1 Analysis

1.1 Introduction

At Bacup and Rawtenstall Grammar School (BRGS), the nature and behaviour of light are taught at GCSE and A-level and are important topics to understand, both for examination purposes, and also to learn about other areas of physics that build on it. The teaching of the behaviour of light in physics lessons requires a lot of notes, diagrams, practice questions and some practical work.

Each student keeps notes from each lesson which include key points to remember, diagrams drawn by the teacher and their attempts at any questions set with some corrections. Students also receive some important diagrams and key points as printed handouts to save time writing out or drawing (particularly for detailed diagrams).

1.1.1 Explanations Involving Diagrams

I have observed that teachers frequently draw diagrams to aid their explanations or use images and diagrams from the Internet on slideshow presentations – these may even include moving two-dimensional diagrams (animated gifs) in a small proportion of cases. Figure 1.1a is a two-dimensional diagram used to show how a convex/converging lens works, part of the GCSE physics syllabus at BRGS.

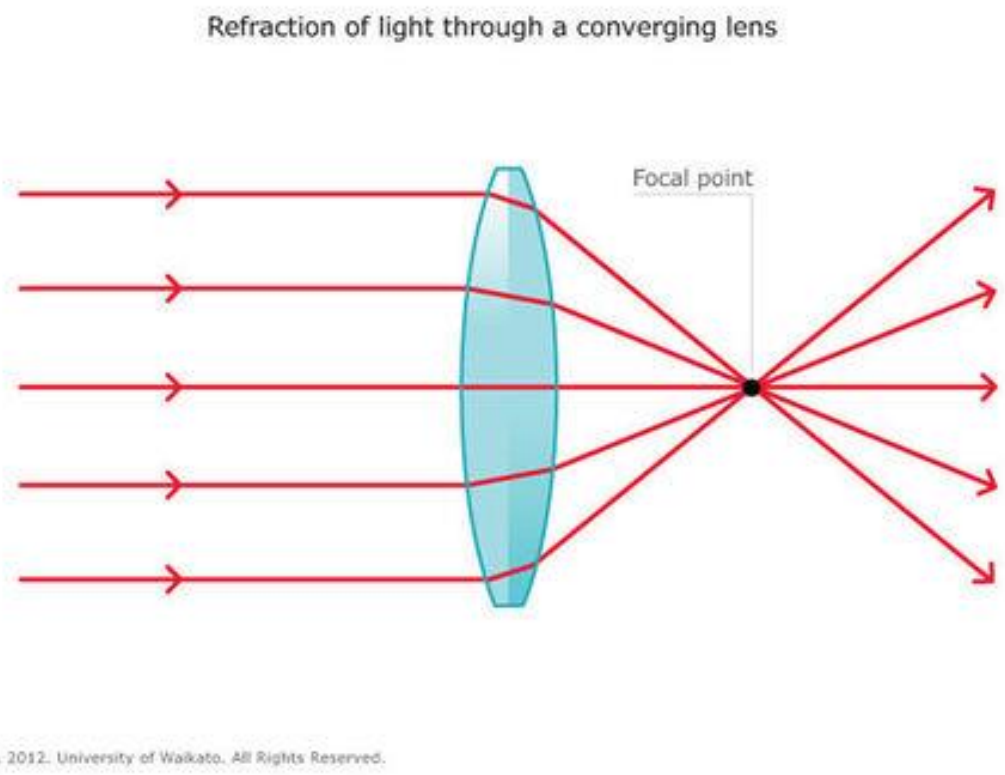


Figure 1.1a – a two-dimensional cross-section showing how a convex/converging lens focuses multiple light beams to a single point (image from www.sciencelearn.org.nz/var/sciencelearn/storage/images/contexts/light-and-sight/sci-media/images/converging-lens/685327-1-eng-NZ/Converging-lens_gallery_supersize_landscape.jpg)

The problem with diagrams like this is that they do not represent the three-dimensional shape of the lens and the convergence in all three dimensions that this causes. A three-dimensional diagram of some sort would be more useful in conveying this, but it is difficult to express three dimensions in a still two-dimensional image; several viewing angles are needed to give a proper sense of depth. This problem can be solved with a three-dimensional animation, but relevant animations are very hard to come by and do not allow users to focus on a particular area of the scene. A three-dimensional image or animation can also be over complicated and less clear than a two-dimensional image due to distortions (mainly that objects further away appear smaller) and a lack of some important viewing angles such as a side-on view (similar to figure 1.1a).

As there is currently no solution to these problems, some students may not understand key ideas and consequently have difficulty tackling the questions. Alternatively, these problems may mean that more of the lesson time needs to be devoted to explanation and answering students' questions, rather than actually practicing calculations and other questions that may be asked in an examination. Either way, these problems are likely having a negative impact on students' examination scores.

A three-dimensional interactive application could theoretically solve all of these issues by allowing the user to control their view of the three-dimensional scene and having smooth transitions between views to allow the user to perceive depth in the same way a 3-D animation would.

Figures 1.1b and 1.1c below show and explain an interactive web application that my client, Mr. Wilkinson, uses to teach total internal reflection to GCSE and A-level students.

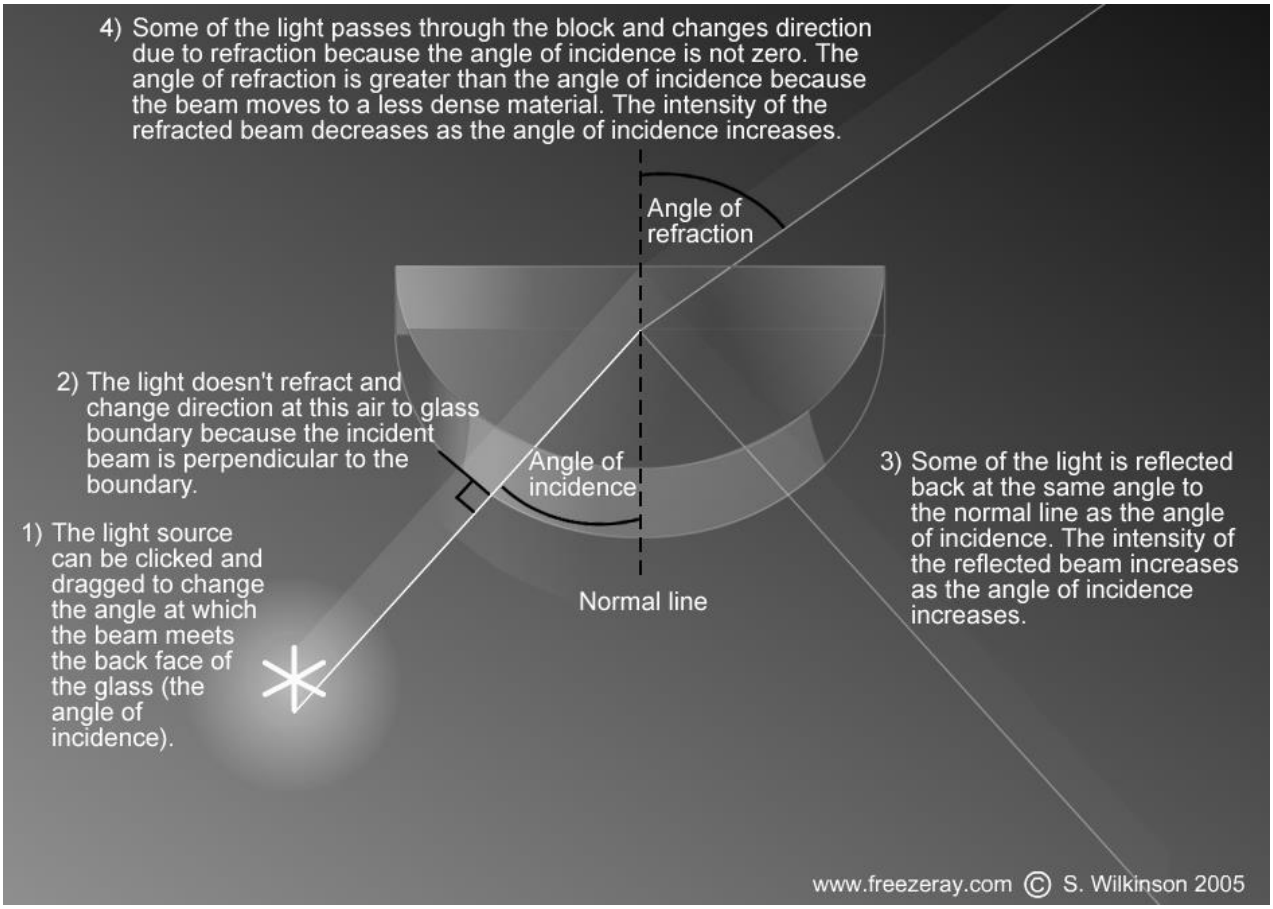


Figure 1.1b – an interactive flash diagram showing refraction and total internal reflection (screenshot from www.freezeray.com/flashFiles/TotalInternalReflection.htm with explanatory annotations by me)

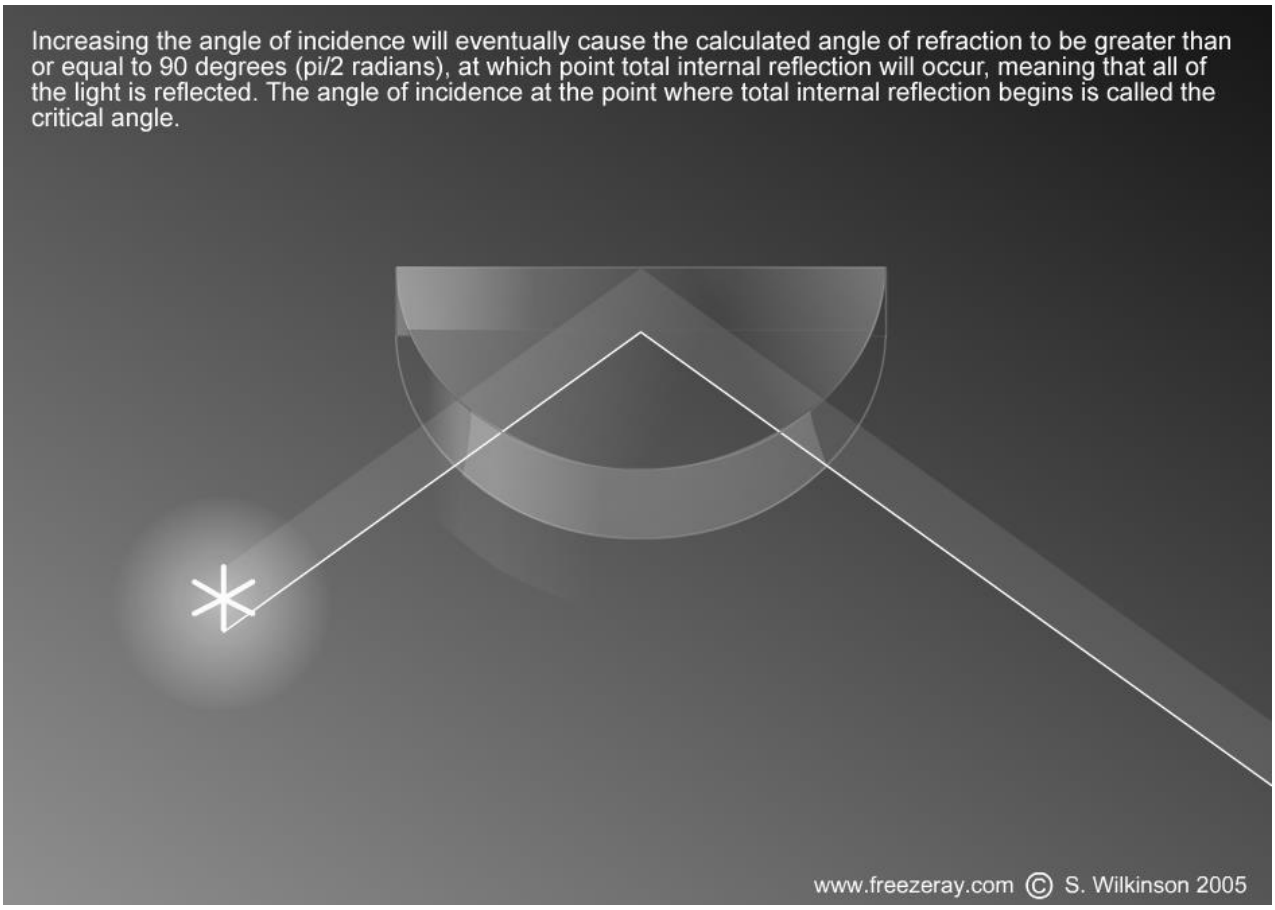


Figure 1.1c – another screenshot from www.freezeray.com/flashFiles/TotalInternalReflection.htm with explanatory annotations by me

Although this is an interactive application and has a 3-D appearance, it is restricted to a single view, so users cannot accurately perceive depth and therefore the positions of objects in a three-dimensional space. Also, this application is restricted to this one glass block and a single light beam that must lie within a plane, meaning that a three-dimensional view of this scenario is no more useful than a two-dimensional view. Ergo, this application is unsuitable for learning about other shapes such as lenses, transitioning to a denser material (when the angle of incidence is not $0°$), or about the behaviour of light in three-dimensions.

1.1.2 Practical Work

The main practical work is an investigation into the refractive index of glass relative to air. Each student is given a glass block, ray box and power pack and they trace around the glass block on a sheet of paper. Students then set up the ray box to shine a beam through the glass block and on the paper draw a line from the ray box to the glass block and another line from the other side of the glass block to show where the beam shines on the paper. Removing the block and drawing another line to show the path of light through the block allows the angles of incidence and refraction to be measured with a protractor. This experiment is repeated with different positions for the ray box and for each run the students calculate the relative refractive index of the glass by using Snell's law:

$$\mu = \frac{\sin(i)}{\sin(r)}$$

$$where\ \mu\ is\ the\ refractive\ index\ of\ glass\ relative\ to\ air,$$

$$i\ is\ the\ angle\ of\ incidence\ and\ r\ is\ the\ angle\ of\ refraction$$

It is often useful for students to compare their results from a practical investigation with the theoretical results in order to observe the amount of experimental error that is present or how much the refractive index of glass (this is always the material used at BRGS) deviates from its typical value. The main source of experimental error is in the measurement of angles with a protractor and the refractive index of glass will also deviate somewhat from its expected value because there are different types of glass as well as glass manufacture not being entirely consistent. Furthermore, students only have the facilities to experiment with light travelling from air to glass, rather than from a vacuum, meaning that a relative refractive index is calculated – air has a refractive index of about 1.0003 (compared to 1 for a vacuum), so the difference from the absolute refractive index of glass due to this should be very small.

Comparing the mean refractive index calculated from the experiment, or even the refractive index calculated from each run to the typically stated value is straightforward, but to compare the angles of refraction from each run to the theoretical angles of refraction currently requires repetitive calculation by the students which could be quickened and made less error prone with an interactive application that takes numerical inputs and gives numerical outputs (unlike Figure 1.1b).

1.1.3 Practice Questions

Students answer questions relating to the topic by applying what has been dictated – these questions are usually one of two main types: explanation or calculation. Practice questions mostly come from past examination papers, but these are limited in number and may not represent everything that could be asked according to the examination board's specification. This means that teachers often write their own questions and sometimes must ensure that the numbers in the question correlate. This process is error prone and time-consuming; when done incorrectly, students can find themselves with impossible or non-sense questions.

The teacher may give answers to questions once the students have had the opportunity to answer them, but usually there isn't time to give answers to all of the questions, perhaps meaning that students don't learn from some mistakes they are making. For past paper questions, answers come from the official mark scheme, and for questions devised by the teacher they will usually have answers, otherwise the teacher asks what answers the students obtained in order to find the modal answer which is likely to be correct if the majority of students agree. Teacher answers and class majority answers may not always be correct.

Teachers can also set more challenging questions for bright students, although they are more difficult than anything that could be asked in an examination. Students may also choose to set themselves more challenging questions outside of lessons.

An interactive application could help teachers to construct calculation questions that make sense (the refractive index of any material cannot be less than 1, for example) and provide reliable answers at the same time. This application would also allow students to mark their answers to questions, and a 3-D application could help produce more challenging questions and answers involving angles in three dimensions.

1.2 User Needs and Acceptable Limitations

Physics teachers primarily carry out the tasks of producing teaching notes, explaining concepts to students, producing handout notes and diagrams and producing practice questions and answers. The Physics teachers who teach areas of the syllabuses relating to the behaviour of light will be the main users of the new system. This group of users would like the new system to aid their explanations, produce clear two and three-dimensional diagrams to be printed for each student and to provide data for setting calculation questions as well as providing answers to them.

The main tasks for each physics student are to produce revision notes, answer practice questions from teachers and past papers and to check their answers to these questions. The students will also be targeted users of the new system not only because they will observe the diagrams produced by the new system, but also because they may use the system to calculate theoretical practical results for comparison with theirs, and students who are willing may use the software outside of lessons if they don't fully understand what has been taught or wish to set themselves more challenging questions such as calculating angles of refraction in three dimensions. This group of users would therefore like new system to produce clear diagrams with a sufficient amount of detail and also to have an intuitive and user-friendly interface.

The students' lack of introduction to the software will be a limitation because there wouldn't be time to teach each physics student to use the system and they most likely would not read the user manual even if it were readily available. Therefore, the system should be user friendly, as the students' only previous experience with the software will be observing their physics teachers use it.

The school's computers and laptops are another limitation, as they don't have particularly high performance, so the new system may need to be efficiently programmed in a suitable programming language, but otherwise this should not be a problem.

All printing in school is done by laser printers with high levels of detail and generous supplies of toner, but almost all printing is done in greyscale due to the special permissions required for colour printing at BRGS. Therefore, the new system should produce diagrams that remain clear in greyscale.

1.3 Data Volumes

Storage space is only required for the application itself and any diagrams that the user chooses to save. The application will likely be under a single megabyte and the size of each image file will depend on its resolution and file format, but should typically be less than a megabyte. This is very reasonable given that modern hard disk drives have capacities on the order of 100GB or 1TB and that the network at BRGS gives each pupil one hundred megabytes or more.

The application will also occupy space in main memory while running, most of which will likely be the colours of the pixels that make up the current image being displayed (likely under a megabyte) and the positions of vertices in three-dimensional space. Each vertex will be defined by 3 floating point numbers (each up to 8 bytes long) and there could be hundreds of vertices in 3-D space at any given time, meaning these could occupy up to 24000 bytes of main memory ($8 \times 3 \times 1000 = 24000$). This is a small amount by today's standards; modern computers have main memory on the order of gigabytes.

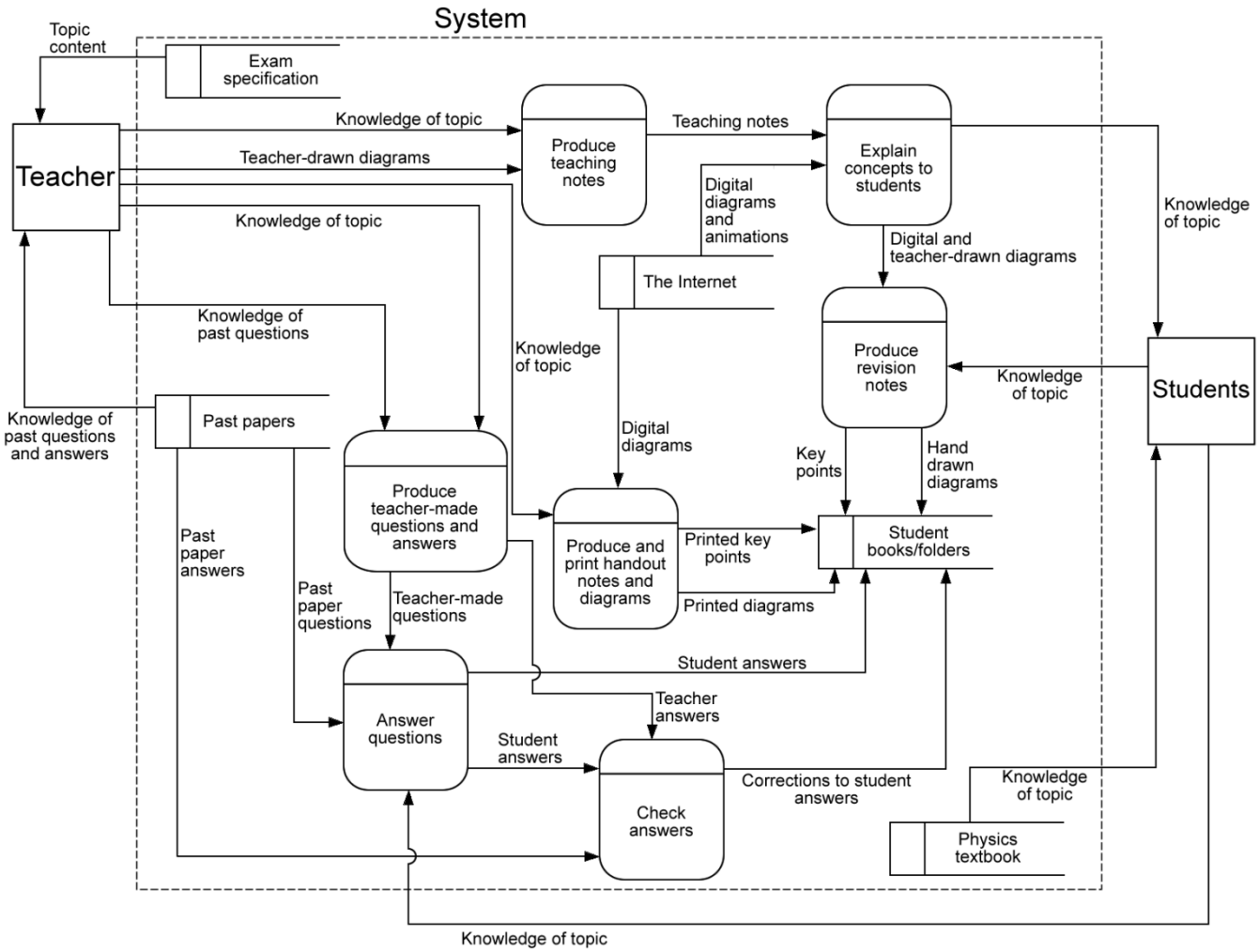
1.4 Data Dictionary

Data field	Uses	Data, resources and equations needed in conjunction for this use
Angle of incidence, i	Calculation of the angle of refraction, r	Relative refractive index, μ $r = \sin^{-1}\left(\frac{\sin(i)}{\mu}\right)$
	Determining whether total internal reflection occurs or not	Critical angle, c $total\ internal\ reflection\ occurs\ if\ i \geq c$
	Calculation of the relative refractive index, μ (such as from experimental results)	Angle of refraction, r $\mu = \frac{\sin(i)}{\sin(r)}$
	Equals the angle of reflection	
Angle of refraction, r	Calculating the angle of incidence, i	Relative refractive index, μ $i = \sin^{-1}(\mu \sin(r))$
	Calculation of the relative refractive index, μ	Angle of incidence, i $\mu = \frac{\sin(i)}{\sin(r)}$
Angle of reflection	Equals the angle of incidence, i	
Critical angle, C or c	Calculating the relative refractive index, μ	$\mu = \sin(C)$ $where\ \mu\ is\ the\ refractive\ index\ relative\ to\ the\ more\ dense\ material$ Or $\mu = \frac{1}{\sin(C)}$ $where\ \mu\ is\ the\ refractive\ index\ relative\ to\ the\ less\ dense\ material$
Material	Determines the absolute refractive index, μ	A lookup table of materials and their corresponding absolute refractive indices
Relative refractive index, μ (n at GCSE); a refractive index relative to a vacuum is also called an absolute refractive index (refractive index is the same as index of refraction). The refractive index of a vacuum is exactly 1 by definition of the term.	Calculating the angle of refraction, r	Angle of incidence, i $r = \sin^{-1}\left(\frac{\sin(i)}{\mu}\right)$
	Calculating the angle of incidence, i	Angle of refraction, r $i = \sin^{-1}(\mu \sin(r))$

Air is taken to have an absolute refractive index of 1 in exams unless stated otherwise in the question.	Calculating the critical angle, C	The light must be moving to a less dense material for there to be a critical angle $C = \sin^{-1}(\mu)$ $where\ \mu\ is\ the\ refractive\ index\ relative\ to\ the\ more\ dense\ material$ Or $C = \sin^{-1}\left(\frac{1}{\mu}\right)$ $where\ \mu\ is\ the\ refractive\ index\ relative\ to\ the\ less\ dense\ material$
	Calculating the light's speed in the material, v_1 or v_2	The speed of light in the material the refractive index is relative to, v_2 or v_1 $v_1 = \frac{v_2}{\mu}$ Or $v_2 = \mu v_1$
	Calculating the reverse relative refractive index – for example calculating the refractive index of air relative to glass using the refractive index of glass relative to air	$\mu_2 = \frac{1}{\mu_1}$ Hence, the refractive index in each equation can be substituted for the reciprocal of the reverse refractive index – this is usually only done when calculating the critical angle because the refractive index relative to the less dense material is typically given in the question or data sheet
	Identifying a material	A lookup table of materials and their corresponding refractive indices
Light frequency, f	Calculating the wavelength of light in a material, λ	Speed of the light through the material, v $\lambda = \frac{v}{f}$
	Calculating the speed of light through the material, v	Wavelength of the light in the material, λ $v = f\lambda$
Light wavelength in a material, λ	Calculating the frequency of light, f	Speed of the light through the material, v $f = \frac{v}{\lambda}$
	Calculating the speed of light through the material, v	Frequency of the light, f $v = f\lambda$
Light speed in a material, v or v_1. The speed of light in a vacuum is	Calculating the light's frequency, f	Wavelength of the light in the material, λ

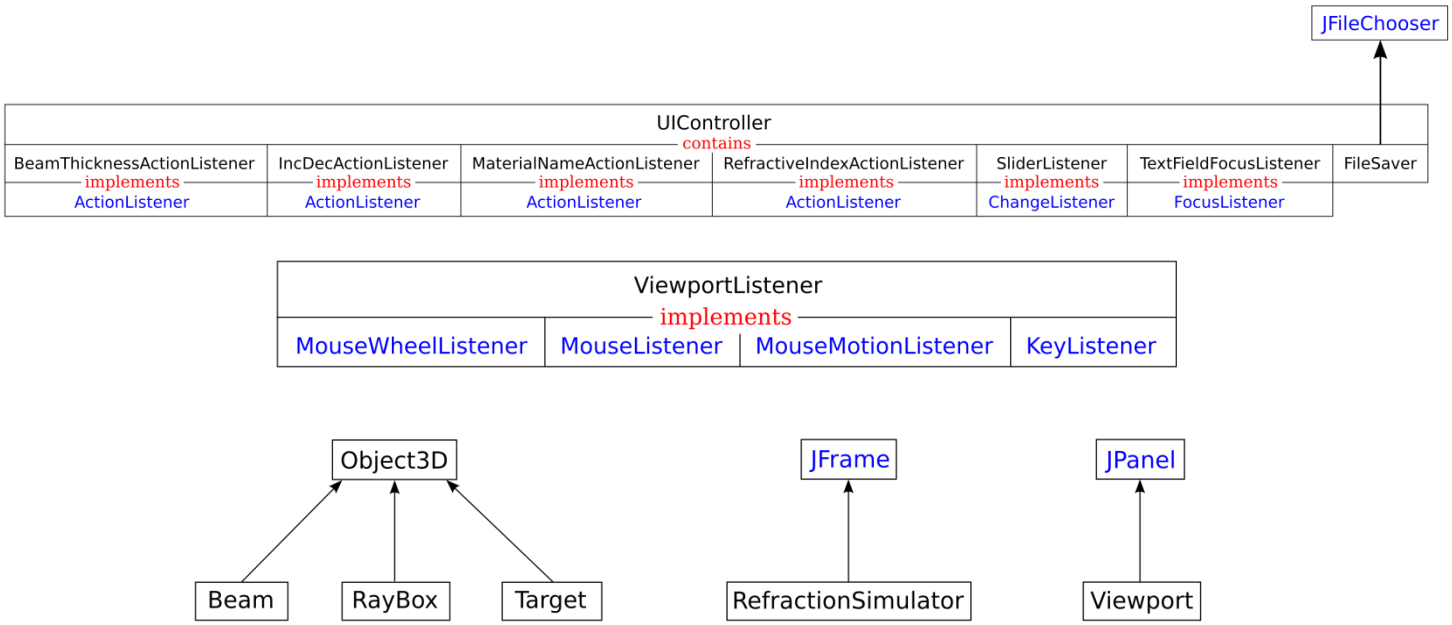
taken to be $3 \times 10^8 ms^{-1}$ and represented by c		$f = \frac{v}{\lambda}$
	Calculating the light's wavelength in the material, λ	Frequency of the light, f $v = f\lambda$
	Calculating the speed of light through another material, v_2	Refractive index of material 1 relative to material 2, μ $v_2 = \mu v_1$
	Calculating the refractive index of material 1 relative to material 2, μ	Speed of light through material 2, v_2 $\mu = \frac{v_1}{v_2}$

1.5 Data Flow Diagram



1.6 Object Analysis

1.6.1 Class Relationships



Key:	class I must develop	pre-defined class or interface	type of relationship

1.6.2 Class Descriptions

- **RefractionSimulator**: a window of the application which starts the rest of the application.
- **UIController**: contains methods for generating and regenerating parts of the user interface and returning them so that the window can display them.
    - **BeamThicknessActionListener**: validates the information entered into a beam thickness text field, rounds it to an integer between 1 and 10 inclusive or the last valid entry and updates the geometry of the beam object.
    - **IncDecActionListener**: enters an appropriate new beam thickness when the user clicks one of the buttons to increment or decrement the beam thickness.
    - **MaterialNameActionListener**: trims whitespace from the beginning and end of the string entered into a custom material name field, restricts the result to 30 characters (and trims again) and replaces a blank name with “Custom material”.
    - **RefractiveIndexActionListener**: validates the information entered into a new material refractive index field and restricts it to a real number between 1 and 100 inclusive.
    - **SliderListener**: when a slider is dragged, this updates other sliders and the position and orientation of the selected ray box.
    - **TextFieldFocusListener**: triggers a text field’s action listener when it loses focus (typically when the user clicks on something else) so that the input can be validated and altered appropriately.
    - **FileSaver**: allows dialog boxes to be created which let the user save an image to a secondary storage device.

- **Viewport**: a user interface component which displays images of 3-D space in real-time as well as handling all of the 3-D objects and updating other areas of the user interface when called to do so by the ViewportListener object.
- **ViewportListener**: responds to both mouse and keyboard events that involve its Viewport object and often call the Viewport object's methods in order to update 3-D space or the user interface.
- **Object3D**: allows for the creation of generic three-dimensional objects and provides methods for manipulating these objects
- **Target**: 3-D objects which are more specifically target objects, meaning they have a material and must be one of the primitive shapes.
- **RayBox**: 3-D objects which are more specifically ray boxes, meaning they are cubes with centre five units from the world's origin and each has its own Beam object.
- **Beam**: 3-D objects which represent beams of light and as such their geometry is dependent on their origin's position, their orientation, the shape of the target object and the materials of the world and target object.
- **Mesh**: represents the geometry (vertices and triangular faces) of a 3-D object as well as its arbitrarily orientated bounding box.
- **Matrix**: represent transformations in n-dimensional space such as projections, rotations and enlargements.
- **Vector**: represent positions or displacements in n-dimensional space for operations that involve matrices.
- **EulerTriple**: represent orientations and angular displacements in 3-D space using Euler angle triples (heading, pitch and bank). Methods are provided for adding angular displacements and converting the EulerTriple to a Matrix object.
- **Ray**: represent lines in three dimensions each with a starting point and a direction. These are used in calculating the path of a beam of light**.**
- **Edge2D**: represent edges (line segments) in two dimensions. These are used in rendering faces and detecting if a ray passes through a particular face.
- **Math2**: contains static methods (accessible without creating an instance of the class) which perform general-purpose mathematical operations not already provided by the Math class.

1.7 Objectives of the Project

1.7.1 General Objectives

By the end of this coursework I aim to complete an application which meets the following objectives:

1. The new system should be able to output graphical and numerical information in a manner suitable for live demonstration and still images for projection onto a screen or whiteboard as well as for greyscale printing.
2. The user should have reasonable control over the format and amount numerical information outputted.
3. Users should be able to modify parameters such as light frequency, materials, shapes and the number of light sources in order to produce a desired diagram, investigate the effects of individual parameters or to set and answer calculation questions in two and three dimensions.
4. The new system should be able to run smoothly on the school's computers and laptops in order to render a three-dimensional scene in real-time while the user orbits, pans, zooms and modifies parameters in real-time.

1.7.2 Specific Objectives

- Input objectives:
    1. Allow users to change their view in the three-dimensional space by turning perspective mode on and off (making objects further away appear smaller) and by panning, zooming and orbiting with the mouse and keyboard or selecting a preset view such as from directly above, below, in front of and behind. The user will be restricted from zooming out too much or panning too far away such that nothing is visible.
    2. Allow users to add and remove ray boxes (each with its own light beam). Appropriate validation will prevent the user adding so many ray boxes that the application uses all of the computer's resources and crashes the computer.
    3. Allow users to set the shape of the target object (which the incident beams of light travel into) from a list.
    4. Allow users to type a label for an object (for example to differentiate two light beams which look similar, particularly for printing).
    5. Allow users to choose the visual properties of light beams individually. These properties include whether wavefronts are to be displayed and in what way they should be represented (selection from a list), what thickness and colour the light beam should be and whether angles relevant to the beam are to be displayed. These options will be controlled within a single form which also includes the light beam's label.
    6. Allow the user to select the surrounding material (which the incident beams of light come from) and the target material (which the incident beams pass into) from a list or let the user define a 'custom' material with a refractive index they define; validation should ensure this is greater than or equal to 1.

7. Allow the user to set the angle of incidence or refraction for a particular light beam by dragging the corresponding ray box or clicking the current angles and typing a new angle between 0 and 90 degrees or 0 and $\frac{\pi}{2}$ radians for each.
8. Allow the user to set a light beam's frequency or wavelength in a material of specified refractive index (this index can be changed).

- Output objectives:
    1. The 3-D viewport should reflect the current settings within the application such as the viewing angle and whether wavefronts and angles are displayed.
    2. The positioning of objects such as ray boxes should reflect the current settings within the application.
    3. The paths of light beams should be the results of processing carried out on the current settings, parameters and the laws of physics that are relevant to this area of physics.
    4. Values in the 3-D viewport such as angles should be kept a readable size despite how far the user has zoomed out.
    5. The user interface should make it clear to the user where different options are and also what values the settings and parameters currently have, for example whether a perspective mode is on or off and what the frequency of a particular beam of light is.
    6. The user should be able to export the current viewport image to a common bitmapped image file format with a name of their choosing. File names will be validated so that they do not include special characters which the file system may not like.
    7. Suitable colours should be used in the viewport to keep diagrams clear when printed in greyscale.
- Processing objectives:
    1. Processing of the relevant object geometries and the current settings and parameters needs to be carried out in order to map the 3-D co-ordinates of the objects' vertices to screen space as well as determine appropriate colours for points on the objects (to give the illusion of light and shadow) and whether other faces obstruct them from direct line of sight of the user (these may be semi-transparent) so that the scene can be rendered correctly.
    2. Processing of user input needs to be carried out in order to ensure the input is valid and to determine what changes need to be made to the interface and parameters as a consequence.
    3. Processing of materials' refractive indices and the positions and orientations of ray boxes is necessary to determine the path of a beam of light and hence derive the geometry of the beam object acting as a visual representation of the light's path.
    4. When an image is exported, processing is needed to validate the file's name and determine the file format. Then, any files with the same name in that directory must be detected so that the user can decide whether the old file should be overwritten. If there is a problem in saving the file, the user should be notified and the cause of the problem stated if it is known.
- Storage objective:

1. The user should be able to choose where to save bitmapped images of the 3-D viewport so that they can be accessed by other applications and even printed and stored in students' books/folders with their revision notes. These image files will store colour data for each pixel in a large grid of pixels.

Client signature:

1.8 Feasibility of Potential Solutions

The simplest method for implementing the new system would be to limit the visual output to two dimensions and use off-the-shelf software meant for developing interactive two-dimensional projects, such as Adobe Flash (or Macromedia Flash, which is installed on the school's network) with ActionScript. The advantages of this are the ease of development and the fact that I am already familiar with Macromedia Flash and ActionScript. However, this doesn't meet the objective of being able to produce three-dimensional diagrams to aid explanation and allow bright students to challenge themselves with calculation questions in three dimensions. Furthermore, this method would not be a complex solution.

Another option would be to create a website or webpage which would draw diagrams using JavaScript and the HTML canvas element; parameters could be modified using a form or multiple HTML forms on the page which use JavaScript to pass the form input to the drawing scripts. I am very familiar with JavaScript, HTML and CSS, making this method less difficult to implement. Also, students can access the application outside of school and no action needs to be taken by the school or users to install the application or web browser plug-ins. This does however require me to maintain a web server and there would also be some difficulty making the application compatible across web browsers – users can access the web through Internet Explorer or Google Chrome in school, as well as any other browser from home. Moreover, JavaScript and HTML forms running within a browser are not as efficient as native software in other programming languages, so three-dimensional rendering may not be practical.

A native application could be developed using Java, which provides functionality such as garbage collection which makes development easier than other languages. I have some familiarity with Java and Java is supposed to be operating system independent, meaning I could also develop on my Linux laptop when a Windows desktop is not available. On the other hand, Java's extra functionality can remove some things from the programmer's control, making this solution less efficient and less complex than it could potentially be.

C++ is commonly used for three-dimensional rendering and is very efficient; this may be useful due to the hardware limitations of the school's computers and laptops. However, building a native application using C++ would be difficult due to the amount of responsibility given to the programmer and the fact I haven't learned any C++. Fortunately, this makes the solution more complex.

1.9 Chosen Solution

I have decided that developing my application using Java would be the best course of action because of this method's balance of performance and ease of development. Using Java also makes my application more likely to be useful to the school for longer than with C++ because the school may decide to change its computers' operating systems in the future (such as for security reasons), meaning that the C++ source code would need to be recompiled, possibly with different libraries and changes to the source code. By comparison, an executable jar file (in Java bytecode format) should be transferrable from the school's old system to the new one unless there have been changes to the Java virtual machine.