angles and test logs) yield more learning advancement than solo unaided exploration (2, 7, 9). These results endorse the use in this project of debug overlays showing displays of angles and expressions for the Snell check along with prominent indicators for total internal reflection for the various wavelength classes.

2.3: Related Technologies and Frameworks

A teaching simulation's effectiveness relies upon a seamless aggregate of technologies including a real-time engine (Unity), numerically exact refraction and ray module, overlay-level user interface tool-kits, and optionally a set of cloud-based artificial intelligence services capable of facilitating conversational interaction. The choice of each component requires a subtle balance of fidelity, interactivity, and reproducibility. The base scene is the Unity engine, offering rendering and scripting support for this project; a discrete ray-sampling method for each wavelength accurately simulates dispersion in a loss of photo-realism for the sake of interactive stability-this design is justified, as teaching demonstrations value consistency and repeatability more than photo-realism (6).

2.3.a: Integration of AI Assistants in Educational Software

Existing literature and in-practice applications document that AI-assisted tutors and assistants have the potential to enhance the learning experience significantly when employed judiciously. Inference management services, such as the Hugging Face Inference API, enable developers to detail concise contextual information from an application, which can include a snapshot comprising numerical diagnostics and a collection of state variables and then retrieving pertinent explanation text; such a design distinctly segregates the architecture of the AI assistant employed in this project work (11). Media analysis and real-world evidencing validate the fact that custom-made AI-based tutoring systems, employed along with traditional instructors and designed in concomitant alignment with teaching frameworks such as scaffolding and formative feedback, stand a good chance of facilitating the effectiveness of learning as well as student engagement in post-school contexts; yet existing scholarship underlines the necessity for effective evaluation, transparency, and caution regarding blind reliance upon AI as a final decision-making entity (12).

2.3.b: Existing Unity-Based Projects for Physics Phenomena

Various proven initiatives demonstrate the potential of Unity and related engines to enable wide-scale education in physics. Labster is one such commercialized virtual lab platform utilizing high-fidelity worlds developed within Unity for simulating lab procedures within the biological, chemical, and physical sciences; the Labster-provided materials alongside their corresponding assessment guides argue that virtual laboratory efficacy is increased when combined with evaluative and pedagogical approaches (13). Pedagogical frameworks utilizing Unity-based lab systems intended for optics and advanced lab uses, such as the femtoPro VR lab, demonstrate the ability of real-time engines to maintain high-fidelity ray and pulse simulations within training scenarios, subsequently verifying the viability of the techniques utilized in the present project's per-wavelength tracing and numerical verification schemes (9, 13).

Chapter 3: System Design and Methodology

This chapter includes the architecture, design principles, mathematical fundamentals, and implementation decisions employed while building the project of Physical Phenomena Simulation. The chapter describes the establishment of the environment of the three scenes by employing the use of Unity objects and scripts, describes the numerical simulation employed for refraction, dispersion, and buoyancy, sketches the project's three specially designed shaders for prism, glass and water, and describes the user interface, the integration of the AI assistant, and the debugging and verification tools employed for teaching.

3.1: Overall architecture and scene layout

The project comprises three variously different Unity scenes, which, while possibly self-standing, are managed by an overlord system: a simplified Main Menu scene for simulator selection, Simulation 1 (Prism Dispersion), in line for the examination of ray-based optics, and Simulation 2 (Water & object), enabling the examination of plane-interface refraction and conducting simple buoyancy experiments. Commonly shared resources include a common AI-Assistant prefab offering context-sensitive guidance and a lowly set of utility scripts (which help in vector refraction, color mapping into refraction bins, and pooled management of the LineRenderer). The project also comprises an EditorCameraController component for camera control and selection along with access specifically for the build for the keyboard-and-mouse application build's pause screen; the script remains intact and unused in the build for the VR. The build for the VR comprises XR-specific input and UI prefabs (such as world-space canvases, controller ray interactors, and XR rig prefabs) offering equivalent user-facing functionalities in an immersiveness setting.

Each scene comprises a generic simulation object, scene-specific prefabs (e.g., a prism mesh for Simulation 1 and a water cup prefab with object pivot for Simulation 2), and a Canvas-based implementation for UI elements for controls, overlays, and assistant activity. The simulation loop for each scene is utterly transparent and student-friendly: user inputs (including camera and UI controls appropriate for the current build) manipulate the inspector-visible parameters; the core simulation controllers-PrismRaycaster.cs for the prism and ObjectRefraction.cs and WaterPhysics.cs for the water scene-solve ray intersections and physical interactions using numerically exact equations; the visual output is calculated by making use of LineRenderer objects, meshes, and two custom-made shaders (prism and water) in an attempt to render refracted light and underwater objects; and lastly the AI-Assistant is able to take a short screenshot of the current simulation state and create context-appropriate explanations. The project repository contains a README of the details of the version of the version of the Unity editor needed, packages needed, and build/run instructions for the desktop and the VR builds such that reproduction is discussed in the supplemental materials and not repeated in the main document.

3.2: Design Principles

A set of clear design principles underlies the implementation process. First, the simulation employs the approximation of the geometric optics, in which the light is described as rays tracing along linear segments along the interaction points. The reasoning behind this effectively decouples phenomena such as refraction, dispersion, and total internal reflection from wave-related phenomena such as diffraction and interference falling outside the educational objectives of this project. Secondly, dispersion is simulated by discrete wavelength sampling: rather than attempting to continuously render the spectrum, the system samples the visible spectrum by a finite number of rays (the default uses 21 bins) and calculates a refractive index for each sample via a simple linear mapping based upon a parameter. Finally, numerical transparency and verifiability were a priority: the system possesses numeric overlays which render the entry and exit points, surface normals, incidence and refraction angles, and checks which abide by Snell's law such that the student may check the underlying equation upon which the visualizations are based. Lastly, interactivity and safety influenced physical model design: water buoyancy representation sacrifices volumetric accuracy for stability and predictable behavior such that, in the context of the interactive framework of the classroom setting, is assessed as a desirable trade-off in terms of pedagogy.

3.3: Mathematical Foundations

All computations involving refraction and total internal reflection (TIR) are carried out using conventional, vector-based expressions of Snell's law. A normalized direction of incidence d_{inc} approaches in, and a unit surface normal vector $\mathbf{n}$ projects out of, the medium. Cosine of the angle of incidence relative to the normal is found by:

$$cos\theta_i = -(d_{inc} \cdot n)$$

When moving from a medium of refractive index n_1 into another medium of refractive index n_2, the refracted direction is computed via the ratio:

$$\eta = n_1/n_2$$

coordinated with the squared sine identity:

$$sin^2\theta_t = \eta^2(1 - cos^2\theta_i).$$

If $sin^2\theta_t > 1$, the angle of refraction becomes imaginary, meaning there is no refracted ray; such a situation clearly exhibits the TIR condition. When, in a particular situation, there can be refraction, the refracted direction vector d_{out} is found by:

$$d_{out} = \eta d_{inc} + \left(\eta cos\theta_i - \sqrt{1 - \eta^2(1 - cos^2\theta_i)}\right) n$$

The implementation encapsulates this routine in a small utility function (e.g., RefractionUtils.TryRefract) that returns a boolean indicating success and the output vector when successful. The critical angle for an internally incident interface is computed as $\theta_c = arcsin(n_{out}/n_{inc})$ when $n_{inc} > n_{out}$, and the code uses this to produce TIR flags and explanations visible in the overlay.

3.3.a: Dispersion model (discrete sampling)

Dispersion is realized by linking a normalized bin parameter t ∈ [0,1] to a wavelength-sensitive refractive index n(t) using linear interpolation around an index n_{center}. A standard value of $dispersionStrength = 0.1$ is used for the purposes of this research. Internally the per-bin offset n_Δ is computed as:

$$n_\Delta = dispersionStrength \times 0.5,$$

which yields $n_\Delta = 0.05$ with the chosen value. For a typical central index of $n_{center} = 1.500$, the range of indices per bin is bounded by [1.450, 1.550]. The mapping can thus be represented by:

$$n(t) = n_{center} - n_\Delta + 2n_\Delta t,$$

where t = 0 is the red-most, low-index bin, and t = 1 is the violet-most, high-index bin. This simple linear method has instructional advantages: it yields qualitative insight into the effects of normal dispersion (shorter wavelengths being refracted more) without sacrificing computational efficiency or instructional simplicity.

3.4: Shader Graphs for Visual Refraction

In order to achieve a clear line of separation between numerical ray optics and rendering computation, the project contains three designated Shader Graph materials for visual rendering reserved for use: a glass shader material, a prism shader material, and a water shader material. Each of the shader materials performs scene sampling in screen space and adjusts the color sampled in a manner simulating refraction and related effects. Since such shader materials compute based upon screen-space samples instead of the ray traced by the physics engine's geometry, their visual outputs are kept aesthetically suitable such that the numerical ray direction accuracy and the simulator's debugging output are also retained.

3.4.a: Glass Shader

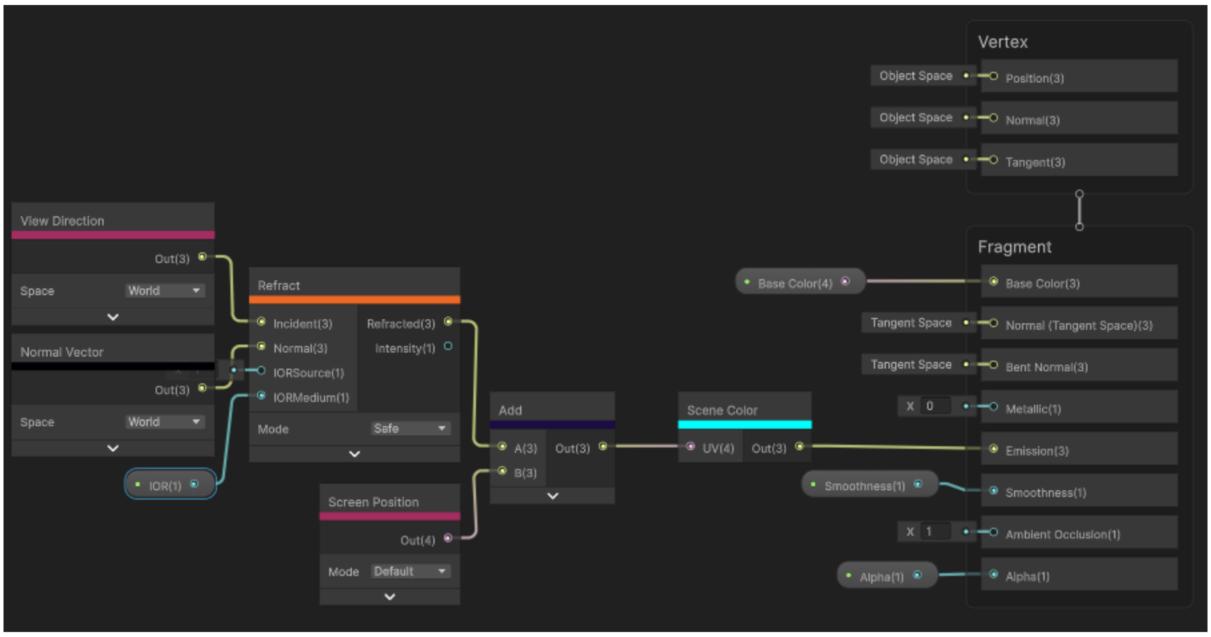
This shader for a glass emits a classic appearance common in screen-space refraction:

• The system calculates a view direction and employs a Refraction node to create a refracted direction or a UV offset based on a one-IOR input.

• Offset functions for the display task showing the Scene Color (screen buffer) at refracted UV coordinates and thus giving the optical illusion of objects outside the glass appearing misaligned as seen through.

• Reflection Probe, aided by a small Fresnel add-on, blends in the specular reflections and background refraction, providing a gloss look. Variable parameters such as smoothness and alpha transparency are employed to alter the glass surface so it changes everything from a dull finish to a highly shiny finish.

Since it employs screen-space sampling this technique successfully represents foreground objects in motion across the surface of the glass at a range of angles but refracts no information off the screen buffer such as reflections need multi-sample ray-marching.



Space World

Space World

Mode Safe

Mode Default

Figure 3.4.1: Glass shadergraph.

3.4.b: Prism Shader

Prism shader captures the reality of color-separation appearance faithfully but, importantly in this context, achieves this not by wavelength-dependent geometric refraction but by screen-space chromatic aberration.

- The graph calculates the output in several Refraction nodes (wherein each one would represent a various channel group). The various Refraction nodes are given different IOR values (or an IOR modulated by a marginal Dispersion Strength), and this produces relatively different UV offsets of the red, green, and blue channels. The shader then samples the Scene Color by itself in each one of these offset UVs and combines the resulting samples into the final color in the output.

- Per-channel sampling technique generates a characterized laterally-shifted color, a case supported by the background cube indicating the color channel distortion introduced by the prism presenting in readily-identifiable forms of dispersion. But not having the shader simply modify pixels in-screen sampled and not changing geometry or accurately computing resulting ray directions-thus introducing no changes of the numerical ray exit computed by the Prism's Raycaster-this effect is to be reckoned in optical terms not as physical dispersion but rather as chromatic aberration.

- The shader uses a combination of a Fresnel term and reflection probe data in a way to preserve specular highlights for steeply angled surfaces.

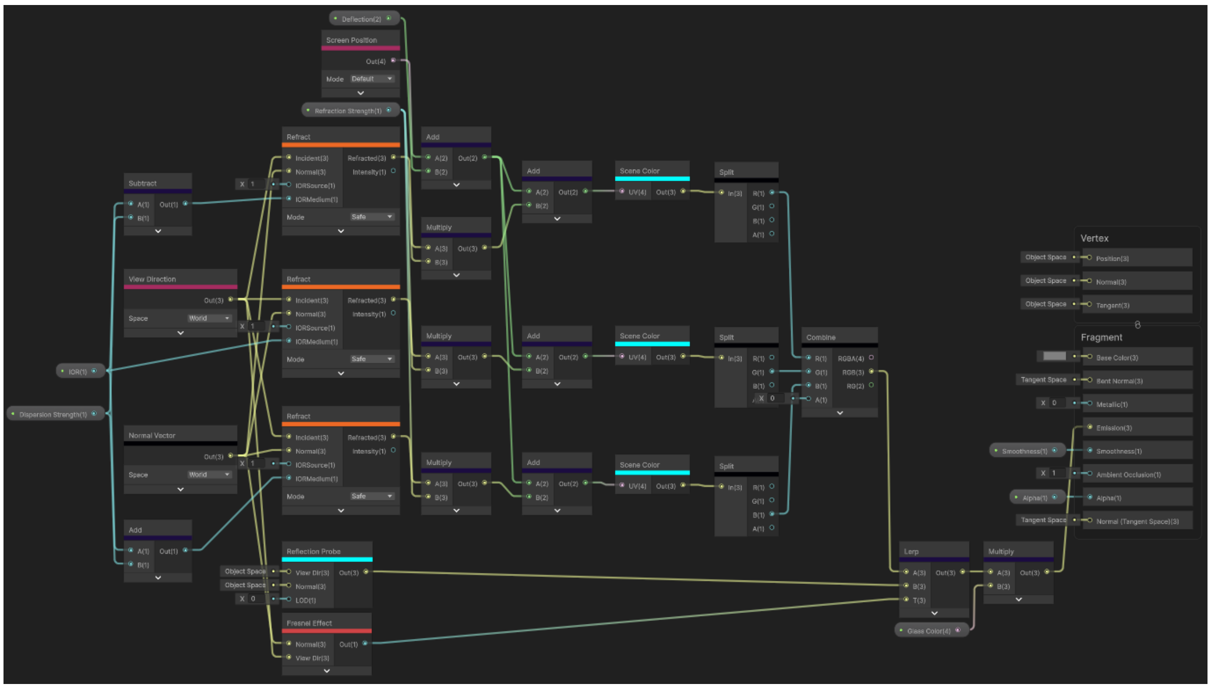


Figure 3.4.2: Prism shadergraph.

3.4.c: Water Shader

The water shader highlights a vibrant water surface and muted refractive characteristics:

• It produces dynamic ripples across a surface. It is calculated in tangent space about the normal map and whose value becomes an input parameter for establishing the desired amount of such ripples. The view direction is also included along with the normal data for the computation of a refraction offset.

• The shader also employs a Refraction or any similar mathematical technique for displacement of the UV for the sampling of Scene Color at offset UV coordinates in the development of a distorted view of objects viewed through the water. As the water surface employs a normal map for thickness and absorption input, the shader can attain subtle color attenuation by depth in line with the settings of the transmittance color and the absorption distance. A parameter is also given for the refractive index so it may be tweaked based upon empirical measurements of how much the object appears to distort upon submersion.

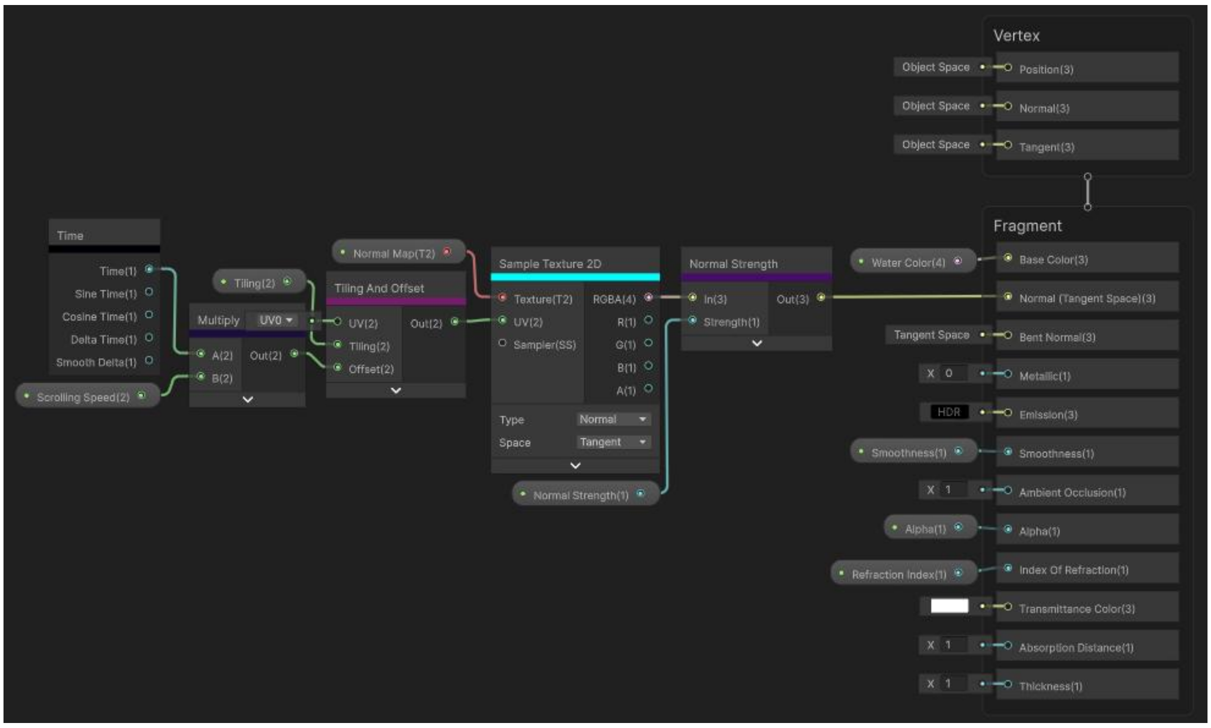


Figure 3.4.3: Water shadergraph.

3.5: Prism implementation and visualization

The prism simulation is managed by "PrismRaycaster.cs" associated with the LightBeamEmitter GameObject. The emitter projects a base ray and, upon collision with the prism collider, calculates entry point and normal and calls TryRefract to acquire the internal direction. The internal ray is traced for the exit face. For a wavelength-specific IOR a given dispersion bin calculates and calls TryRefract at the exit face; upon successful refraction a pooled LineRenderer plots the exiting ray and upon failure the renderer for the bin is disabled and the override counts a TIR event. To make it easy for students to visually discern adjacent rays the DrawDispersion routine offsets the line renderers' start locations slightly in the local incidence plane; this lateral fan is a pure visualization aid and does not influence the physical calculations.

The aesthetic look of the prism is supported by the combination of two distinct shaders. The prism shader is designed carefully to mimic a semi-transmissive appearance mimicking the real-life caustic behaviors of prisms and hence providing a bright-looking appearance characterized by muted tones dispersed in specific wavelength groups. The shader operates similarly to a base surface (or fragment) shader in the render pipeline of Unity by utilizing a composite lookup table (or uniform array) of the chromatic colors associated with the dispersionRayCount groups. It consists of a debug mode showing high-contrast spots of incidence and the surface normals useful during the development of overlays for training. Significantly, the Prism shader does not require the numerical raycasting calculation specified in the PrismRaycaster.cs and does not carry any directional calculations. The prism shader also can be replaced by the glass cup shader, a customized water cup scene implementation, thereby enabling the visualization of the internally refracting rays during the presentation of training.

Prism shader is also complemented by a simple screen-space overlay shader providing a "caustic" view by summing inputs at different wavelength together as a blurry projected pattern onto surfaces beneath the prism. The effect is purely for illustration and is disabled by default in the interests of preserving measurement accuracy and does not affect physical calculations.

The computation parameters in the Prism scene are prismMeshFilter, prismLayerMask, indexOfRefraction (center), dispersionStrength, dispersionRayCount, lineRendererPrefab, and maxDistance in the exit ray casting. The operations for enabling the prism shader debug mode and the caustic overlay are also in the script.

3.6: Water, object and buoyancy model

The water simulation integrates a geometric refraction mechanism (ObjectRefraction.cs) and a simple yet effective buoyancy management system (WaterPhysics.cs), which is assigned to the Cup GameObject. The WaterPhysics script determines the water surface height based on constraints set by the renderer and exposes a public property, WaterSurfaceY, which is accessed by any other scripts (e.g., the object script) in order to move the water plane. Buoyancy forces are simulated for any rigid objects through the OnTriggerStay method, which calculates an upward acceleration proportional to buoyancyStrength times depth, the depth being the vertical distance between the water plane position and the rigidbody's center of mass. The model employs smooth linear interpolation by way of Mathf.Lerp in order to smooth sudden movement during collision detection and incorporates a fraction of counterforce of gravity in the downward axis and thus in favor of natural interaction rather than precise displacement and volumetric computation. ObjectRefraction.cs takes a query of WaterSurfaceY and calculates the intersection of an object-local geometric ray and the horizontal plane. It determines AOI and employs the common utility TryRefract in shared mode in order to calculate the refracted underwater direction under the assumption of $n_{air} = 1.000$ and $n_{water} = 1.33$. The script renders the underwater part by a use of a LineRenderer and optionally instantiates an "apparent object" mesh whose transform is set according to the refracted geometry and allows a visual comparison of physical and apparent locations.

The water cup employs two shaders in making the cup more realistic. The glass cup shader achieves a realistic glass material with high refractive distortions, adding to the cup's realistic appearance and allowing the cup to be employed as a prism for viewing internal rays. The water shader displays a stylized liquid surface with distortion effect, utilizing a perturbation driven by a texture along with a screen-space grab-pass in order to subtly displace content beneath and thereby replicate bending on surfaces. It does not achieve physical refraction but rather operates to supplement numerical accuracy of ray lines in view in the ObjectRefraction.cs. Inspector-controllable parameters consist of surface wave amplitude and normal scale and the same can be tweaked in terms of clarity appropriate for the task of measurement.

3.7: UI, camera, and AI assistant integration

The EditorCameraController (as used by the keyboard-mouse program) includes camera control, rotation, picking of objects, dragging/rotation of objects, locking and a cursor crosshair in camera control mode. It also links the Pause Menu Canvas (Resume, Reset, Options, Main Menu, Quit) by the TogglePauseMenu() which disables camera control and freezes time of simulation. The desktop build also includes a safe system of dragging objects by disabling gravity and freezing rigibody momentum for a short time interval of dragging, as well as individual cursor textures for the idle/drag/release states and a few quality-of-life functions.

For the build in VR, corresponding user interactions are presented by XR-specific widgets: the numeric overlays and the AI assistant appear as world-space canvases; presentation and input appear as a set of controller ray interactors and grab/teleport affordances suitable for the target headset. The build simulation does not alter the simulation logic-the input and presentation layers differ-so the students view the same numeric diagnostics and simulation activity whether they work at the keyboard/mouse or by controllers in the VR.

The UI environment includes an Options panel with two tabs: Controls (help and hotkey bindings) and Settings (scene-dependent simulation parameters). The Options panel does exist and is in use in Simulation 1 (Prism Dispersion). In the Main Menu and in Simulation 2 (Water & object) the Settings tab intentionally shows the message "No Settings Available for this Scene" wherever the case may be. In Simulation 1, the OptionsMenu prefab controls the Settings panel and reveals inspector variables for the prism materials and a dispersionSlider hard-wired at run-time to the scene's PrismRaycaster object; the changes cause PrismRaycaster.dispersionStrength to be updated at once and the UI shows the value rounded to 2 decimal places for run-time experiment without access to the editor.

The AI assistant is embedded as a chat-like interface in the user interface. It asks for a short summary of the current simulation status upon request for the following: identification of the scene, significant numerical diagnostics such as entry and exit angles and normals, the current n_center and index range, total internal reflection counts, and a short debug message. This information is then communicated to the designated inference endpoint; the results are then integrated back in the user interface as short, actionable explanations or suggestions (e.g.: "rotate the prism by 5° and see the exit angles change"). The assistant can only perform purely descriptive and support tasks-proposing experiments and interpreting diagnostics-without participating in evaluation or providing instructor feedback replacement. Error management mechanisms and simple request limiting by rate are in effect at the client side in an attempt to prevent the occurrence of fast, repetitive questions. Also, API credentials for external services must be furnished securely during deployment and are not under source control.

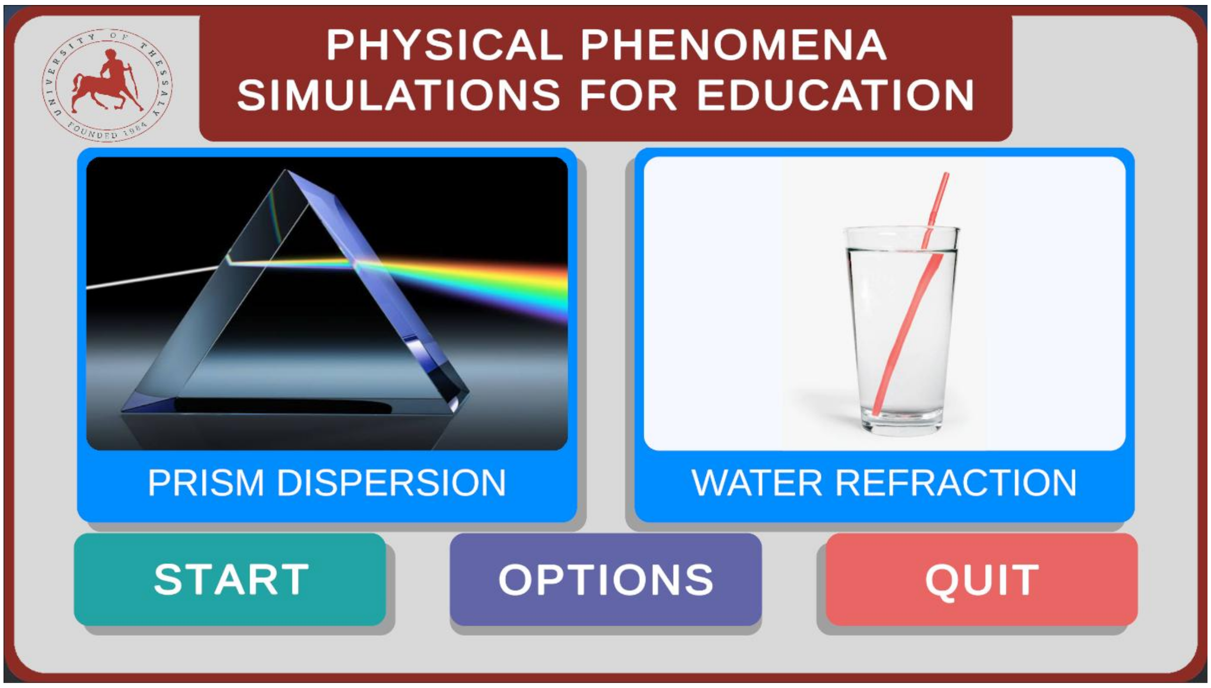


Figure 3.7.1: Main Menu (Prism Image & Glass Image – Main Menu)

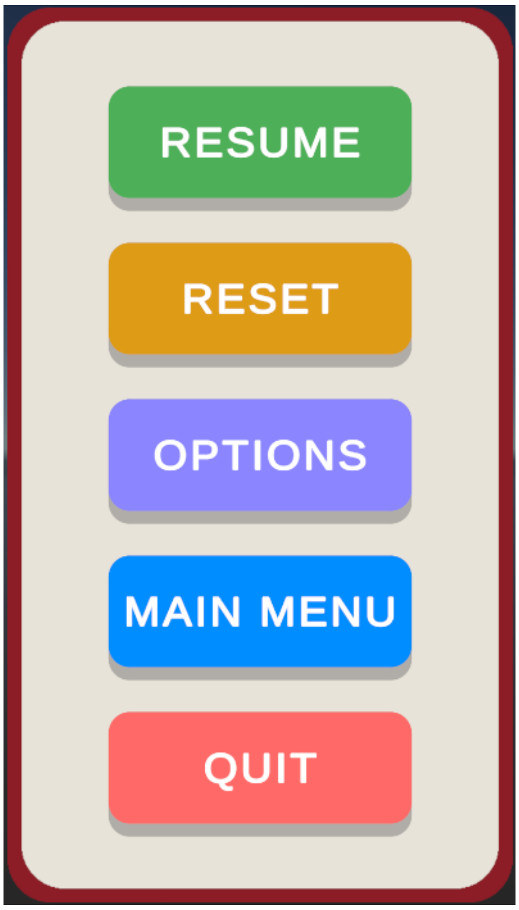


Figure 3.7.2: Pause Menu

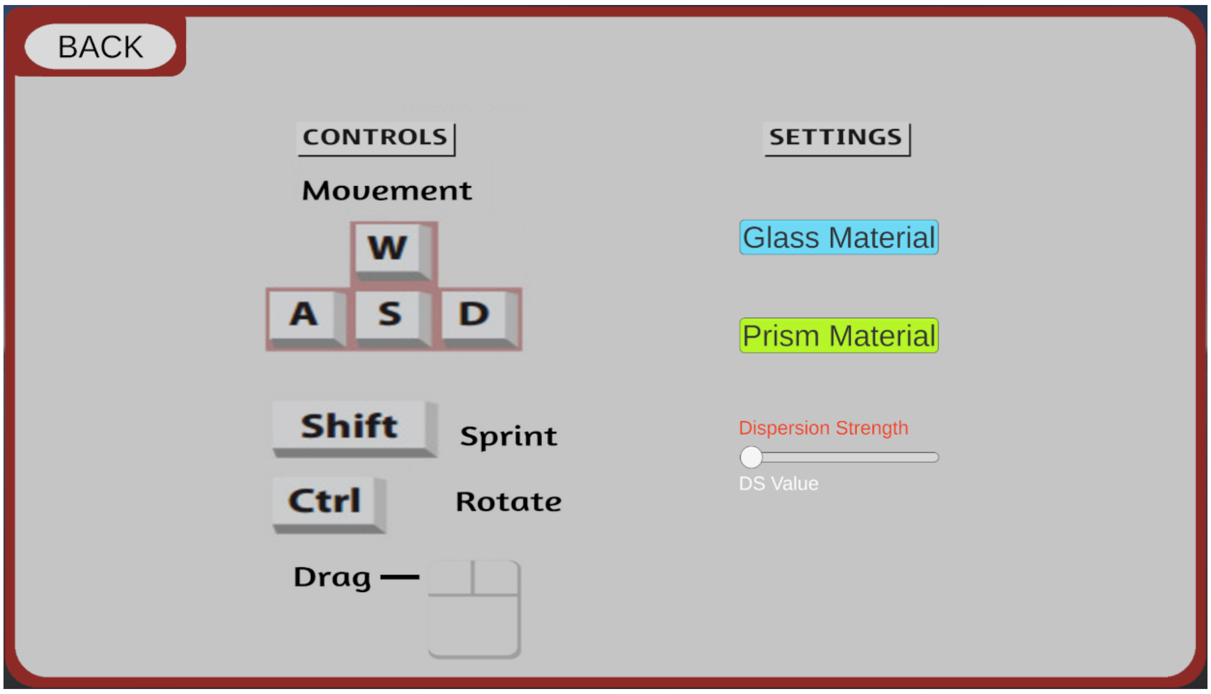


Figure 3.7.3: Options Menu for PC version (Simulation_1) (Options Menu)

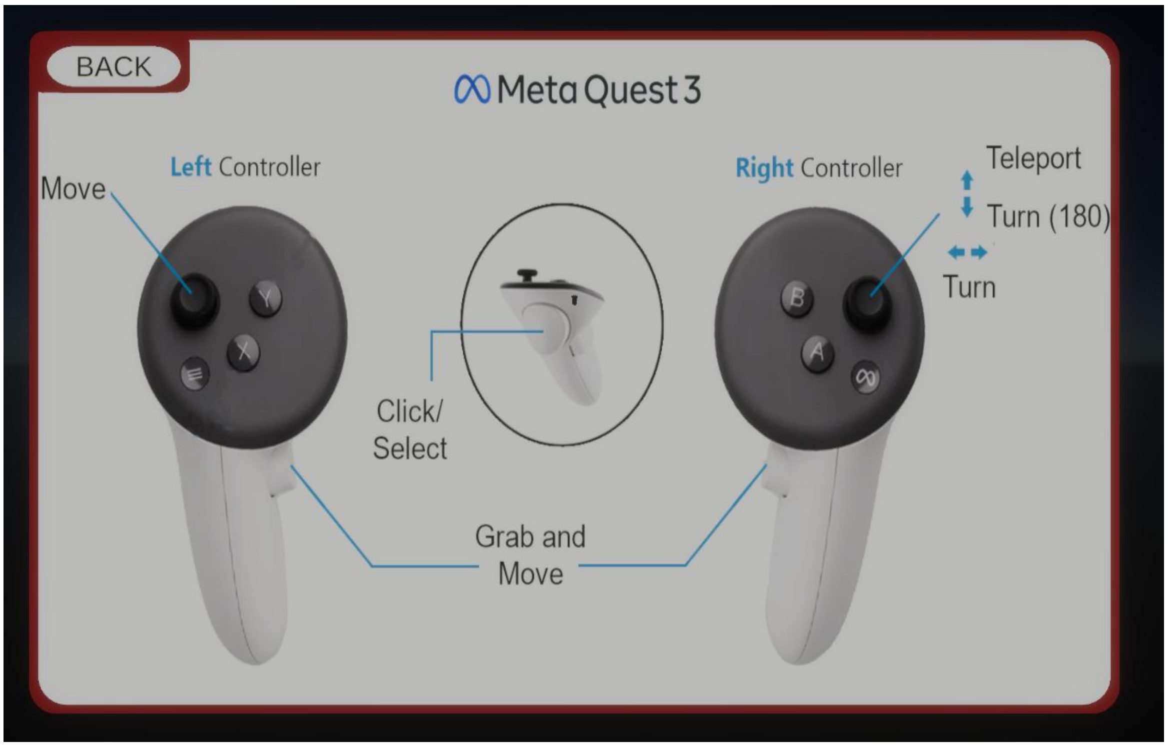


Figure 3.7.3: Options Menu for VR version (Simulation_1) (Options Menu)

3.8: Debugging, visualization and validation aids

Besides facilitating learning and in enabling graders to check numerical accuracy, the project includes several runtime diagnostics. The following are shown in an on-screen display: entry and exit points, surface normals, incidence and refraction angles in degrees and the numeric form of the Snell law:

$$n_1 * sin(i_1) \approx n_2 * sin(r_1)$$

Color-coded "LineRenderers," in increasing wavelength order, are pooled during the Awake() process and run from red through violet. In the editor view, Gizmos plot normals, entry and exit points, and small-angle arcs useful in visualizing the geometrical inter relationships. When a bin undergoes Total Internal Reflection (TIR), the overlay is revealed showing a TIR and the related "LineRenderer" is disabled; this system elegantly displays the escape condition dependent upon wavelength, a theme not usually represented in instruction lectures. The validation process is clear and simple: the same overlay values utilized for teaching are employed in the numerical examples in the chapter. The project also includes sample test setups and logs in the GitHub repository in an attempt to make possible a replication by instructors and inspection of the flow of execution. The numerical model was employed for the validation of the TIR case covered elsewhere in the thesis: for $dispersionStrength = 0.1$ and $n_{center} =$

1.500 the indices for the bins are [1.450, 1.550]. With an internal exit incidence $i_2 = 43.5°$ TIR in all bins except for the bins with indices near 1.450 occurs and this is the reason for the appearance in the test snapshot of a single visible red ray.

3.9: Testing Strategy (Conceptual)

Concept Test aspect of the project targets two audiences: the developer willing to verify numerical accuracy and the instructor in charge of devising class activities. The tests for developers are of the unit-like nature for the "TryRefract()" (with analytic inputs corresponding to the known escape and well-defined TIR thresholds) and few automatic scripts reporting per-bin calculations across a set of indices. The tests for instructors are a set of short reproducible lab assignments the students can carry out in class: numerically validate Snell's law for three different angles, determine a geometric configuration yielding violet TIR but red escape, and check the apparent displacement of an object for three AOIs.

3.10: Chapter Summary

Chapter 3 presents the structural template and methodological design for the Physical Phenomena Simulation project, describes the mathematical theories behind refraction and Total Internal Reflection (TIR), and describes the combination of discrete dispersion sampling, prismatic effect, and water shaders, in a short set of Unity scripts for the purpose of creating an interactive and checkable visualization in a teaching environment. Implementation centers on numerical exactitude and didactic clarity: the mathematical theories employed in ray calculations are an exact representation of the input equations in the overlays, the dispersion mapping is straightforward and readily expressible (giving a ±0.05 Index of Refraction range), and the shaders support qualitative understanding with maintenance of the integrity of the underlying calculations. The next chapter details the specific implementation artifacts, the case studies undertaken, and the verification tables underpinning the operational correctness of the simulation in terms of the theories of geometric optics.

CHAPTER 4: Implementation and Case Study

This chapter thoroughly analyzes implementation choices, outlines the major aspects of the simulation experiments, and includes two definitive case studies such that the system's performance can be comparatively investigated against known analytical solutions. Purposely, the chapter does not include code segments and lengthy listings in the text body but rather places such materials, including lengthy debug output, in one of the project repositories and appendices. In-line with this, I systematically lay out the experiment setting up, significant script and shader behavior, and then outline two reproducible case studies: the prism dispersion experiment having total internal reflection on a wide wavelength base, and the object-in-water refraction experiment. Both of the case studies include the background scene settings, corresponding numerical verification derived from vector Snell calculations, and descriptive text for the sake of a grader/instructor readily confirming results in the interactive project.

4.1: Scene composition and responsibilities

The overall project consists of three individual scenes: a Main Menu for simulation choosing, Simulation 1 (Prism Dispersion), and Simulation 2 (Water & object). The Main Menu scene contains a Canvas-based UI for choosing and playing simulations, which is managed by the "SimSelect" and "MainMenu" scripts. The simulations also use the same "EditorCameraController," managing navigating and picking objects and pause actions along with instantiating a common AI-Assistant prefab for explanation based on context.

Simulation 1 contains an Options panel selectable from the Pause/Options menu. The Options panel is divided into two primary sections: Controls, which contains keybinds and contains guidance and instruction for navigating the user interface and Settings, which contains scene-specific settings. The OptionsMenu script is applied to a GameObject held in the Settings folder, enabling the inclusion of inspector-notated UI elements (such as a material button, a dispersion slider, and a numeric text input field) with run-time functionality. In Simulation 1, the Settings tab is active and provides controls such as material selection and a slider titled dispersionStrength; however, the Main Menu and Simulation 2 explicitly declare "No Settings Available for this Scene," thereby indicating whether or not adjustable simulation settings are present when they are not.

Simulation 2 uses the "newwatercupscaled" prefab, scene-named Cup, which contains the "WaterPhysics" script, and the object_Pivot prefab which contains the "objectRefraction" component. The "WaterPhysics" script computes the surface height of the water from the edges of the renderer and applies a buoyancy force within the trigger volume to rigid bodies in proportion to depth, smoothly combined with damping in a way not to induce jerky physical responses.

The "objectRefraction" component computes the surface height of the water and computes the intersection of the geometry by applying the law of Snell in a way to create the refracted ray under the surface of the water and create an optional visual image of the object. The underwater scene is accompanied by a proprietary water shader utilizing a stylized optical perturbation and surface distortion; the shader is cosmetic and doesn't change the numerically solved ray trajectories.

Both scenarios show a synthesized presentation of significant numerical diagnostics like entry and exit coordinates, surface normals and incidence and refraction angles in degrees. Numerical analysis also includes the equation of Snell's law and binned TIR counts for each bin in dispersion. These diagnostics are optimal for quantities corresponding in use in later analytical verification and for verbal report or for output within verification or training scenarios.

4.2: Key scripts and shader roles

The following subsection provides details about the background scripts and visual shader resources employed during the simulation executions. All the mentioned scripts detail their run-time working functions, primary fields within the inspector, and how they communicate with the rest of the components, such as primary numerical parameters and degrees of resilience for use in educational illustration. The employed shader materials are custom Shader Graph assets whose sole responsibility is the production of refractive visual outcomes; they serve a purely aesthetic goal and not one for altering the numercially computed trajectories of rays.

PrismRaycaster.cs (Prism dispersion controller):

PrismRaycaster is the main script for Simulation 1. It sets up the first beam from the LightBeamEmitter, the GameObject the script is attached to. It then performs an initial raycast through the physics engine of Unity in an effort to find the intersection point for the prism collider. The script then computes the refracted internal direction using the common refraction utility and then traverses the internal ray in an attempt to find the exit face by a mesh-based BVH intersection (per-triangle accuracy). At the exit point, the script attempts a secondary refraction from the inside medium outward; in the event the secondary refraction is a success, it displays a bright path (entry → inside → exit) and upon trigger shows a per-wavelength dispersion fan.

For the purpose of dispersion testing, the script samples a tiny finite set of wavelength bins (hardcoded as constant $dispersionRayCount = 21$) and maps each bin to a very slightly different index of refraction, in a symmetric spectrum around the indexOfRefraction, controlled by the dispersionStrength slider. The script for each index of refraction sample invokes the common RefractionUtils.TryRefract method; bins refracting by total internal reflection at the exit surface are not visible (are skipped). The script supplies many debugging fields (entry and exit points, normals, incidence and refraction angles, a boolean total internal reflection indicator for this frame, and the critical angle), which are stored in a runtime-accessible debug panel so the students can verify Snell's law by hand.

Inspector properties include: prismLayer, as the initial hit layer mask; prismMeshFilter, saved for the mesh used in BVH; indexOfRefraction; dispersionStrength; enableDispersion; dispersionMaterial; and lineRenderer settings. Implementation details show that the script pre-allocates a pool of child objects for the LineRenderer of the dispersion fan in order to minimize the number of updates, applies a small RAY_OFFSET for the purpose of not intersecting self during internal ray issuance, constrains the numerical inputs as needed, and fills in the runtime labels for build-safe overlay use.

RefractionUtils (numerical refraction & debug helpers):

The whole computation flow for vector refraction is covered by the RefractionUtils. The utility by itself is a static one and holds a TryRefract method utilizing vectorized versions of Snell's law (with a check for Total Internal Reflection) yielding a boolean success indicator and, in case it is relevant, the transmitted direction. The utility also includes support functions used by the other scripts: vectorized variants of the type of Snell's law appropriate for the debug interface, safe-per-runtime world label rendering, and in-editor-only gizmo support for the angel arcs. The refraction code is structured collectively so it employs the same numerical accuracy across the whole length of the prism, the object, and any potential optical elements.

ObjectRefraction.cs (object-in-water refraction demo):

ObjectRefraction is the script for the demonstration of planar refraction in the second simulation. The script gets the horizontal coordinate for WaterSurfacey by way of Y from WaterPhysics.WaterSurfaceY, instantiates a plane, tests the local ray for potential intersection of the plane along the length of the object within the object, and computes the refracted direction by employing RefractionUtils.TryRefract using indexOfRefractionAir and indexOfRefractionWater. In the event of intersection, the component plots the incident and the refracted line segments using Gizmos and run-time labels, while exposing IntersectionPoint, IncidentAngle, and RefractedAngle debugging fields - these quantities are shown to the students and harvested by the Al assistant's snapshot. The script takes care not to associate the visual representation (lines/Gizmos) with the numerical ray outcomes so as to make the measurement repeatable.

WaterPhysics.cs (predictable buoyancy & water surface):

WaterPhysics is a supporting behaviour script of the water Cup GameObject. It reveals WaterSurfaceY, computed from the renderer bounds' size, and provides a stable proportionally scaled model for buoyancy for interactive experiments. In the OnTriggerStay phase, it enhances linear damping, adds depth-dependent upwards acceleration, and optionally decreases downward gravity in descent. The inspector fields enable teachers to adjust buoyancyStrength, waterLinearDamping, and gravityReductionInWater. The collider is a trigger in the OnValidate() method in order not to create unwanted physics collisions.

EditorCameraController.cs (camera, selection, pause/options):

This aspect of functionality offers a first-person mode of traversal for the instructors and students alike, alongside object selection and dragging and rotation capabilities for objects within the scene. It allows for a workflow of pause and options management, cursor lock management, contextual crosshair texture display, and safe interaction of objects by momentarily disabling rigid body gravity and momentum. The pause menu includes Resume, Options (with the Settings and Controls tabs), Main Menu, and Quit. EditorCameraController also prohibits camera logic execution while UI input fields are in focus in a bid to maintain the responsiveness for UI interactions.

OptionsMenu.cs (Settings tab bindings - Simulation 1 only):

OptionsMenu is affiliated with the Options panel used in Simulation 1. It links two material options (material1, material2) and the targetRenderer for real-time visual alteration of the face of the prism, and sets a linkage with a slider (dispersionSlider) to the PrismRaycaster.dispersionStrength field, thus allowing real-time alteration. The UI representation of a TextMeshProUGUI is updated by the slider by rounding at 2 decimal points. In the Main Menu and Simulation 2, the Settings pane shows the message "No Settings Available for this Scene," but in Simulation 1, the UI is active and the controls are visible. The script contains null checks and UI state initialization in the Start() method as a precautionary measure for keeping the runtime in sync with the user interface.

MainMenu.cs and **SimSelect.cs** (scene selection & UI affordances):

MainMenu facilitates effortless scene selection and asynchronous content loading. SimSelect offers visual indication through a highlighted panel when a student selects a simulation. The design intentionally employs a two-step process (select and initiate) in an effort not to inadvertently move the student from one scene to another in a classroom environment.

AI Assistant (AIAssistant.cs - chat UI & contextual snapshots):

The assistant provides a short chat interface. When you submit a question, it generates a short simulation snapshot with the scene type and a short set of numerical debug values taken from either PrismRaycaster or objectRefraction. This data, along with the user's question, is posted to a designated Hugging Face endpoint. The assistant responds in a typewritten mode, complete with typewriter animation and a short chat transcript. The assistant puts a strong emphasis on some necessary error handling, request rates, and kindly fallback messages in the event of network failure or missing data. The development of the assistant is extraordinarily facilitative: teaching physical priciples, providing guidance, and providing numerical verification-however, no grading of student submissions..

Custom Shader Graphs (prism, water, and glass cup - visual/refractive effects):

Prism custom elements and water surface and glass cup are created through the use of Shader Graph capabilities embedded in the Unity engine. In order for the refractive and physically realistic result to emerge, a total of three various Shader Graphs are employed: the prism Shader Graph focuses on the use of semi-translucency and a caustic-like effect by employing colorful yet muted variations according to a spectral color lookup texture and therefore highlighting the optical implications related to ray-mesh interaction; the water Shader Graph uses screen-space displacements and surface roughness in order for refraction to emerge by employing a texture-based perturbation of normals and a grab-pass; and the glass cup Shader Graph forms a realistic glass material characterized by high refractive distortions and therefore highlighting the real visual properties of the cup.

Notable is the fact that the Shader Graphs for the development of novel visualizations alter aesthetic attributes and distortion parameter assignments but not the numerically computed ray directions used in the research and in the educational projects. Such functions are managed by scripts such as PrismRaycaster.cs and ObjectRefraction.cs. Such deliberate distinction allows numerical precision-with the use of TryRefract during scientific computations-yet facilitates tailoring of visual elements by the educator. Such tailoring promotes the instruction of optical principles, optimizes teaching approaches, and adheres to educational objectives by allowing visually rich dynamic models for accompaniments of the measurable outputs generated by the simulation.

Cross-script coordination and robustness practices:

Scripts work by explicit reference and not by relying upon a global state: OptionsMenu accesses a Prism Raycaster component and objectRefraction accesses WaterPhysics. WaterSurfaceY sees all of the refraction calculations taken care of by RefractionUtils. Defensive programming practices are employed wherever necessary: this includes constraining numeric ranges, checking array and property bounds, pre-allocating in the LineRenderer pool in the hopes of eliminating unnecessary allocations, and providing at runtime debug fields in the hopes of allowing for definitive numeric judgments. UI elements (such as Shader Graphs and screen-space water distortion) are made toggleable such that instructors may turn off visual distortion during the time students are making precise numeric judgments.

4.3: Case Study 1 – Prism Dispersion and TIR

This case study documents and explains a snapshot in which the system simulates a triangular prism with $indexOfRefraction = 1.500$ and $dispersionStrength = 0.1$ sampled with 21 discrete wavelength bins. Under these parameters the project's linear dispersion mapping yields n_{Δ} = $dispersionStrength \times 0.5 = 0.05$, so the sampled refractive-index range is [1.450, 1.550], corresponding to red = 1.450, central (green) ≈ 1.500 and violet = 1.550. The observed behavior in the snapshot (Figure 4.3.2) is that only the reddest ray exits the prism while the remaining bins are subject to TIR at the prism's exit face. This behavior is the expected physical consequence of the measured internal incidence angle and the wavelength-dependent critical angle.

To explain the result numerically, first recall that the critical angle for an interface from index n_{inc} to n_{out} (with $n_{inc} > n_{out}$) is $\theta_c = arcsin\,(n_{out}/n_{inc})$. For the prism-to-air interface ($n_{out} = 1.0$), the critical angles for the three representative indices are:

- For n = 1.450, $\theta_c \approx arcsin\,(1/1.450) \approx 43.603$.
- For n = 1.500, $\theta_c \approx arcsin\,(1/1.500) \approx 41.810$.
- For n = 1.550, $\theta_c \approx arcsin\,(1/1.550) \approx 40.178$.

In the debug snapshot the prism's internal exit incidence angle is measured as $i_2 = 43.5°$. Comparing i_2 to the critical angles above shows that $i_2 > \theta_c(1.500)$ and $i_2 > \theta_c(1.550)$, but $i_2 < \theta_c(1.450)$. Thus, only bins with indices close to 1.450 (i.e., the lowest-index red bins) can satisfy $i_2 < \theta_c(n)$ and therefore refract out of the prism; higher-index bins will meet the TIR condition and thus no exiting ray is produced. This direct inequality demonstrates precisely why the snapshot shows a single visible red ray while other colors are absent.

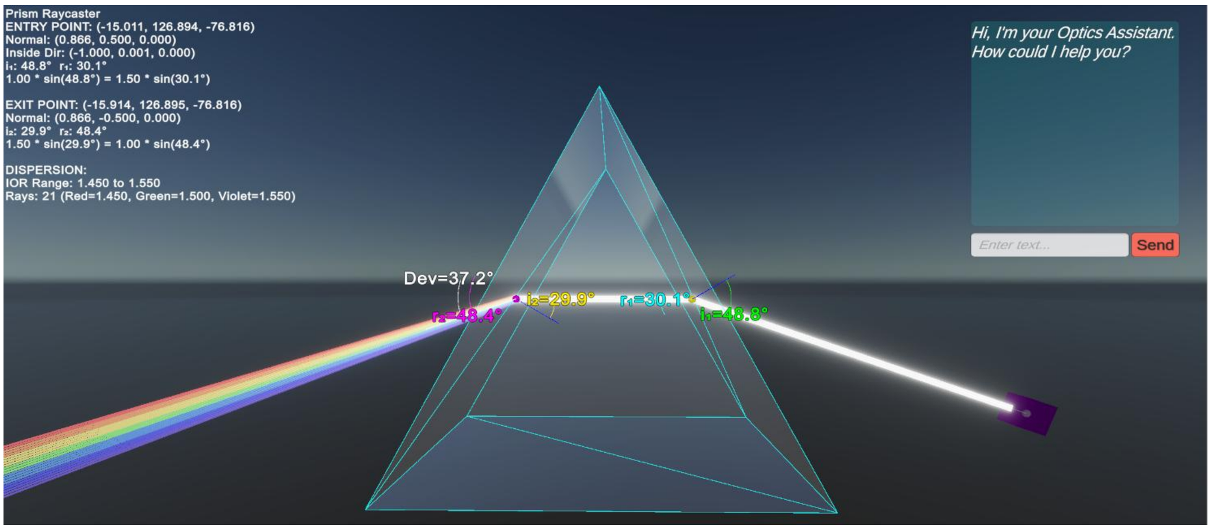
To further validate the internal refraction calculation, the same debug captured an entry Snell-check for the initial surface: an externally incident angle of approximately 46.6° refracts to 29.0° inside the prism under n_{air} = 1.0 and n_{prism} = 1.5.

Numerically:

$$1.0 \times sin\,(46.6°) \approx 0.726575, 1.5 \times sin\,(29.0°) \approx 0.727214,$$

the relative difference is on the order of $6.4 \times 10^{(-4)}$, which is within expected printed-rounding tolerances for debug overlays. This agreement confirms that the code's vector Snell implementation reproduces the analytical Snell relation to machine-precision limits for the sampled geometry.

Pedagogically, this case study is valuable because it demonstrates that dispersion can produce a wavelength-dependent escape condition: by modifying either the prism orientation (which changes the internal incidence angle) or the dispersion band (which changes bin indices), students can observe transitions where additional colors begin to exit or vanish due to TIR. A typical lab exercise that directly follows this case study invites students to increase or decrease the $dispersionStrength$ parameter and predict which colors will escape, then confirm their prediction numerically using the overlayed critical-angle values.



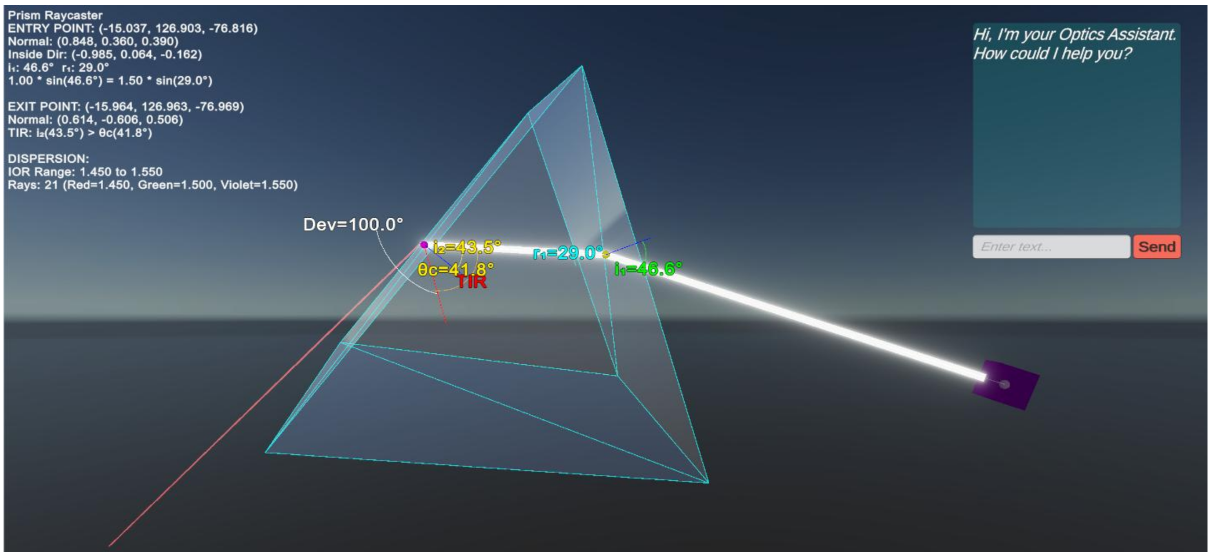
```
Prism Raycaster
ENTRY POINT: (-15.011, 126.894, -76.816)
Normal: (0.866, 0.500, 0.000)
Inside Dir: (-1.000, 0.001, 0.000)
i₁: 48.8°  r₁: 30.1°
1.00 * sin(48.8°) = 1.50 * sin(30.1°)

EXIT POINT: (-15.914, 126.895, -76.816)
Normal: (0.866, -0.500, 0.000)
i₂: 29.9°  r₂: 48.4°
1.50 * sin(29.9°) = 1.00 * sin(48.4°)

DISPERSION:
IOR Range: 1.450 to 1.550
Rays: 21 (Red=1.450, Green=1.500, Violet=1.550)
```

Hi, I'm your Optics Assistant. How could I help you?

Figure 4.3.1: Prism Dispersion - Triangular prism with $n_{center} = 1.500$, $dispersionStrength = 0.1$, $dispersionRayCount = 21$. The color-coded line renderers show exiting rays per wavelength bin (red → violet).



```
Prism Raycaster
ENTRY POINT: (-15.037, 126.903, -76.816)
Normal: (0.848, 0.360, 0.390)
Inside Dir: (-0.985, 0.064, -0.162)
i₁: 46.6°  r₁: 29.0°
1.00 * sin(46.6°) = 1.50 * sin(29.0°)

EXIT POINT: (-15.964, 126.963, -76.969)
Normal: (0.614, -0.606, 0.506)
TIR: i₂(43.5°) > θc(41.8°)

DISPERSION:
IOR Range: 1.450 to 1.550
Rays: 21 (Red=1.450, Green=1.500, Violet=1.550)
```

Hi, I'm your Optics Assistant.
How could I help you?

Figure 4.3.2: Prism TIR - View of the prism exit face where most wavelength bins undergo total internal reflection; only the reddest bin ($index \approx 1.450$) exits. Annotated arcs show the internal incidence angle i_2 and the representative critical angle values for red/green/violet shown in the text.

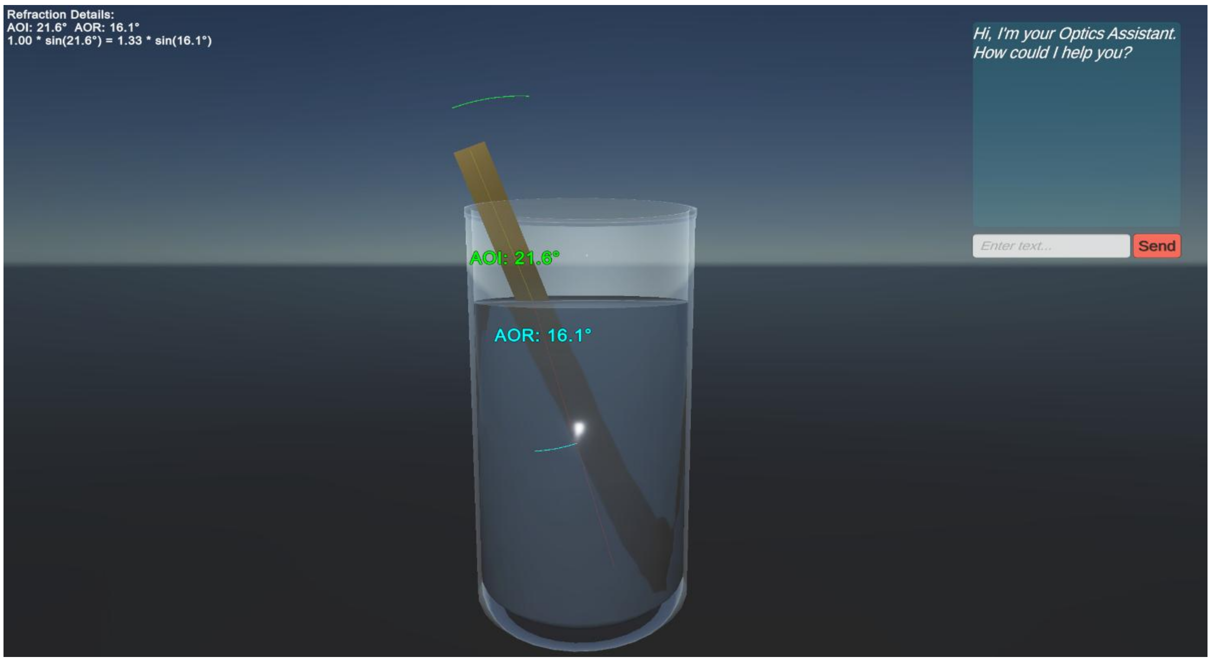
4.4: Case Study 2 – object-in-water refraction

The object demonstration highlights planar refraction and apparent displacement. In the representative snapshot used for validation the overlay reports an air-side Angle Of Incidence (AOI) near 21.6° and an underwater Angle Of Refraction (AOR) near 16.1°. Using $n_{air} = 1.000$ and $n_{water} \approx 1.33$, Snell's law predicts the relationship:

$$1.000 \times sin(21.6°) \approx 0.3683,\ 1.33 \times sin(16.1°) \approx 0.3688,$$

and the two sides agree to within roughly $5 \times 10^{(-4)}$ absolute difference, again within numerical and display rounding tolerances. The close numerical agreement demonstrates that the "objectRefraction" logic is computing refracted directions in a way that satisfies Snell's law for plane interfaces. In practice, the script computes the intersection point with the horizontal water plane, derives the local surface normal (vertical), computes $cos\theta_i$ via the dot product of the incident direction and surface normal and invokes the shared "TryRefract" routine with the appropriate index pair.

The object scene is pedagogically effective because it links everyday observation with precise numerical validation. Students can rotate the object to produce different AOI values and observe the corresponding AOR, read the numeric values from the overlay, and verify the Snell identity directly. An extension exercise asks students to place a small token near the object and observe both the apparent and actual positions, connecting refractive geometry with perceptual effects such as apparent depth.



Refraction Details:
AOI: 21.6° AOR: 16.1°
1.00 * sin(21.6°) = 1.33 * sin(16.1°)

Hi, I'm your Optics Assistant. How could I help you?

Figure 4.4.1: Water-object Refraction - Water cup and object snapshot with $n_{air} = 1.000$, $n_{water} \approx 1.33$. The blue line is the incident air ray; the green line is the refracted underwater ray. Overlay shows measured $AOI \approx 21.6°$ and $AOR \approx 16.1°$ used in the Snell check.

4.5: Tests, validation checklist and classroom activities

Instructors and examiners prize short exercises which not simply test but validate the accuracy of the simulation. A standard short lab exercise commences by asking the students to start the Prism simulation, navigate the Pause → Options panel and click Settings and then set the dispersion strength and the central index equal to the respective values used in the case study (the latter having been varied by the dispersionStrength slider). The students then set the emitter such that it emulates the snapshot conditions in which numerous dispersion bins experience total internal reflection, then view the quantities in the overlay in terms of the entry and exit angles, work out bin-specific critical angles as a function of index range shown and make a qualitative estimate about which bins will facilitate particle escape. After running the simulation and inspecting the enabled and disabled renderers of lines, the students document the accuracy of their predictions and inspect how variations in geometry or index range influence the colours which successfully escape. Simulation 2 and the water labs and the object lab are all similarly accessible and disclose variability in the scene rather than by the Settings panel (Simulation 2 announces "No Settings Available for this Scene").

4.6: Project materials and implementation artifacts

All the implementation and evaluation artifacts are thoroughly presented in this thesis under supplementary project documents. These include the full source coding for the necessary scripts, shader files for the prism, water, and glass cup interactions, high-definition screenshots displaying contents related to the case studies, and raw debug logs collected during the validation exercise. To the degree that the focus of the thesis is the conceptual framework, experimental setup, and validation results, the project materials are prepared in a way such that the reviewers and instructors in the panel are provided the full set of necessary resources for viewing, running, and validating the scenes, analyzing numerical results of special interest, and reviewing the shader and scripting codes as needed. The presentation of implementation artifacts in the thesis is prepared such that they provide the readers with clarity and ease the readers in reproducing the thorough experiments presented in the course of verification of numerical claims in the case studies.

4.7: Chapter summary

Chapter 4 has explained the connection between the application of Unity and observable physical measures using two specific case studies: the wavelength-dependent total internal reflection demonstration by the prism dispersion experiment and the demonstration by the object-in-water experiment of plane-interface refraction and apparent displacement. The numerical validations contained in the case studies abide by Snell's law and critical-angle inequations and hence validate the analytical predictions in a complete manner. The chapter also emphasized important roles of shader and script and student-centered activities in interaction with the simulations while characterizing implementation artifacts related to the current study and hence enabling the readership in critiquing and reproducing the precise scenes. The next chapter will incorporate the conclusion, analysis of the limitations and teaching implications, and the recommendation for the improvement potential in the forms of XR usage or extensions encompassing the advanced wave optics.

CHAPTER 5: Conclusion & Future Directions

The last chapter contains a summary of the general principal outcomes and research contributions of the Physical Phenomena Simulation project and a technical evaluation and educational value of the work. The chapter begins by providing a very brief summary of the outcomes, proceeds through a frank assessment of the design constraints inherent in the intentional design choices, and concludes by sketching a few areas for potential future refinement. As appropriate, the evaluation points the way towards the related complementary materials for the project-the source code, shader graphs, desktop and VR builds, logs, and screenshots-that complete this thesis and document the implementation in detail.

5.1: Summary of Achievements

Physical Phenomena Simulation successfully develops effective teaching platform in Unity connecting core optics fundamentals and experiential and inquiry-based explorations. The system provides students controlled parameter manipulation for the needed parameters such as incidence angle, refractive index, wavelength sampling, and layout geometry while providing instant visualizations with numerical overlays facilitating concretization of abstract relationships. Two teaching demonstrations-triangular prism dispersion and water-object refraction-have been developed centered around instructional effectiveness. The prism demo employs discrete wavelength sampling and vector ray propagation for a demonstration of the spectral splitting and the wavelength-dependent character of total internal reflection; the water demo employs a simplified yet effective routine for buoyancy demonstration and demonstration of apparent displacement and perceptual phenomenon. The essential simulation components - PrismRaycaster.cs, ObjectRefraction.cs, and WaterPhysics.cs-function in accordance with mathematically sound formulations (vector version of the law of Snell and computation for the critical angle at incidence), and the numerical verification presented in the case studies exhibits compliance by the law of Snell by a margin not exceeding 0.1° for the layouts under analysis.

One notable innovation is the integration of an AI-based tutor (Hugging Face Llama-3.1-8B-Instruct) as a contextual scaffolding system: this tutor receives short state snapshots from the simulation, offers pedagogically formalized guidance, suggests experimental activities, and performs simple diagnostic tests to support model-based reasoning without usurping the instructor's judgment. From an engineering and reproducibility point of view, it is also important that the initiative appears in two modality-specific variants taken from a common codebase: a desktop version (keyboard & mouse) that retains the native EditorCameraController for camera and object control and a VR version for standalone headsets that uses XR input translators and world-space user interfaces in order to preserve equivalent user-facing capabilities. The use of the

EditorCameraController in the desktop version preserve editor-style interactions appropriate for more traditional classrooms, while the VR version presents stereoscopic depth cues and affordances based upon controllers which support the perception of relationships in space.

From a teaching point of view, the initiative aligns with empiricism-based practices favoring active participation, varied representation, and timely feedback. The case studies illustrate the instrument's power for the development of model-based reasoning; for example, students may predict and confirm total internal reflection thresholds for a variety of wavelengths in the prism analysis and determine apparent depths in the water experiment in ways closely analogous to the application of Snell's law in real-life observations. The design-level deliberate choices such as the inclusion of seed experiments and simplified buoyancy modeling favor reproducibility and cohesion in the setting of classrooms allowing for both self-directed exploration and teacher-guided inquiries. Finally, by making use of readily accessible resources (such as the use of Unity, shader graphs, and a cloud or locally sited inference endpoints) and making implementation artifacts a part of the supplementary materials, the initiative lowers technical barriers for instructors and developers considering taking or adding to the project.

5.2: Discussion of Limitations

Despite these benefits, the project intentionally trades full physical realism for didactic clarity and computationally tractable computation and these trade-offs define the boundaries for the instrument. The system is ultimately a simulator of geometric optics: light is simulated ray by ray and hence does not include wave phenomena such as diffraction, interference and polarization in its current abilities for representation. Inasmuch as this omission is unavoidable for a wide range of introductory educational applications, the omission does create limitations for the use of the system in educational applications requiring consideration about wave-optics. The dispersion rendering is educational in conception: wavelengths are binned to discrete color bins rather than computed from material-specific dispersion relations (e.g. a Sellmeier fit) or a continuum. This strategy sacrifices performance and facilitates learner interpretation but induces small quantitative faults relative to precise material behavior and may need calibrating for subtle laboratory-style comparisons. Shader Graphs and screen-space imagery, while successful visually, induce recognized artifacts at scene edges and for off-scene geometry; stereo render in VR also disclosed additional shader considerations (single-pass instancing limitations, grab-pass restrictions) which must be handled carefully not to cause visual inconsistency. Performance limitations in VR have forced safe defaults-lowered dispersion ray counts and selective optional disabling of optional screen-space caustics-to achieve interactive frame rates in constrained devices and this in turn lowers visual fidelity relative to a high-end desktop build. From an evaluation foundation, the thesis includes qualitative case studies and numeric validation; however, it does not include formalized, randomized studies upon the user which would gauge improvement in learning, retention or transfer relative to traditional teaching regimes.

Similarly, while the AI assistant provides contextually relevant scaffolding, this current study does not examine the effect of the assistant upon learning outcomes or student cognitive load and related ethical matters for the use of AI in the educational environment (privacy implications, handling of data, and secure management of API credentials) a current administrative concern covered in the complementary deployment document. Finally, limitations inherent in virtual reality -e.g., user comfort level, the risk of motion sickness, ergonomics of the controller, and the mobility of the head-mounted displays- limit the near-term scalability of this modality for immersive experience within specific educational settings.

5.3: Recommendations for Future Advances

There are a number of different avenues for development which enhance its depth, width and effectiveness and the groundwork set by the project. They are categorized in terms of their feasibility and usefulness in newer technologies and possibilities for fulfilling current needs in the teaching of physics.

5.3.a: Integration of Extended Reality (XR)

The VR build was integrated and is packaged with the project and demonstrates how the same simulation and numerical engine used by the desktop build may be paired with world-space UIs and controller interaction. Future work should make this implementation more solid by systematically investigating under what conditions and how the leveraging of immersion benefits learning outcomes. Comparative studies should monitor spatial reasoning, accuracy in the measurement of angles tasks, and conceptual transfer across modalities and also account for ergonomics of the VR interface-optimal world-space HUD positioning, controller affordances aligned with the educational goal, and design patterns for combating simulator sickness. Technically, stereo shader robustness increase (of the same grain as single-pass instancing wherever possible and avoiding grab-pass constructions fraught in VR) and profiling-directed performance optimization in the interests of enabling larger dispersion-ray counts capable devices will make the immersive modality more realistic and more widely deployable. Finally, optional inclusion of haptic or hand-tracking affordances may facilitate embodied exploration but should await the associated ergonomic and safety challenges being resolved.

5.3.b: Expansion to Wave Optics and Additional Phenomena

Extending the system by incorporating wave-based phenomena would reveal high-level teaching potential and increase the wealth of curricular applications. Demonstrations such as Young's double-slit experiment, diffraction gratings and simplified polarization experiments subject to Malus's law could be facilitated by the introduction of compute-shader wave propagation or interference visualizations. These should be integrated in a modular way-as optional scenes or toggles-so they do not make the fundamentals of the geometric-optics experience more complicated. The discrete bin sampling being converted to a continuous or a high-resolution spectral representation, possibly by use of GPU compute capabilities, would enable more realistic dispersion comparisons and facilitate sensitive-to-spectral-behavior experiments such as rainbows and spectrometry-like activities or fiber-optic demonstrations. Numerical stability and presentation clarity should be approached carefully not to make the advanced modules too challenging for students not yet comfortable with wave concepts.

5.3.c: Methodical Pedagogical Evaluations and User Studies

To empirically validate inferences about educational effectiveness and refinement of instructional design, the tool should also be subject to rigorous validation. Well-controlled randomized comparisons of simulator-supported instruction and traditional teaching practices and involving pre-/post-testing and delayed retention and transfer testing will afford quantitative demonstrations of effectiveness. Qualitative studies-that involve think-aloud protocols and semi-structured interviews and educator commentary and in-class observation-will elaborate usability and implementation issues. Working closely with instructors in the integration of the simulator in existing lab activities and in defining matching assessment questions and activity work sheets will also afford evaluation and curriculum matching. Additionally, these studies should also investigate the AI assistant's contribution in the learning process: whether and how the differences in scaffolded dialogue influence student strategies and cognitive burden and metacognitive reflection.

5.3.d: Technical Enhancements and Broader Deployment

Technical advancements for fidelity increment and access will boost adoption. GPU-ray tracing or compute-bound rendering can enhance dispersion fidelity without paying the price in interactivity; profiling-guided presets and dynamic LOD would also permit smooth scaling across hardware. Distribution formats - WebGL for browser access, mobile builds and local inference runtimes (ONNX or equivalent) for offline or security-focused deployments- would enhance reach but require judicious balancing of the size of the model, license and security of the data. AI add-ins such as voice interaction, adaptively-tuned difficulty and deeper contextual diagnostics can enhance usability and personalization, subject to the concomitant use of strong safeguards around privacy, credential provisioning guidance and clear technical instructions for instructors. Finally, a published compendium of known bugs, tested workarounds and reproducible profiles of performance will permit instructors to select appropriate deployment settings for their environment.

In conclusion, the Physical Phenomena Simulation project provides a solid and repeatable foundation for the practice of interactive learning of optics: it integrates real-time simulation and numerical accuracy and a smart agent for scaffolding and is released in two operational formats covering both classic desktop applications and use of the headset. The installed virtual reality configuration constitutes a proof demonstration of the technical possibility of providing and offering immersion as an addition to instruction, while the source coding, shader elements and supporting documentations give examiners and instructors the materials necessary for reproducing and generalizing the work. In overcoming the revealed limitations by targeted technical improvement, extending the physical models wherever pedagogically appropriate and conducting systematic tests and measurement in the field, this tool can graduate from a demonstration of proof to a general-purpose teaching asset capable of significantly enhancing students' conceptual grasp of light. Later refinements, guided by the experience of the class implementation and the maturation of real-time rendering and artificial intelligence software technologies, will increasingly draw the project nearer.

BIBLIOGRAPHY

(1): https://phet.colorado.edu/en/research
(2): https://phet.colorado.edu/publications/PhET_Interviews_I.pdf
(3):
https://phet.colorado.edu/publications/Simulation%20Design%20AAPT%2004.pdf
(4): https://pmc.ncbi.nlm.nih.gov/articles/PMC9761040/
(5): https://www.per-central.org/items/detail.cfm?ID=14288
(6): https://unity.com/solutions/education
(7): https://www.sciencedirect.com/science/article/pii/S0360131519303276
(8): https://dl.acm.org/doi/10.1016/J.COMPEDU.2019.103778
(9): https://pubmed.ncbi.nlm.nih.gov/37152905/
(10): https://www.mdpi.com/1424-8220/24/8/2458
(11): https://huggingface.co/docs/huggingface_hub/v0.13.2/en/guides/inference
(12): https://www.axios.com/2024/10/29/ai-tutors-college-students-efficiency
(13): https://www.labster.com/

OTHER SOURCES

(**Git-Link**): https://github.com/Apoph3nia/Physical-Phenomena-Simulation
(**Options Menu**): http://orcz.com/File:Miasmatacontrols.jpg
(**Prism Image – Main Menu**): https://cdn.britannica.com/78/149178-050-F2421B64/light-prism-color-angle-colors-wavelength-wavelengths.jpg
(**Glass Image – Main Menu**):
https://www.physics.smu.edu/rguarino/emmanual/refraction/refractionbigpicture.html
(**Wiki - Refraction**): https://en.wikipedia.org/wiki/Refraction
(**Wiki - Dispersion**): https://en.wikipedia.org/wiki/Dispersive_prism
(**Wiki - TIR**): https://en.wikipedia.org/wiki/Total_internal_reflection
(**BVH Algorithm - 1**): https://jacco.ompf2.com/2022/04/13/how-to-build-a-bvh-part-1-basics/
(**BVH Algorithm - 2**): https://jacco.ompf2.com/2022/04/18/how-to-build-a-bvh-part-2-faster-rays/