

Міністерство науки та спорту України
Головне управління освіти і науки Дніпропетровської облдержадміністрації
Дніпропетровське територіальне відділення МАН України

Відділення: фізика

Секція: теоретична фізика

**Оптичний конструктор: віртуальне дослідження проходження променів
світла крізь оптичні системи**

Роботу виконав:

Бабкін Владислав Володимирович,
Учень 11 класу Дніпропетровського ліцею
інформаційних технологій при
Дніпропетровському національному
Університеті імені О.Гончара

Науковий керівник:

Козлова Тетяна Іванівна,
вчитель вищої категорії,
вчитель-методист,
відмінник освіти України

Дніпропетровськ – 2012

Зміст

Вступ.....	3
1. Теоретична частина: «Геометрична оптика та її закони»	4
1.1 Промінь світла та незалежність променів один від одного	4
1.2 Закон прямолінійного поширення світла в однорідному середовищі.....	5
1.3 Закон відбиття світла від дзеркальних поверхонь, плоскі та сферичні дзеркала	5
1.4 Кулі та лінзи, закон заломлення світла	7
1.5 Повне внутрішнє відбиття.....	10
1.6 Побудова променів у тонкій лінзі	11
2. Практична частина	14
2.1 Цікава задача: Чи буде промінь відбиватися по замкнутій траєкторії?	14
2.2 Використання відображення променів у прямокутнику з дзеркал для розв'язання задач механіки	16
2.3 Огинальні криві в еліптичних та сферичних дзеркалах.....	19
2.4 Хід променів у кулі	21
2.5 Чи дійсно лінза та сферичне дзеркало фокусують промені у одну точку, або явище сферичної аберації.....	22
2.6 Опис «Оптичного конструктора»	24
2.6.1 Можливості конструктора.....	24
2.6.2 Принципи побудови програми та декілька слів про мову програмування	25
2.6.3 Можливості переведення інтерфейсу програми користувачем	27
2.6.4 Обмеження програми	27
2.6.5 Напрями використання програми.....	28
3 Висновок.....	28
4 Джерела інформації та використані програмні засоби	29
5 Додатки	30
5.1 Програмно-апаратні вимоги конструктору	30
5.2 Фрагменти коду	30
5.2.1 Кореневий об'єкт.....	30
5.2.2 Код перетину кола та прямої.....	34

Вступ

Перші оптичні прилади та пристрої з'явилися багато сторіч тому. Створювались вони вмілими майстрами, які мали не лише вмілі руки, а й велику спостережливість. Перші лінзи та телескопи були зроблені на фундаменті розрахунків та попереднього досвіду. Розрахунки велися за допомогою законів геометричної оптики: закону прямолінійного поширення світла в однорідному середовищі, законів відображення та заломлення світла, отриманих експериментально. З відкриттям хвильової природи світла ці закони не зменшили своє значення, але зараз їх можна вивести з різних принципів теорії хвильової оптики.

Саме виконання цих законів ми бачимо не тільки у дослідженнях ходу променів світла у оптичних системах, а й у астрономії. Ці закони можна зв'язати і з іншими галузями фізики, наприклад механіки, через схожості у математичних виразах певних правил та законів. Саме цьому робота присвячена дослідженню законів геометричної оптики та моделюванню ходу променя у оптичних системах – ця задача вирішує одночасно і пряму свою мету – знайти цю траєкторію, та має і багато інших використань, що особливо добре показано на прикладі «більярдів». Також такий проект можна використати у галузі астрономії для побудови телескопів, на уроках фізики як у школах, так і у вищих навчальних закладах, у дослідницькій праці.

У роботі досліджено різні оптичні ефекти та явища – каустики, огибаючі, сферична аберація, повне внутрішнє відбиття, хід променя у кулі та лінзі, «більярди», тощо. А створений «прилад» – «Оптичний конструктор», моделююча програма – може бути використаний і далі, бо він не був лімітований лише цими задачами. Цим робота і показує свою цінність. Не зважаючи на очевидне відношення до теми оптики, її можна використати і для певних досліджень у галузі механіки. Розроблена модель виходить за шкільний курс геометричної оптики вивченими у ній явищами та навіть самими правилами відбиття/заломлення у об'єктах – наприклад за шкільним правилом ні у лінзі, ні у дзеркалі не буде сферичної аберації, а у програмі це явище побачити легко.

Створена програма – багатомовна, що дозволяє ще поширити можливості використання. До того ж вимоги програми до комп'ютеру – дуже низькі.

Докладніше і про теорію, і про програму, і про рішення задач – у відповідних частинах роботи.

Теоретична частина: «Геометрична оптика та її закони»

Промінь світла та незалежність променів один від одного

У геометричній оптиці розглядається проходження світлових променів крізь середовище. Але щоб з'ясувати що ж таке світловий промінь, треба дізнатися що таке світловий пучок.

Світловий пучок – оптичне випромінювання, що поширюється у напрямі від (чи до) певної обмеженої ділянки простору, що зветься фокусом (центром) світлового пучка.

Світловий промінь – дуже вузький світловий пучок, що поширюється у напрямі від світлої точки, а *світла точка* – це ділянка простору, розмірами якої у даній задачі ми можемо знехтувати і прийняти цю ділянку за точку.

У геометричній оптиці будемо розглядати лише монохроматичний промінь – той, який складається лише з одного кольору зі спектру.

Зазначимо, що ми можемо розглядати промені окремо один від одного, тобто в геометричній оптиці ми будемо розглядати промені які не оказують дії один на інший – це і називається *незалежністю променів один від одного*.

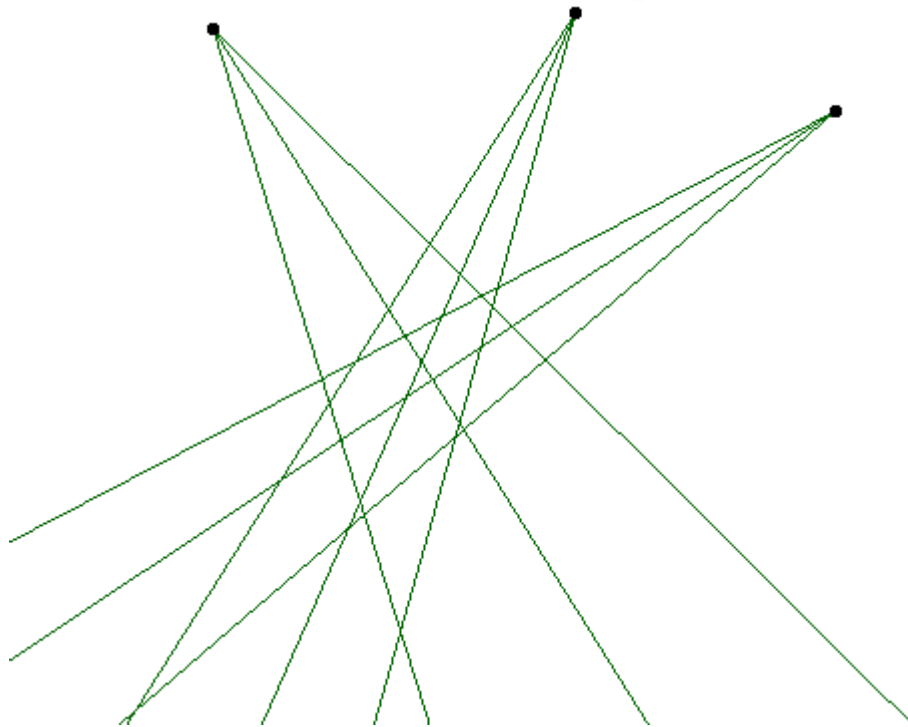


Рис. 1. Незалежне поширення променів світла – не зважаючи на те, що промені перетинаються, їх траєкторії не викривлюються, а залишаються прямими.

Закон прямолінійного поширення світла в однорідному середовищі

До речі, на попередньому рисунку промені світла поширюються як прямі лінії. Це підтверджується законом прямолінійного поширення світла в однорідному середовищі – промінь в однорідному середовищі переміщується по прямій лінії до перетину з перешкодою/границею поділу двох середовищ.

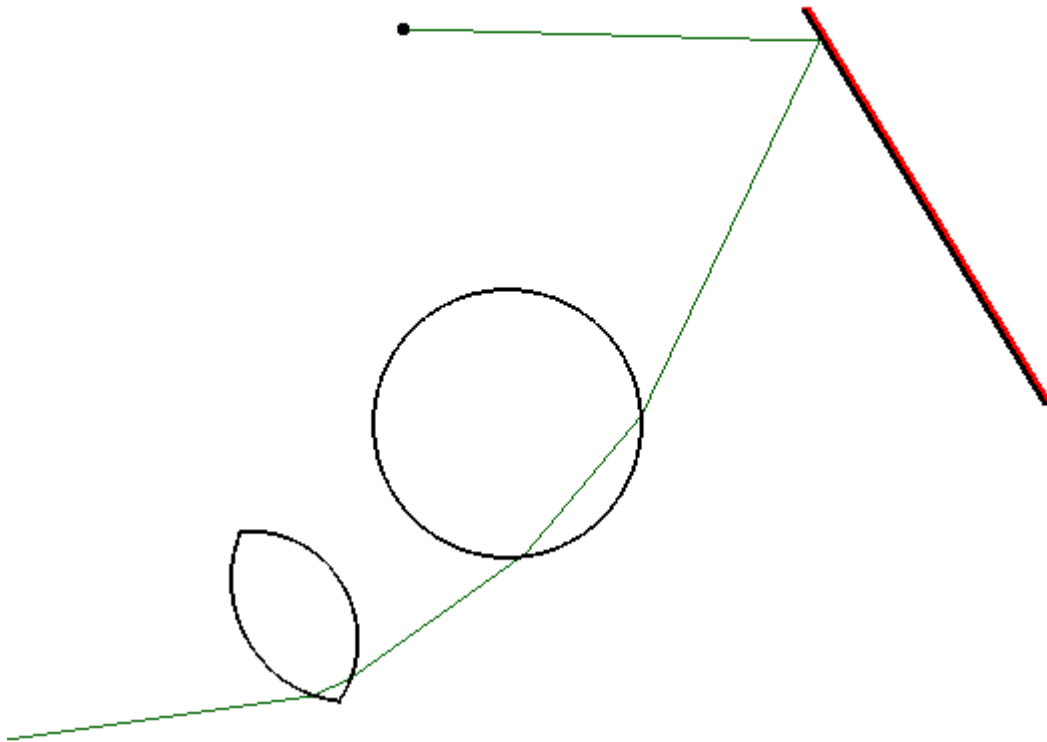


Рис. 2. Промінь світла до перетину з дзеркалом, кулею, та лінзою поширюється прямолінійно. Після перетину його новий напрям визначається законами геометричної оптики, після чого промінь продовжує прямолінійне поширення, але у новому напрямі

Закон відбиття світла від дзеркальних поверхонь, плоскі та сферичні дзеркала

Вище було зазначено, що при перетині з дзеркалом промінь змінить свій напрям. *Закон відбиття від дзеркальних поверхонь* формулюється так: промінь падаючий, промінь відбитий, та перпендикуляр до поверхні у точку падіння знаходяться в одній площині, та кут падіння дорівнює куту відбиття. *Дзеркальна поверхня* – це така поверхня нерівностями якої можна знехтувати (розміри цих нерівностей менше довжини хвилі світла). Якщо поверхня не

дзеркальна, то промінь від неї дзеркально відбиватися не буде. У геометричній оптиці такі поверхні розглядаються як поглинаючі. За цими правилами можемо побудувати продовження променя після відбиття від плоского дзеркала.

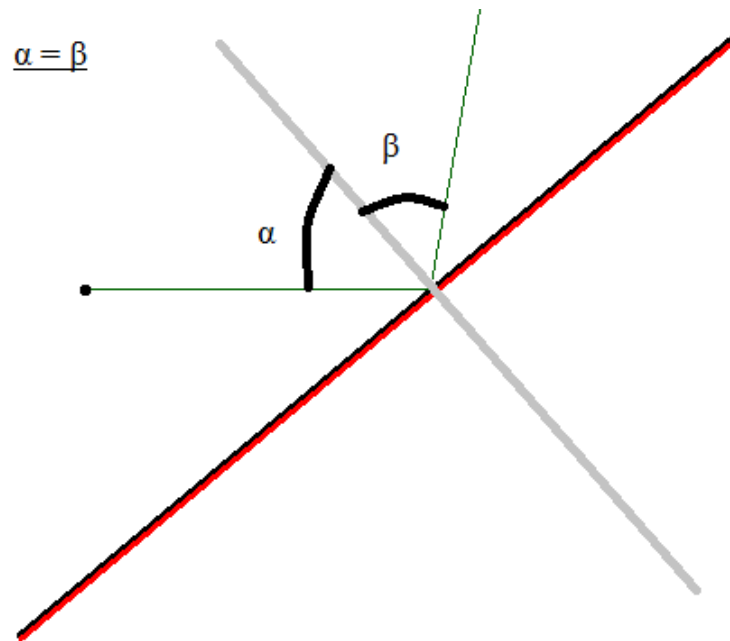


Рис. 3. Закон відбиття світла від плоского дзеркала. Кут падіння α дорівнює куту відбиття β . Сірим позначено перпендикуляр до дзеркала. Зелений – промінь. Початок – у джерелі.

Плоске дзеркало – прилад, що відбиває промені. З одного його боку – відбиваюча поверхня, а з іншого – поглинаюча. Дзеркальна сторона позначається чорним кольором, а поглинаюча – червоним.

Щодо сферичного дзеркала, цей закон не змінюється. *Сферичне дзеркало* – прилад, що являє собою частину кулі, яка з одного боку відбиває світло, а з іншого – поглинає. Приклади сферичних дзеркал:

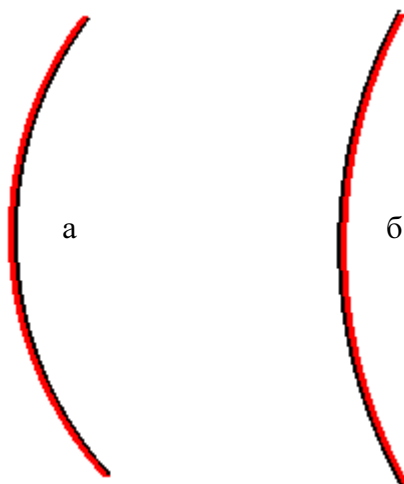


Рис. 4. а) Увігнуте дзеркало б) Опукле дзеркало

Для побудови віддзеркаленого променя від такого дзеркала, необхідно побудувати дотичну до точки падіння променя, і розглядати цю дотичну як звичайне плоске дзеркало. Для побудови дотичної проведемо радіус з центру дзеркала (центр дзеркала – центр кола, частиною якого є дзеркало), і далі проведемо перпендикуляр до цього радіусу у точку падіння променя. Цей перпендикуляр за правилами геометрії – і є дотична

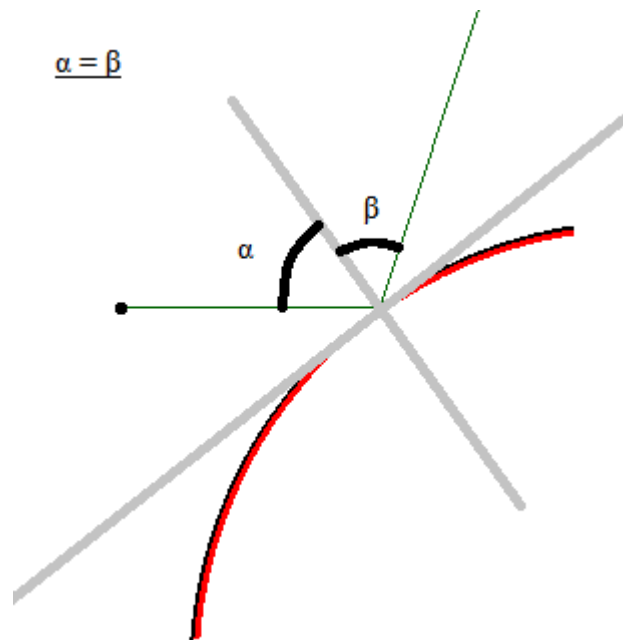


Рис. 5. Закон відбиття світла від сферичного дзеркала. Кут падіння α дорівнює куту відбиття β . Сірим позначено перпендикуляр до дотичної та саму дотичну. Зелений – промінь. Початок – у джерелі.

Кулі та лінзи, закон заломлення світла

Куля – оптичний об’єкт, що має форму однойменної геометричної фігури та певну оптичну густину.

Оптична густина середовища (абсолютний показник заломлення середовища) – відношення швидкості світла у вакуумі до швидкості світла у даному середовищі.

Однорідне середовище – середовище, у границях якого оптична густина незмінна.

Лінза – оптичний об’єкт, виготовлений із прозорого матеріалу, обмежений двома заломлюючими поверхнями, які мають спільну вісь, або

взаємно перпендикулярні площини симетрії. У роботі будемо розглядати лише заломлюючі поверхні, що мають форму кулі. Усі лінзи – товсті, тобто мають так відстань між поверхнями по головній оптичній вісі, якою неможна нехтувати. Тобто такі лінзи – не тонкі, і правила тонких лінз для них не працюють, що буде додатково показано у практичній частині.

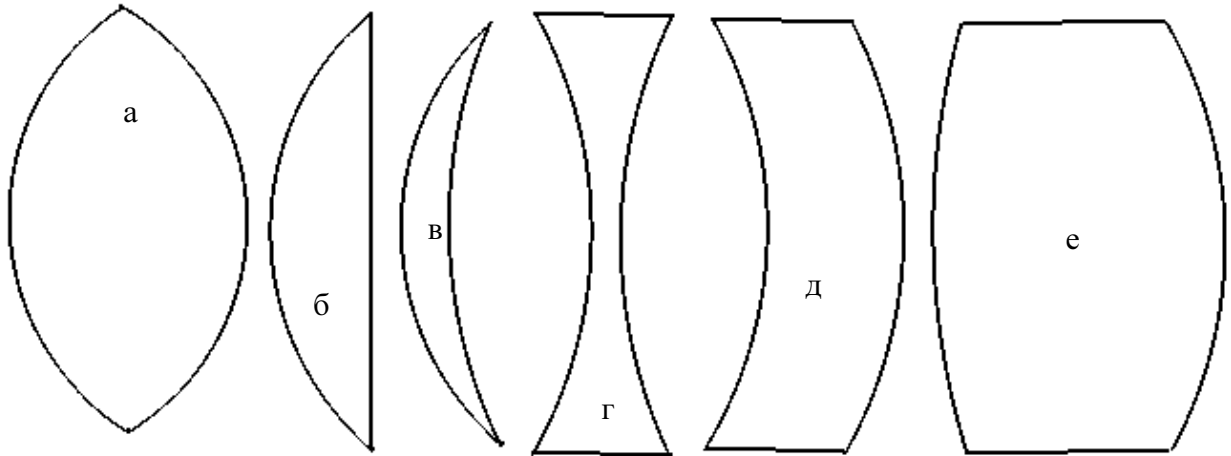


Рис. 6. Приклади лінз: а) двовипукла б) плоско-випукла в) випукло-ввігнута г) двоввігнута д) ввігнуто-випукла (інший вид) е) двовипукла (інший вид)

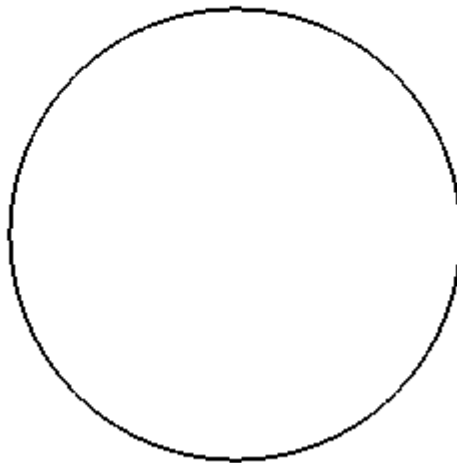


Рис. 7. Куля

Закон заломлення – промінь падаючий, промінь заломлений, та перпендикуляр у точку падіння знаходяться у одній площині. Кут заломлення для плоскої границі поділу двох середовищ: $\frac{\sin \alpha}{\sin \gamma} = \frac{n_2}{n_1}$, де n_2 – оптична густина

другого середовища, n_1 – оптична густина першого середовища, α – кут падіння, γ – кут заломлення. Остаточно кут заломлення визначається за такою формулою: $\gamma = \arcsin \frac{n_1 \sin \alpha}{n_2}$ (якщо $\frac{n_1 \sin \alpha}{n_2} \leq 1$).

Щодо використання цього закону для лінзи та кулі – потрібно провести дотичну до відповідного об'єкту, а потім заломлення піде вже через цю дотичну. Побудову дотичної описано вище. Лінза являє собою дві частини кулі та дві плоскі грані, тому для неї усе аналогічно до кулі – будуємо дотичну до частини кулі, на яку промінь впав, якщо він впав на частину кулі а не на, плоску грань, а далі заломлення крізь дотичну чи крізь плоску грань.

$$\frac{\sin \alpha}{\sin \gamma} = \frac{n_2}{n_1}$$

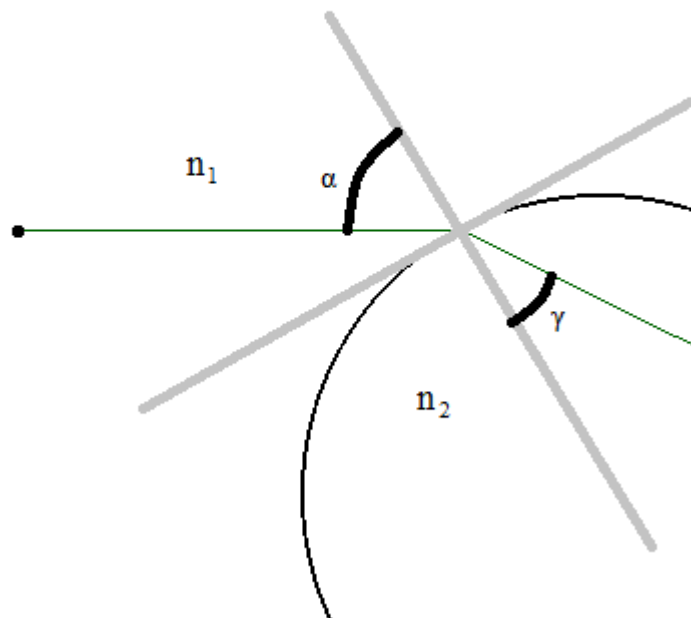


Рис. 8. Заломлення крізь сферичну поверхню

Із законів відбиття та заломлення, якщо їх взяти разом, виходить наступне: промінь падаючий, промінь заломлений, промінь відбитий та перпендикуляр до точки падіння завжди знаходяться в одній площині. Саме цей факт дозволяє нам розглядати двовимірну геометричну оптику, бо якщо б ці промені були б у різних площинах, то нам довелося б брати тривимірний випадок цієї науки, а він набагато складніший ніж плоский.

Повне внутрішнє відбиття

Повне внутрішнє відбиття – явище відбиття електромагнітних чи звукових хвиль від границі поділу двох середовищ за умови, що хвиля падає із середовища, де її швидкість поширення менша. У випадку оптики таке середовище буде мати більший показник заломлення. Тобто таке явище в оптиці може відбутися, якщо промінь йде із середовища з більшою оптичною густиною у середовище з меншою оптичною густиною. Визначимо, коли ж таке станеться. Позначимо середовище, з якого йде промінь за n_1 , а середовище, у яке він має потрапити за умови заломлення за n_2 . Запишемо закон заломлення світла: $\frac{\sin \alpha}{\sin \gamma} = \frac{n_2}{n_1}$. Тепер виразимо: $\sin \gamma = \frac{n_1 \sin \alpha}{n_2}$. Якщо $n_1 > n_2$, то вираз $\frac{n_1 \sin \alpha}{n_2}$ може стати більшим за 1. Якщо так буде, то заломлення не відбудеться, і промінь відіб'ється замість того, щоб заломитися. Саме це явище у геометричній оптиці і зветься повним внутрішнім відбиттям. Ось два приклади (у обох випадках середовище лінзи позначиться як n_1 , а зовнішнє середовище як n_2):

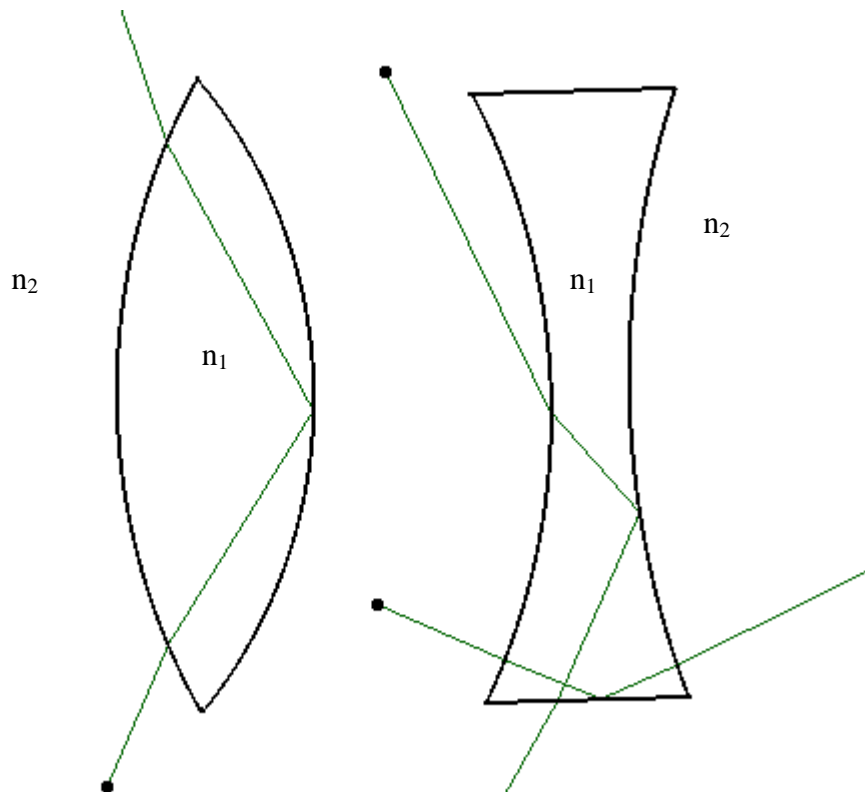


Рис. 8. Повне внутрішнє відбиття.

Побудова променів у тонкій лінзі

Тонка лінза – лінза з товщиною (відстанню між зовнішніми точками сфер) d значно меншою за радіуси кривизни $d \ll R_1$ і R_2 . Якщо умова не виконується – лінза товста. Так як товщина тонкої лінзи дуже мала, то її умовно вважають площиною (чи прямою у двовимірному випадку). Центр лінзи називають *оптичним центром*, а пряму, що з'єднує оптичний центр та обидва фокуси – *головною оптичною віссю*.

Фокальна площина – площина, що перпендикулярна до головної оптичної вісі та проходить крізь фокус.

Правила побудови ходу променя у збиральній лінзі такі – будуємо промінь до перетину з нею, після цього проводимо побічну оптичну вісь (пряма, що проходить крізь центр лінзи та паралельна променю) та будуємо фокальну площину, наступним кроком проводимо продовження променя крізь перетин фокальної площини та побічної вісі, та перетин променя з лінзою.

Щодо розсіювальної лінзи, то правило модифікується так – ми проводимо фокальну площину з боку самого променя, а не з протилежного як у збиральній. Потім, з'єднуючі перетини, ми отримуємо «уявну» частину продовження променя, тому проводимо його за лінзу. Більш детально – на рисунках.

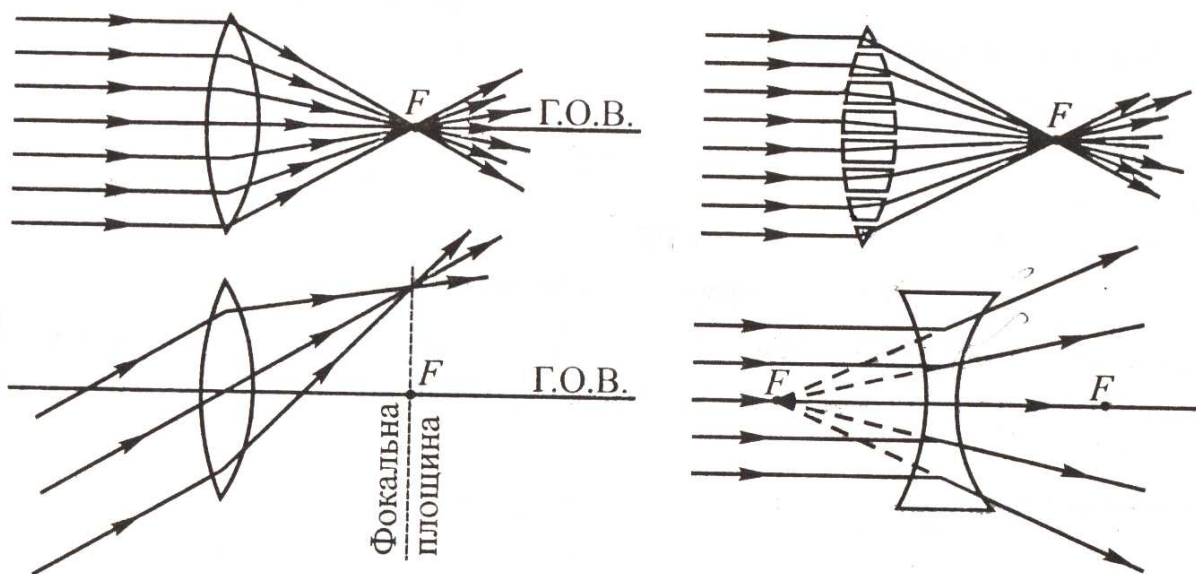


Рис. 9. Побудова ходу променя у тонких лінзах.

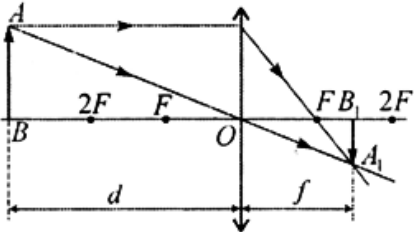
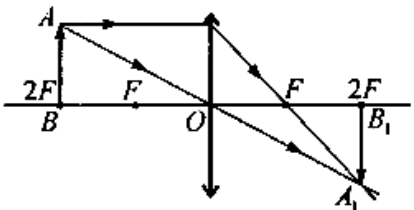
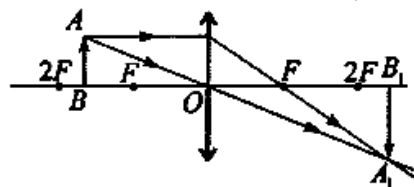
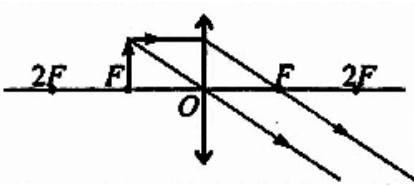
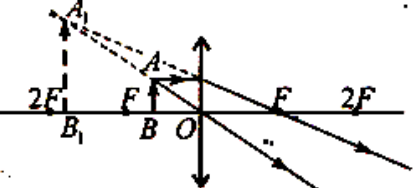
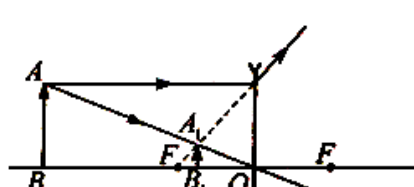
Зображення у лінзах					
Розміщення предмета	Зображення предмета				Застосування
	Характеристика	Вид	Величина	Розміщення	Око, фотоапарат
	Дійсне	Обернене	Зменшене	$F < f < 2F$	
	Дійсне	Обернене	Рівне величині предмета	$f = 2F$	—
	Дійсне	Обернене	Збільшене	$f > 2F$	Проекційний апарат
	—	—	Нескінченне	$f \rightarrow \infty$	—
	Уявне	Пряме	Збільшене	$f > F$	Лупа, окуляри
	Уявне	Пряме	Зменшене	$f < F$	Дверне вічко, оптичні системи

Рис. 10. Побудова зображень у лінзах

Як можна побачити з рисунку – для побудови зображення точки треба, щоб був перетин двох променів (або їх продовжень) що йдуть з неї, а для отримання зображення прямої, паралельної лінзі, необхідні зображення двох її кінців.

Тонка лінза – ідеалізована товста. Наприклад, у тонкій лінзі не відбувається сферична аберация світла (фокус є не прямою лінією, а точкою, про це детальніше у п. 2), не буває повного внутрішнього відбиття та інше. Через це жодна лінза не може дати ККД у 100% – будуть певні втрати енергії, наприклад, через вищезазнані явища. Щодо роботи – у ній найбільш детально розглянуто реальні, неідеалізовані лінзи, а тонкі лінзи наведено для порівняння.

Для порівняння – хід променів у тонкій лінзі (зліва), та у реальній лінзі (з конструктору, справа):

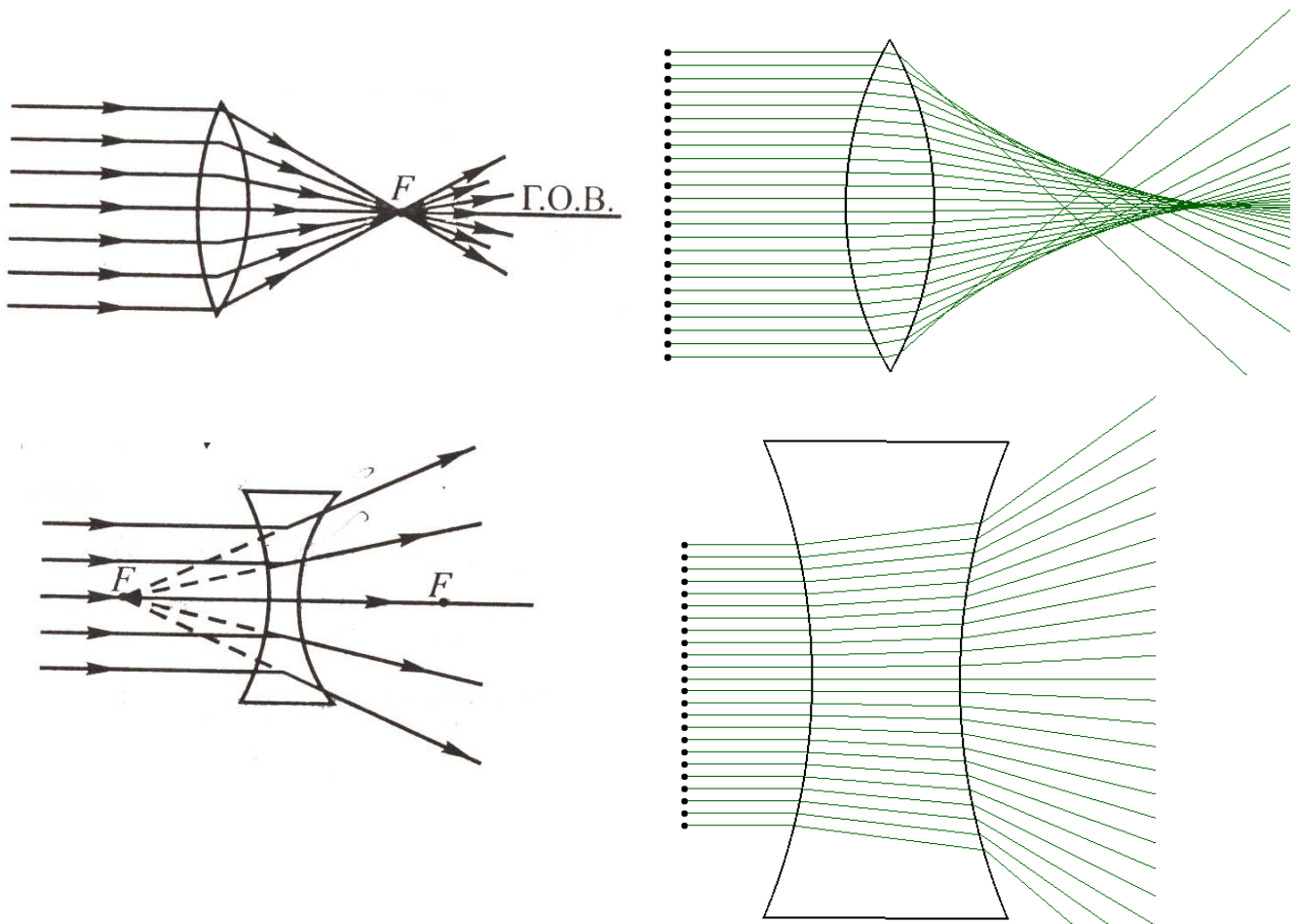


Рис. 11. Порівняння ходу променів у реальних та тонких лінзах.

Практична частина

Цікава задача: чи буде промінь відбиватися по замкнутій траєкторії?

Одна з цікавих задач – чи буде промінь віддзеркалюватися по якийсь замкнутій траєкторії. Було знайдено відповіді лише на деякі окремі випадки, але за допомогою конструктора можна моделювати хід променя. Траєкторія

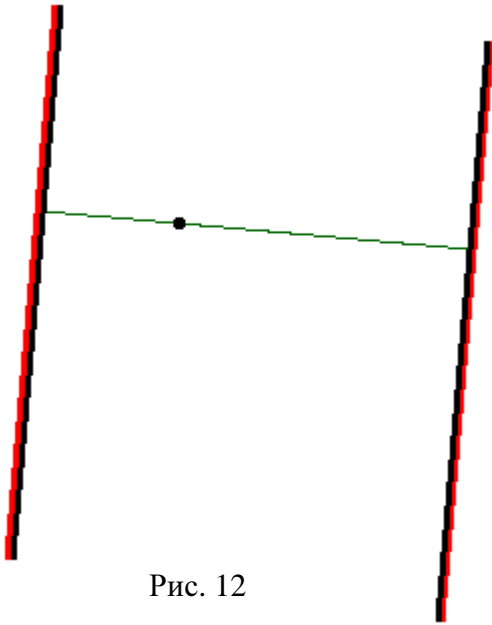


Рис. 12

буде замкнутою якщо є два паралельні плоскі дзеркала, і промінь у початку йде по перпендикуляру до них. На малюнку зображено два дзеркала, паралельні одне до одного, та промінь, що йде перпендикулярно до них. Це так через рівність кутів відбиття від дзеркала та падіння до нього. Через це промінь, що йде перпендикулярно до дзеркала кожен раз буде змінювати напрям на протилежний, і так до нескінченності.

Інший випадок – коли дзеркала формують квадрат, та промінь йде під кутом 45° до одного з них. Тоді промінь формує квадрат чи прямокутник:

У цьому випадку промінь знову відбивається під кутом 45° відносно наступного дзеркала. Далі на наступне дзеркало він впаде знову під кутом у 45° . Тому з точок перетину променя та дзеркала вийде рівнобедрений трикутник (за властивістю трикутника).

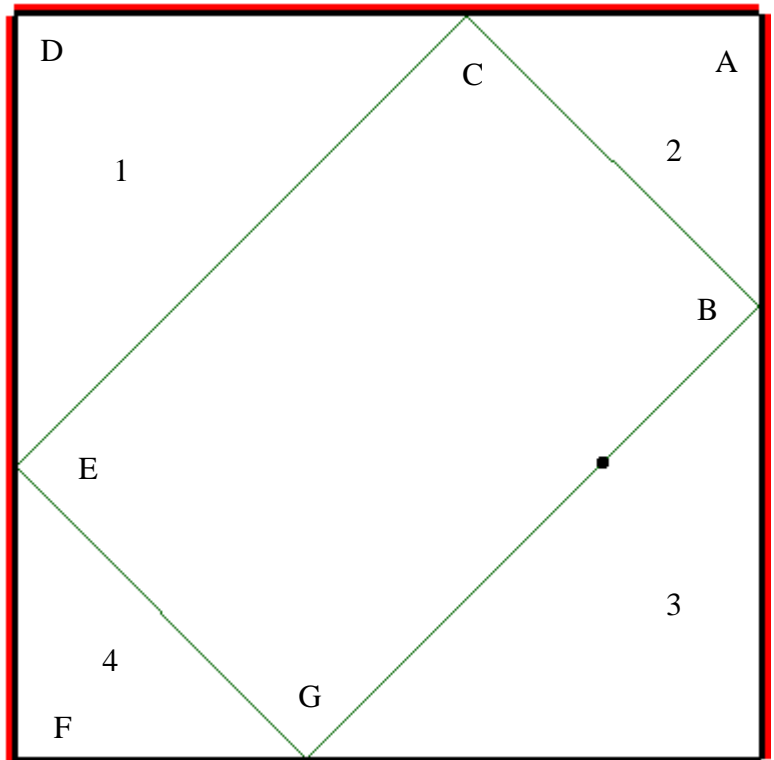


Рис. 13

І так 4 рази. Тому 4 утворені трикутники – рівнобедрені. (вони помічені цифрами на малюнку). Позначимо трикутники так: 4 – EFG ($EF = FG$); 1 – EDC ($ED = DC$); 2 – CAB ($CA = AB$); Тепер:

$FD = DA$ як сторони квадрату.

$FD = FE + ED$;

$DA = DC + CA$; Тому:

$FE + ED = DC + CA$;

$ED = DC$ як сторони рівнобедреного трикутника. Тому виходе, що

$FE = CA = FG = AB$;

Тому трикутники EFG та CAB рівні за двома сторонами та кутом між ними (кут $A =$ куту $F = 90^\circ$). Тому $EG = CB$; Аналогічно $GB = EC$; Саме тому EGBC – прямокутник. Якщо вважати, що траєкторія буде незамкненою, то отримуємо що одна зі сторін квадрату більша за іншу, що неможливо.

Траєкторія у загальному випадку замкнеться, якщо промінь «впаде» на будь-яку точку на оптичному об'єкті хоча б двічі під одним і тим самим кутом. Тоді промінь почне йти по тому ж самому шляху, по якому він шов після першого відбиття/заломлення, і знову повернеться до тієї самої точки 3-й, 4-й і так далі рази. Якщо ж промінь жодного разу впаде на одну і ту саму точку під одним кутом двічі – промінь до нескінченості буде відбиватися від об'єктів системи або вийде за межі системи. Цим обумовлені деякі обмеження програми, описані нижче. Ось два оригінальних загальних випадки:

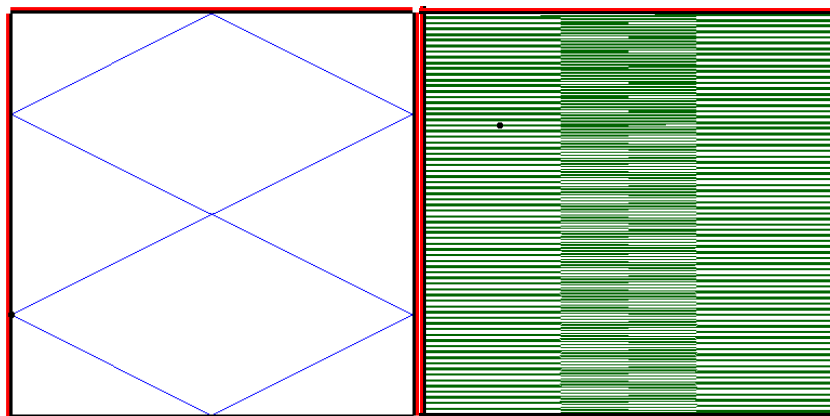


Рис. 14

Іншим випадком, для якого знайдено відповідь, є прямокутник. Про нього нижче.

Використання відображення променів у прямокутнику з дзеркал для розв'язання задач механіки

У попередньому пункті було дано відповідь на запитання про замкненість (або періодичність) траєкторії в квадраті. У цьому пункті буде розкрито узагальнений до прямокутника та променя, що йде під будь-яким кутом, випадок.

Нехай у нас є більярдне поле прямокутної форми без луз, по якому може без тертя та затрат енергії на відбиття кататися кулька з виконанням наступного правила: кут падіння дорівнює куту відбиття. Тоді перше значення ходу променя між дзеркалами, що йдуть у прямокутнику таке: Вони описують траєкторію кульки, що буде рухатися по такому столу. Цією траєкторією можна описати також рух малого тіла по тору, але про це дещо пізніше, спочатку розберемо рух по площині.

Спочатку «випрямимо» траєкторію руху тіла. Для цього продублюємо прямокутник для повного заповнення площини. Отримаємо сітку. Візьмемо ломану, що описує траєкторію руху тіла на полі. Для її «випрямлення» ми можемо використати такий алгоритм. Позначимо ломану за $P_1P_2P_3\dots$. Візьмемо точку P_2 та побудуємо продовження прямої P_1P_2 за площину дзеркала до перетину з наступною частиною сітки. Згідно закону побудови отриманий відрізок P_2P_3' симетричний до відрізка P_2P_3 . Тобто частину $P_1P_2P_3$ випрямлено. Далі використаємо цей алгоритм для випрямлення кожної ділянки траєкторії. Отримаємо пряму, що йде по площині. Якщо її «звернути» за оберненим алгоритмом – отримаємо траєкторію руху тіла по столу. Важливо зазначити, що усі описані траєкторії не проходять через кути прямокутника, бо нема чітко визначеного напрямку руху після відбиття від нього. На малюнку показано випрямлення траєкторії для перших чотирьох точок ломаної.

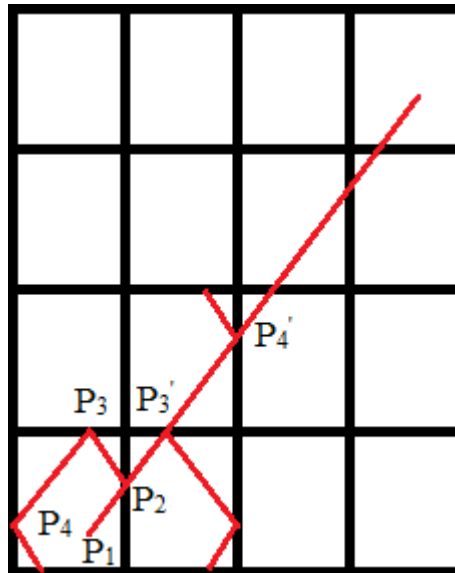


Рис. 15. Випрямлення траєкторії. Початкове поле – нижній лівий прямокутник.

Тепер, коли у нас є випрямлена траєкторія, визначимо, коли ж кулька буде рухатись по замкненій траєкторії. Якщо траєкторія періодична, тобто замкнена, то кулька пройде точку M (початок траєкторії) та через точку M з індексами m_0 та n_0 (вони показують, скільки відбиттів потрібно від горизонтальних та вертикальних сторін прямокутника для переходу у цю точку) таку, що її індекси – парні числа, а позиція відповідає позиції точки M при звертанні траєкторії. Це так, бо для збереження напрямку після відображень від сторін потрібно зробити парне їх число. На малюнку зображені точки M та M_{m_0, n_0} разом з індексами для кожної ділянки випрямленої траєкторії.

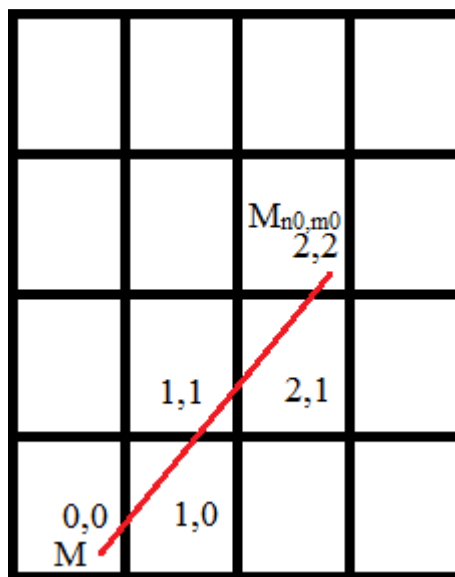


Рис. 16. Ознака замкненості траєкторії.

З цього випливає, що якщо відношення сторін прямокутника зіставимо з тангенсом кута нахилу початкового напрямку траєкторії – вона періодична. Аналогічне твердження – якщо тангенс кута нахилу діагоналі прямокутника до сторони відноситься до тангенсу нахилу траєкторії до сторони як раціональне число – траєкторія періодична. І навпаки, якщо отримане число – ірраціональне, то траєкторія неперіодична, а з цього випливає, що вона буде повністю заповнювати усе поле для більярду, тобто не буде такої точки на полі, у якій не побуває кулька. Усе сказане про траєкторію кульки тут – вірно і для променя, що йде між дзеркалами. Тобто промінь або йде по періодичній траєкторії, або повністю заповнює собою увесь простір прямокутнику.

Перейдемо до тору. У ньому, як і у кульці, наявні меридіани та паралелі. Якщо побудувати розвертку тора на площину, то ми отримаємо прямокутник, на якому буде сітка, що відповідає меридіанам, та паралелям, якщо склеїти тор з цього прямокутника.

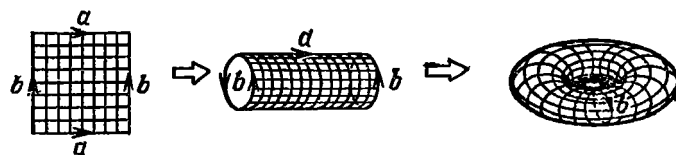


Рис. 17. Пояснення до розвертки тору.

Нехай кулька рухається по тору з певною кутовою швидкістю відносно меридіану та другою кутовою швидкістю відносно паралелі. Тоді вона буде створювати «обмотку». Якщо її представити у розвороті, то вона утворює прямі лінії, паралельні одна одній (бо площина замкнена відносно країв). Якщо ці прямі об'єднати через продовження за площину – отримаємо нескінчену пряму, яку можна звернути до ломаної лінії, що описує рух кульки у більярді.

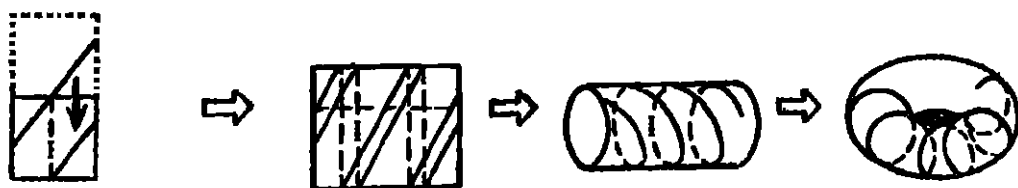


Рис. 18. Пояснення розвороту обмотки.

При чому тангенс куту нахилу такої прямої визначають кутові швидкості. Якщо цей тангенс відноситься до тангенсу кута нахилу діагоналі частини сітки як раціональне число, то траєкторія руху кульки по такому полю замкнена, а тому замкнена і траєкторія руху кульки по обмотці тору, і навпаки. При чому якщо це число ірраціональне, то на торі не буде точки, де не побуває кулька.

Підводячи підсумок, вивчення більярдів у оптиці та механіці допоможе у вивченні руху тіл у таких фігурах, як тор, що є важливим як для фізики, так і для математики. Більярди можуть бути не тільки прямокутними, а також багатокутними та криволінійними. Досліди показують, що у еліпсі траєкторія руху виглядає досить хаотично. Можливо на основі аналогічних перетворень з еліпсом вдасться знайти закономірності у такому руху, або навпаки – використати цей хаотичний рух для написання таких речей, як генератор випадкових чисел. Це означає, що дослідження більярдів може дати неочікувані результати та принести велику користь.

Огинальні криві в еліптичних та сферичних дзеркалах

Під час віртуальних дослідів ходу променя у сферичному дзеркалі спостерігалось явище огинання відбитими променями певної ділянки. Для сферичного дзеркала ця ділянка була круглої форми та розміщувалася посередині дзеркала. Тоді я вирішив знайти більше матеріалу з цієї теми. Ось декілька «фотографій» моїх дослідів.

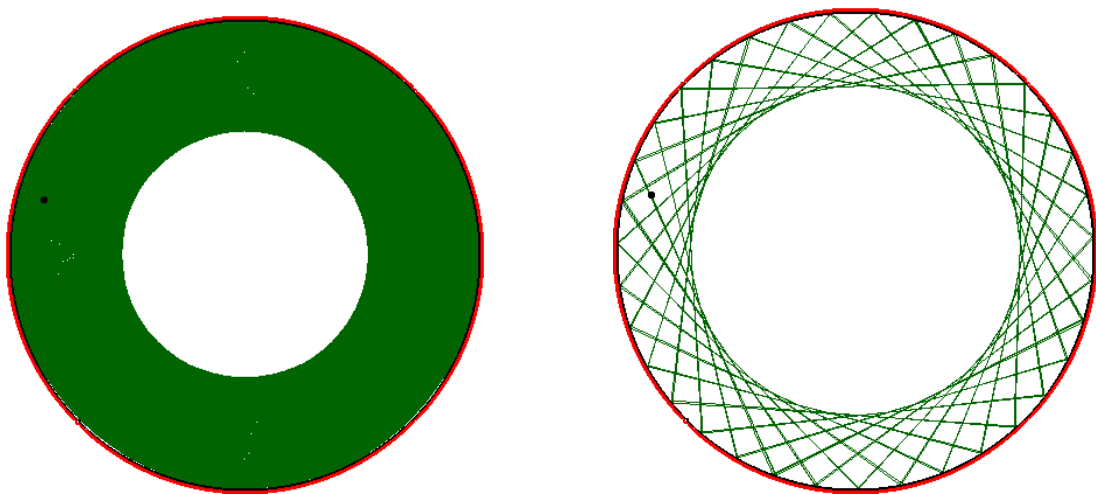


Рис. 19. Дослідження огинальних у сферичному дзеркалі для великої кількості відбиттів (5000 відбитих променів, зліва), та для малої кількості (75 відбитих променів, справа).

За результатами область має форму колу та кожен відбитий промінь є дотичною до цього колу. Спираючись на вже описану властивість більярдних траєкторій з опису прямокутного випадку, можемо сказати, що кулька, яка рухається по сферичному більярдному столу, буде котитися по саме цій траєкторії, і, на відміну прямокутного випадку, кулька не буде проходити крізь будь-яку точку столу, бо у центрі є таке коло, яке не перетнула траєкторія.

Також за результатами дослідів аналогічне явище можна побачити і для частин кіл. Ось більярд для чверті кола, півкола, та певної іншої частини кола:

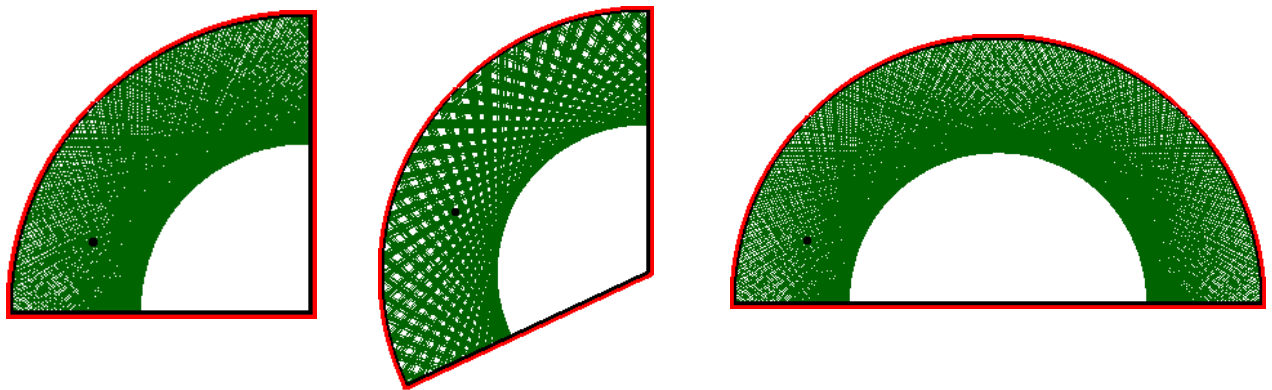


Рис. 20. Більярд у чверті кола (зліва), півколі (справа), частині кола (посередині).

Це явище досить відоме, і спостерігається воно і у еліптичних більярдах, але тут огинаюча крива отримує вигляд не кола, а еліпсу:

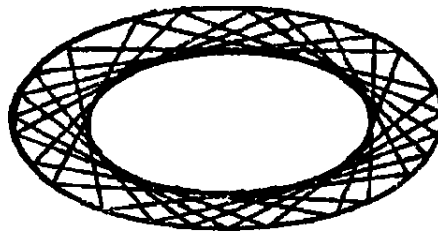


Рис. 21. Більярд у еліпсі.

Тобто вивчення ходу оптичного променя у сферичних поверхнях дозволяє знаходити траєкторію руху тіла, що пружно від них відбивається. Також визначати, чи буде траєкторія «заповнювати» простір столу (чи будуть точки, у яких не побуває тіло), чи буде вона періодична або чи буде вона являти собою многокутник без само перетинів. А ці задачі досить важливі, та можливості їх використання показані у роботі.

Хід променів у кулі

Досить неочікуваним є результат дослідження ходу променів у кулі. У ній ніколи не відбувається повне внутрішнє відбиття. Спочатку такий результат дослідів мене здивував, але з часом я зміг його довести.

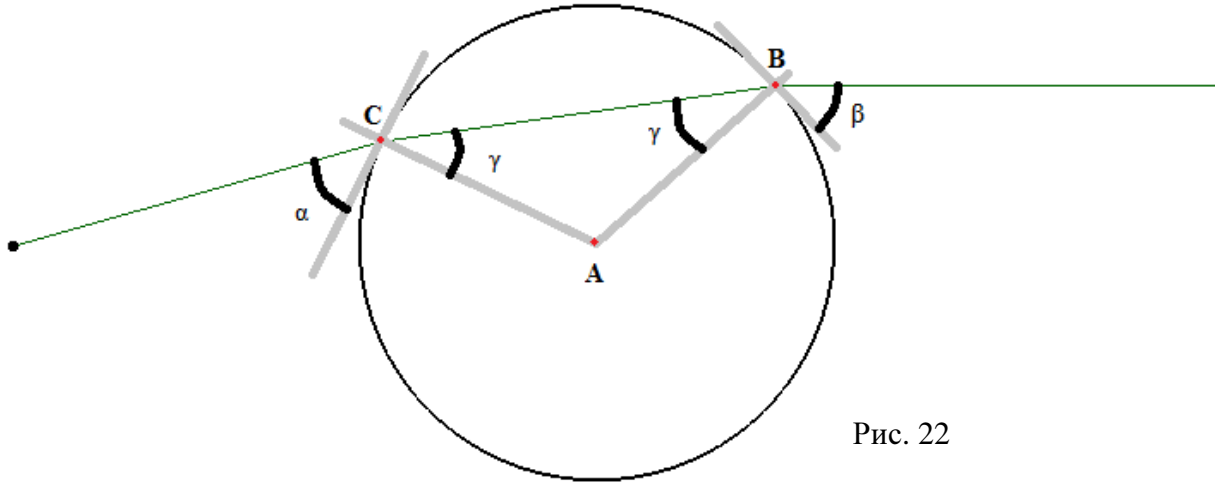


Рис. 22

На рисунку зображено хід променя крізь кулю з центром А. Нехай цей промінь перетнув кулю у точках В та С; Нехай кут падіння променя на кулю з зовнішнього середовища дорівнює α , оптична густина середовища – n_1 , оптична густина кулі – n_2 , крім того $n_2 > n_1$. Тоді, згідно формулі, кут заломлення $\gamma = \arcsin \frac{n_1 \sin \alpha}{n_2}$. Так, як трикутник ABC – рівнобедрений ($AC = BC = R_{\text{кола}}$), то кут ABC дорівнює куту ACB = γ . Другий раз кут заломлення стає кутом падіння, друге середовище стає першим, а перше середовище – другим. Нехай промінь другий раз заломився під кутом β .

Маємо два наступні вирази:

$$\sin \gamma = \frac{n_1 \sin \alpha}{n_2};$$

$$\frac{\sin \gamma}{\sin \beta} = \frac{n_1}{n_2} \Rightarrow \sin \gamma = \frac{n_1 \sin \beta}{n_2};$$

З них виходить, що $\sin \beta = \sin \alpha$, а так як обидва кути знаходяться на проміжку від 0 до 90°, то $\beta = \alpha$, тому $\sin \beta$ завжди менший або рівний 1, що унеможливорює повне внутрішнє відбиття у даному випадку.

Чи дійсно лінза та сферичне дзеркало фокусують промені у одну точку, або явище сферичної аберації

Для мене неочікуваним було те, що ні лінза, ні сферичне дзеркало не фокусували паралельно падаючі на них промені в одну точку. Спочатку я подумав, що це помилка у моєму конструкторі, але все ж таки вирішив дослідити це явище. У цей момент я і дізнався про сферичну аберацію світла та зрозумів, що ніякої помилки немає.

Сферична аберація — дефект зображення, при якому промені, що проходять поблизу оптичної осі системи і промені, що проходять на віддалі від оптичної осі і не збираються в одну точку.

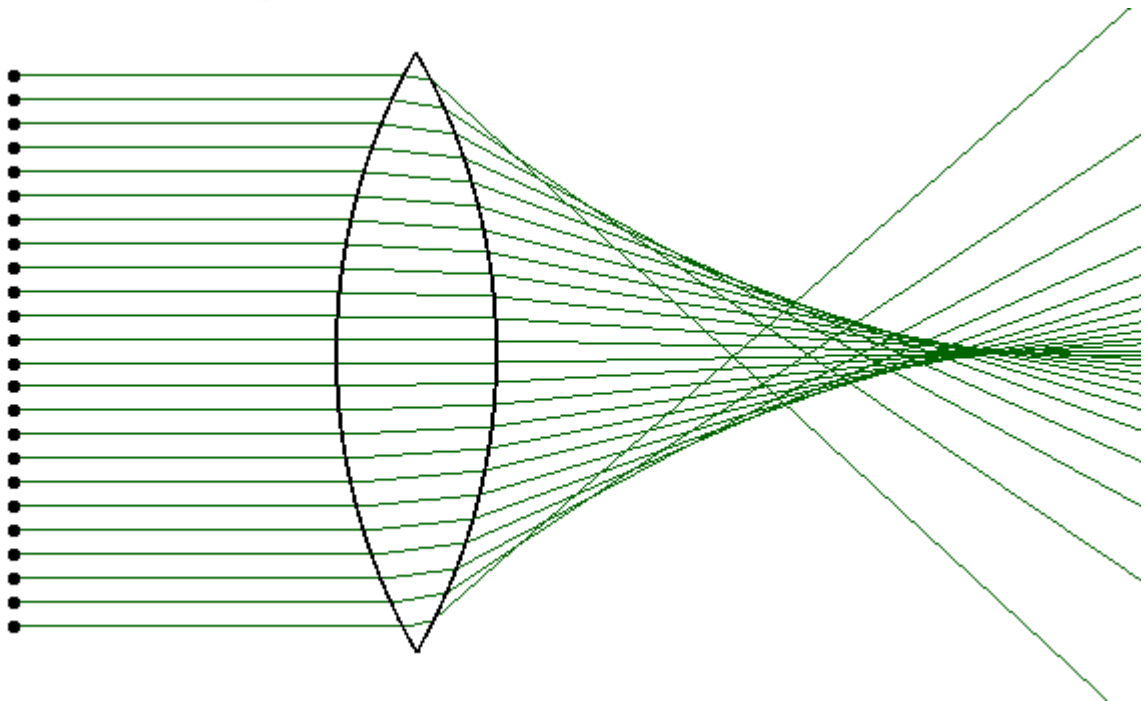


Рис. 23. Сферична аберація у лінзі

Сферична аберація у лінзі або у їх системах пояснюється тим, що її заломлюючи поверхні зустрічають окремі, скільки завгодно віддалені від центру промені, під різними кутами. Через це більш віддалені промені від оптичної вісі лінзи будуть заломлюватися більше, ніж менш віддалені, через що вони створять свої власні точки сходження, віддалені від фокальної площини. Аналогічний опис явища і для сферичного дзеркала – воно буде зустрічати промені під різними кутами, як і лінза. І фокусування в одну точку також не відбудеться.

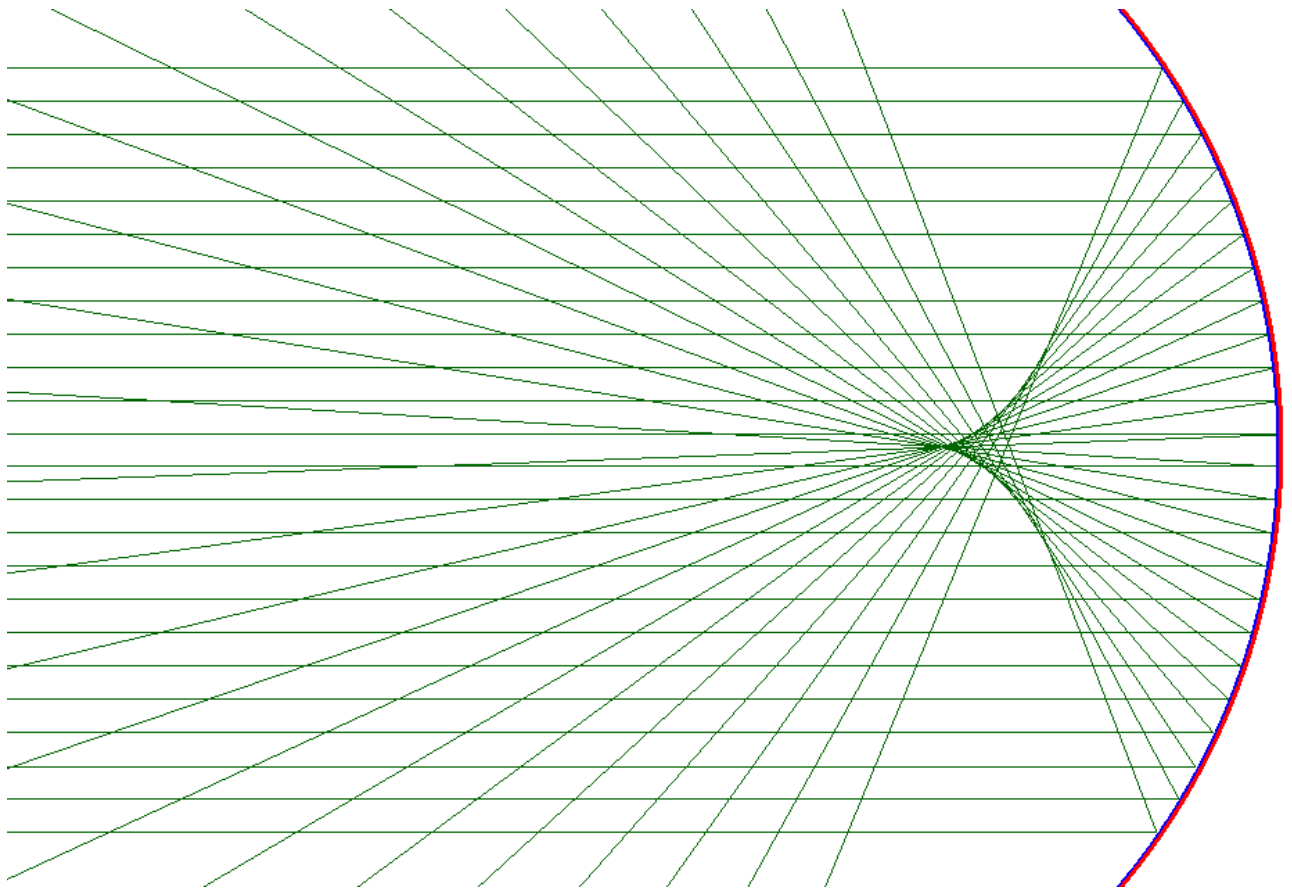


Рис. 24. Сферична аберація у сферичному дзеркалі

Через аберацію виникають огибальні ефекти – каустики. Можна побачити, що промені через те, що не сходяться у одній точці, а сходяться на прямій – утворюють криву поверхню. У геометричній оптиці каустики являють собою абсолютно тонку поверхню, або криву у двовимірному випадку.

За допомогою моделюючої програми можна побачити наявність нелінійних особливостей огинаючих променів. Ось приклади:

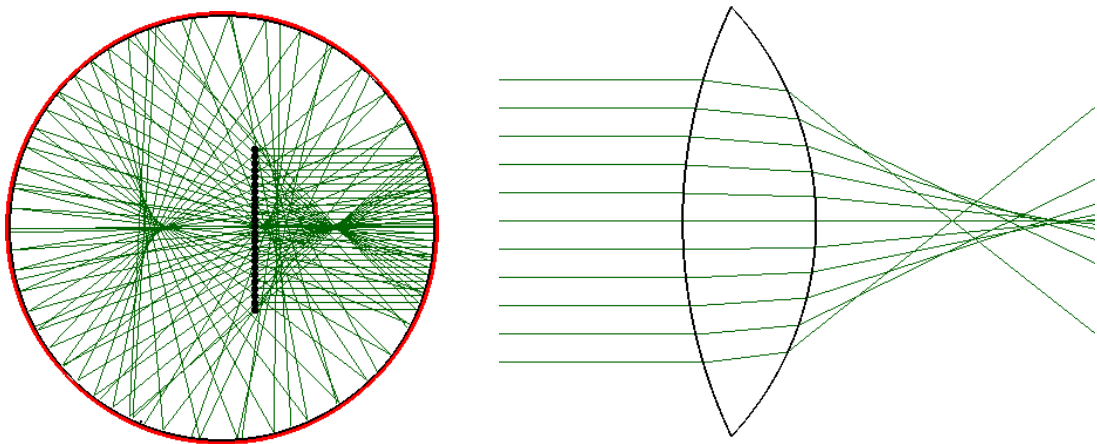


Рис. 25. Каустики у дзеркалі та лінзі

Опис «Оптичного конструктора»

Можливості програми

У ході праці був створений «Оптичний конструктор». Що ж це за програма? «Оптичний конструктор» – програма, що дозволяє змодельовати хід променя крізь лінзи, дзеркала та кулі, як через окремі тіла або як через систему тіл, зібрану з лінз, куль та дзеркал у будь-якій конфігурації. До того ж лінзи у конструкторі не ідеальні, а більш приближені до реальності – товсті та з їх недоліками, такими як сферична аберація, повне внутрішнє відображення і т.п. Аналогічно сферичні дзеркала також не фокусують промені в одну точку, а діють як їх реальні «аналоги» – тобто вони також мають, наприклад, сферичну аберацію. Променів, світлих точок, сферичних та плоских дзеркал, лінз може бути скільки завгодно (докладніше в опису обмежень програми). Усі побудови, що наведені у роботі, окрім побудов з тонкою лінзою – виконані у цьому конструкторі.

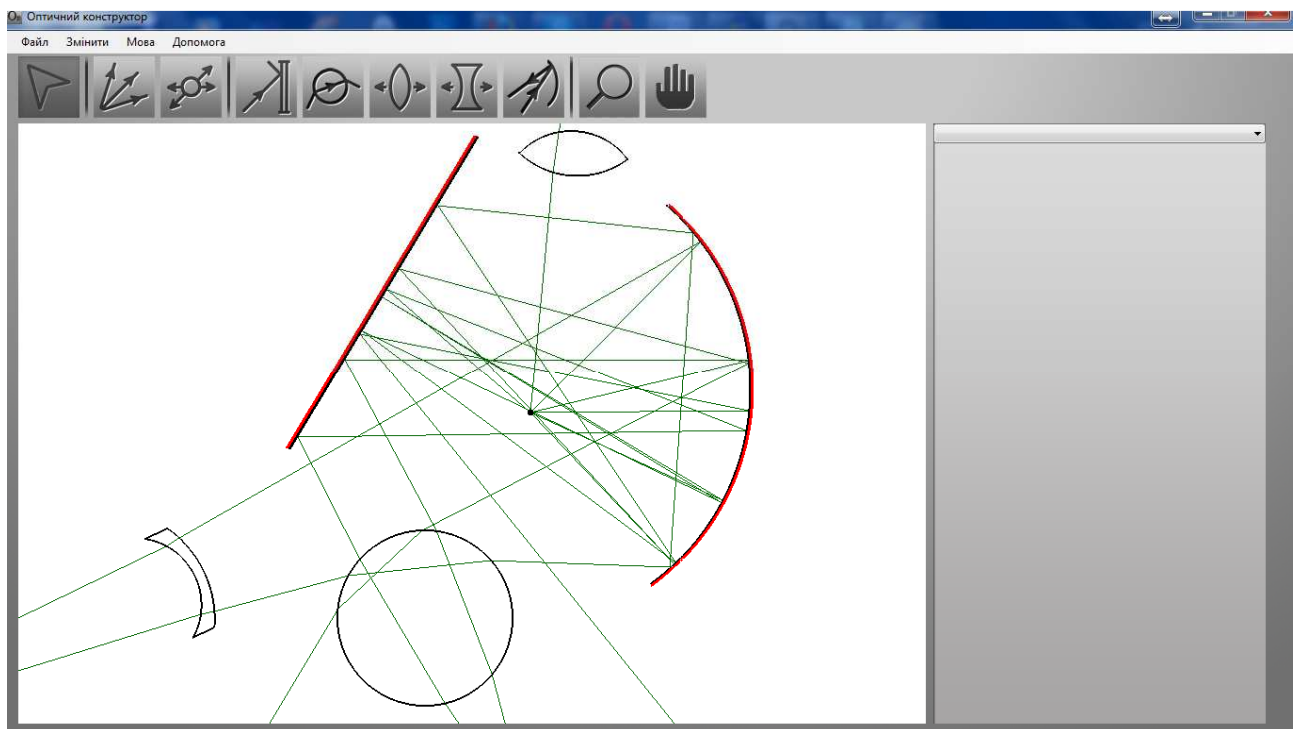


Рис. 26. Побудова, виконана у конструкторі. Тут: сім променів, куля, дві лінзи, сферичне дзеркало, плоске дзеркало, світла точка.

Процес побудови повністю автоматизований – користувачу достатньо лише створити систему та пустити крізь неї промені – програма сама побудує хід променів крізь систему.

Принципи побудови програми та декілька слів про мову програмування

Програма побудована на принципах об'єктно-орієнтованого програмування, та реалізує усі три принципи.

Наслідування реалізовано наступним чином: кожен фізичний об'єкт (з точки зору програмування промінь також легше вважати об'єктом, тому далі мова буде йти лише про об'єкти) наслідує кореневий об'єкт (його код наведено у документації). Це значно полегшило написання програми, бо без цього кожен тип довелося розглядати окремо, а за цим принципом побудови променю не потрібно знати, від якого саме типу об'єкту він відбився.

Поліморфізм реалізовано так: кожен фізичний об'єкт, що має оптичну густину, наслідує інтерфейс, який описує йому функції для об'єктів з оптичною густиною. Це також полегшило написання променя, бо тепер є лише два випадки побудови променя - фізичний об'єкт, та фізичний об'єкт що має густину. Саме тому до програми легко додати інші об'єкти.

Інкапсуляція присутня в кожному об'єкті програми. Тобто, лінія «вміє» перетинатися з іншими лініями та кулями і т.п. Також, лінія вміє конструюватися за двома точками, а куля – за трьома. Це дозволяє перенести геометрію, пов'язану з перетином фігур на більш низький рівень. Тобто коли я писав сферичне дзеркало, я вже не замислювався над перетином променя та частини кола, а визивав вже існуючий метод і т.д.

Форми відрізані від коду ядра – тобто є окремий об'єктний менеджер, що «завідує» усією оптичною системою у цілому, і програма може звернутися до цих об'єктів лише через менеджера. Це робить код зрозумілішим для тих, хто буде його відкривати у майбутньому, а також ще полегшує написання та розширення програми.

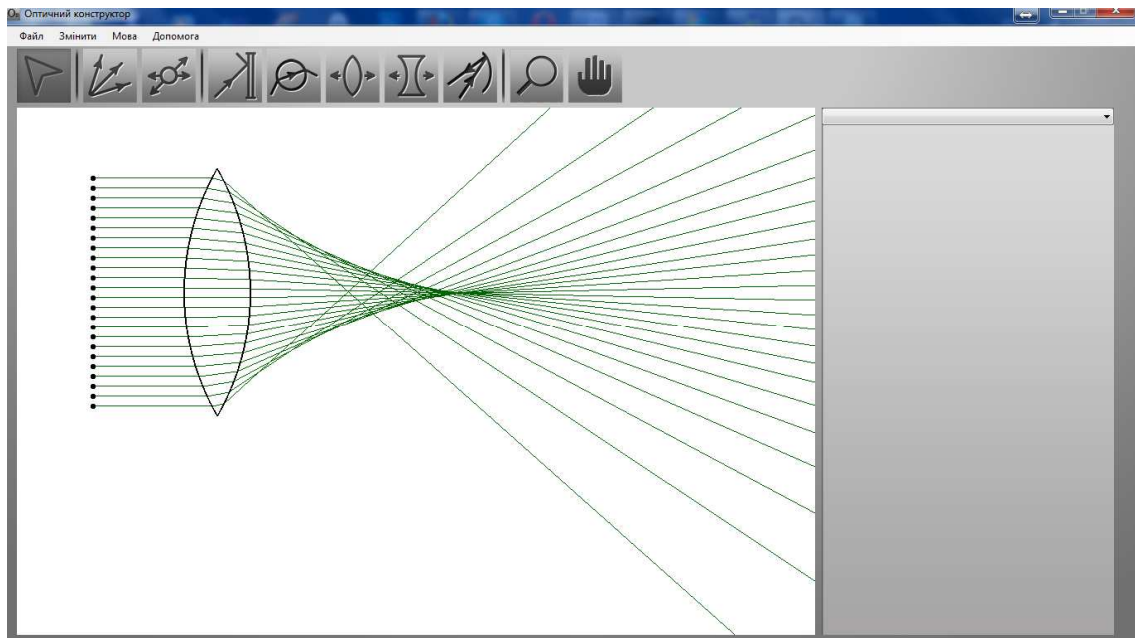


Рис. 27. Головне вікно програми. Показана сферична аберація світла

У програмі наявна лише дві форми – головна та форма налаштувань. Додавати інші форми було б незручно користувачу. На головній формі знаходиться робочий простір, інструменти, полоса меню, полоса статусу та поле редагування об'єкту. На полосі меню знаходяться меню «Файл», «Змінити», «Мова», «Допомога». У «Файлі» знаходяться пункти «Відкрити», «Зберегти», «Зберегти як...», «Закрити проект», «Очистити робочий простір» та «Вихід». Назви говорять самі за себе. У пункті «Змінити» наявні підпункти «Скопіювати», «Вставити» та «Налаштування». Тут також усе зрозуміле. У «Мові» обирається одна з трьох мов: англійська, російська та українська. Також користувач може перевести програму на іншу мову. Докладніше про це – нижче. Пункт «Допомога» веде до файлу довідки з програми.

У вікні налаштувань користувач може змінити декілька параметрів та додати свій власний файл переведення інтерфейсу на іншу мову.

Програма написана на мові програмування C# – однієї з найсучасніших мов програмування для Windows. Також був задіяний TaoFramework (а саме OpenGL для прискорення графіки у режимі 2D), та .NET Framework 2.0. Спочатку планувалося використання .NET Framework 4.0, але ж відкомпілювавши програму під другу версію цього фреймворку, я побачив, що вона працює і на ньому, і вирішив використати цей старіший фреймворк щоб програму можна було запустити на ширшому спектрі комп'ютерів. Швидкість роботи програми від цього не впала.

Можливості переведення інтерфейсу користувачем

Одна з оригінальних можливостей програми – це додання мови, яке може зробити користувач, якщо він зміг перевести програму на свою мову. Усе, що для цього потрібно – взяти існуючий файл переведення, та кожен рядок цього файлу перевести, та у такому ж самому порядку розмістити у новому файлі. Після чого цей новий файл можна дати конструктору, і він розмістить у своєму меню нову мову, яку можна відразу ж і використати.

Обмеження програми

До програми введені п'ять обмежень. Перші два – лише умовні, а третє має суттєвий фізичний зміст. Четверте та п'яте обмеження – виходять з логіки.

Перше та друге обмеження – на кількість об'єктів та променів. Програма оптимізована для роботи з 10 променями та 10 об'єктами. Це так, бо одна з цілей використання цієї програми – у сфері навчання. Вчитель повинен знати, коли учні можуть перестати сприймати матеріал через надвелику кількість об'єктів чи променів, тому програма про це повідомить. Але обидва обмеження не жорсткі, і дозволивши програмі додати більше ніж 10 об'єктів/променів, можна будувати і такі системи, але побудова променя у реальному часі (через декілька мілісекунд) може дещо сповільнитися.

Третє обмеження має суттєвий фізичний зміст. Можна будувати лише 500 віддзеркалень/заломлень одного променя. Це так, бо є такі системи, де промінь буде відбиватися нескінченно довго (як показано вище), і якщо працювати без цього обмеження – програма може легко «зациклитися». Цю кількість можна збільшити або зменшити у налаштуваннях програми, але зовсім відключити – неможна.

Четверте та п'яте обмеження такі: неможна, щоб оптичні об'єкти перетиналися (окрім променя з усіма іншими об'єктами), та неможна розмістити яскраву точку у внутрішній частині об'єкту, що має об'єм. Обидва обмеження виходять, якщо міркувати так: «Усі об'єкти суцільні та однорідні за внутрішнім складом, тому як же вони можуть перетинатися та мати у собі якесь мале тіло?».

Напрями використання

- Уроки з фізики при поясненні теми геометричної оптики
- Уроки з астрономії при поясненні принципу роботи телескопу
- Проектування різноманітних оптичних систем будь-якої складності
- Проведення досліджень з ходу променя крізь оптичні системи
- Використання програми в цілях самонавчання

Висновок

У ході роботи були розглянуті усі закони геометричної оптики. Було вирішено декілька практичних та написана моделююча програма з широким спектром використання. Задачі були вирішені не тільки з геометричної оптики, а і з механіки, тобто складніші задачі були розглянуті за допомогою простіших законів. Також рішення таких задач показує, що закони геометричної оптики можна використовувати і в інших гілках науки.

Сам конструктор виконаний на високому рівні, має зручний і добре зрозумілий інтерфейс, і допомагає у розумінні матеріалу з курсу оптики. Цей конструктор має велику кількість напрямів використання, тому досить актуальний у наш час. Я вважаю що я досягнув своєї мети, бо створив саме те, що планував створити. Також було глибоко вивчено закони геометричної оптики та вирішено декілька практичних завдань. Хочу винести подяку Кашкіну Ю.І. за допомогу у створенні алгоритмів роботи цієї програми та Козлові Т.І. як науковому керівнику та за допомогу у отриманні знань, яких не вистачало для виконання проекту.

Джерела інформації та використані програмні засоби

Джерела інформації

- «Библия С#», М. Фленов, «БХВ-Петербург», Санкт-Петербург, 2009.
- <http://www.codeproject.com/>
- <http://stackoverflow.com>
- <http://www.google.com>
- «Физика. 11 класс», А.Т. Глазунов, О.Ф. Кабардин, А.Н. Малінін, В.А. Орлов, А.А. Пінскій, «Просвещение», Москва, 2002.
- «Фізика. 8 клас», С.В. Мельничук, П.Ф. Пшенічка, «Генеза», Київ, 2006.
- <http://en.wikipedia.org>
- <http://ru.wikipedia.org>
- <http://uk.wikipedia.org>
- «Уроки фізики. 11 клас», М.І. Тимочків, Навчальна книга – Богдан, Тернопіль, 2007.
- «Математические бильярд», Г.А. Гальперин, А.Н. Земляков, издательство «Наука», 1990.
- «Основы физики. Том 2», Б.М. Яворский, А.А. Пинский, издательство «Наука», Москва, 1972.
- «Физика. Оптика и волны. Часть 2», А.С. Ахматова, издательство «Наука», Москва, 1973.

Використані програмні засоби

- Microsoft Windows 7
- Microsoft Visual Studio 2010 Ultimate
- .NET Framework 2.0
- TaoFramework (OpenGL)
- Microsoft Office
- Mozilla Firefox
- Оптичний конструктор (створений власноруч)

Додатки

Програмно-апаратні вимоги конструктора

- Операційна система Windows 98 або старша.
- Встановлений .NET Framework 2.0 або старше.
- OpenGL-сумісна відеокарта з 128 мегабайтами відеопам'яті
- Маніпулятор «Миша»
- Клавіатура

Можна побачити, що будь-який сучасний комп'ютер зможе запустити цю програму.

Фрагменти коду

Кореневий об'єкт

```
using System;
using System.Drawing;
using OpticalBuilderLib.Configuration;
using OpticalBuilderLib.MathLib;

namespace OpticalBuilderLib.OpticalObjects
{
    public abstract class ObjectProto //Прототип об'єкту
    {
        public event EventHandler<EventArgs> ObjectChanged; //Об'єкт
        змінився - подія
        public void RaiseChanged() //Об'єкт змінився - пробудити подію
        {
            if(ObjectChanged!=null) ObjectChanged.Invoke(this, new
            EventArgs());
        }
        protected string name;
        public string Name //Зміна та отримання імені об'єкту
        {
            get { return name; }
            set
            {
                if (this is Ray) //Якщо об'єкт - промінь
                {
                    if (!(ObjectCollection.Instance.RayExists(value)))
                    //І такого імені не існує
                    {
                        name = value; //Задаємо ім'я
                        ObjectCollection.Instance.RaiseObjectsChange();
                    }
                }
            }
        }
    }
}
```

```

    }
    else //Якщо об'єкт - не промінь
        if (!(ObjectCollection.Instance.ObjectExists(value)))
//I такого імені не існує
        {
            name = value; //Задаємо ім'я

            if (this is BrightPoint) //Якщо об'єкт - джерело
світла
            {
                ((BrightPoint)this).InvokeNameChange();
//Вказуємо проміням на зміну імені джерела світла
            }
            ObjectCollection.Instance.RaiseObjectsChange();
//Вказуємо на зміну об'єктів
        }
        RaiseChanged(); //Вказуємо на даного об'єкту
    }
}
protected int id;
public bool Selected = false;
public virtual int ID //Внутрішньопрограмний ідентифікаційний
номер
{
    get
    {
        return id;
    }
    set
    {
        if ((ObjectCollection.Instance.ObjMaxID == value) ||
(ObjectCollection.Instance.Allow_List_Access == true))
            id = value;
        RaiseChanged();
        ObjectCollection.Instance.RaiseObjectsChange();
    }
}
protected SystemCoordinates coordinates;
public SystemCoordinates Coordinates //Координати центру об'єкту
{
    get { return coordinates; } //отримання
    set
    {
        if(!(this is Ray)) MoveTo(value, coordinates); //якщо
об'єкт не промінь - переміщуємо зі старих до нових координат
        RaiseChanged(); //Оповіщуємо про зміну об'єктів
        ObjectCollection.Instance.RaiseObjectsChange();
    }
}
}

```

```

        public abstract string GetTypeSpecifier(); //Отримання
специфікатора типу - реалізується у самих об'єктах
        public abstract void Drawer(); //Малювання об'єкту на моніторі -
реалізується у самих об'єктах
        public abstract string GenerateSaveString(); //Генерування строки
зберігання - реалізується у самих об'єктах
        public abstract SystemCoordinates IntersectWithLine(out bool
PointFound, Line ToIntersect, out bool angle_got, out Angle anglee, Ray
ray); //Перетин з променем
        public override string ToString()//Метод конвертації об'єкта в
строку - внутрішньопрограмний
        {
            return name;
        }
        public static string GenName(string typespec, bool ray = false)
//Метод генерування імен об'єктів автоматично
        {
            string SubName = typespec.Substring(1);
            SubName = SubName.Substring(0, SubName.Length - 1);
//Отримуємо підстроку специфікатора
            SubName = STranslation.T[SubName]; //Переводимо на обрану
користувачем мову
            bool gened = false;
            for(int i = 1; !gened; i++) //доки не знайдемо вільного імені
- шукаємо його
            {
                if (!ray) //якщо не промінь
                {
                    if (!(ObjectCollection.Instance.ObjectExists(SubName
+ i))) //ім'я знайдено - виходимо з метода
                    {
                        gened = true;
                        return SubName + i.ToString();
                    }
                }
                else //інакше
                {
                    if (!(ObjectCollection.Instance.RayExists(SubName +
i))) //ім'я знайдено - виходимо з метода
                    {
                        gened = true;
                        return SubName + i.ToString();
                    }
                }
            }
            return SubName; // ніколи не виводиться, але потрібно
компілятору
        }
        public abstract int DistanceToPointS(Point X);
        public virtual bool IsPreloml() //Чи є даний об'єкт линзою чи
кулею? - переписується у самому об'єкті
        {

```



```

        return false;
    }
    public virtual void MoveTo(SystemCoordinates to,
SystemCoordinates from) //переміщення об'єкту з одних координат в інші -
дописується у об'єктах
    {
        coordinates = new SystemCoordinates(coordinates.X + to.X -
from.X, coordinates.Y + to.Y - from.Y);
    }
    public virtual void BeingRemoved() //Дії при видаленні об'єкту
    {

    }
    public virtual void Rotate(SystemCoordinates to,
SystemCoordinates from) //Поворот (по куту між вектором A(coordinates,
from) и вектором B(coordinates, to)
    {

    }
    public static string GetSpec(int id) //Генерування специфікатора
за номером (не плутать з ідентифікатором об'єкту)
    {
        if (id == 1) return "{Ray}";
        if (id == 2) return "{Mirror}";
        if (id == 3) return "{Sphere}";
        if (id == 4) return "{BrightPoint}";
        if (id == 5) return "{Lense}";
        if (id == 6) return "{SphericalMirror}";
        return "{null}";
    }
    public static string GetSpec(ObjectTypes type) //Генерування
специфікатора за типом
    {
        if (type == ObjectTypes.Ray) //Промінь
            return GetSpec(1);
        if (type == ObjectTypes.Mirror) //Дзеркало
            return GetSpec(2);
        if (type == ObjectTypes.Sphere) //Куля
            return GetSpec(3);
        if (type == ObjectTypes.BrightPoint) //Джерело світла
            return GetSpec(4);
        if (type == ObjectTypes.Lense) //Линза
            return GetSpec(5);
        if (type == ObjectTypes.SphereMirror) //Линза
            return GetSpec(6);
        return GetSpec(-1); //жоден з типів
    }
}
public enum ObjectTypes //Типи об'єктів
{
    Ray = 1,
    Mirror = 2,

```

```

        Sphere = 3,
        BrightPoint = 4,
        Lense = 5,
        SphereMirror = 6,
    }
}

```

Код перетину кола та прямої

Даний код належить колу, тому усі змінні, що не були оголошені – оголошені у кодї кола

```

public List<SystemCoordinates> IntersectionWithLine(Line line)
{
    List<SystemCoordinates> ToReturn = new
List<SystemCoordinates>();
    if (line.Vertical == true)
    {
        //Якщо пряма вертикальна, то line.B == X;
        if ((line.B < (center.X - radius)) || (line.B > (center.X
+ radius)))
        {
            //Якщо вертикальна лінія не торкається кола - нічого
не робимо
        }
        else
            if ((line.B == (center.X - radius)) || (line.B ==
(center.X + radius)))
            {
                //Якщо вертикальна лінія дотикається до кола -
додаємо координати дотику
                SystemCoordinates Coords = new
SystemCoordinates(line.B, center.Y);
                ToReturn.Add(Coords);
            }
            else
                if (line.B == center.X)
                {
                    //Якщо вертикальна лінія перетинає коло по
діаметру - додаємо координати перетину
                    ToReturn.Add(new SystemCoordinates(line.B,
center.Y + radius));
                    ToReturn.Add(new SystemCoordinates(line.B,
center.Y - radius));
                }
            else
            {
                //Якщо вертикальна лінія перетинає коло не по
діаметру - додаємо координати перетину

```

```

        ToReturn.Add(new SystemCoordinates(line.B,
center.Y + Math.Sqrt(radius * radius - (line.B - center.X) * (line.B -
center.X))));
        ToReturn.Add(new SystemCoordinates(line.B,
center.Y - Math.Sqrt(radius * radius - (line.B - center.X) * (line.B -
center.X))));
    }
    }
    else
    {
        //Якщо пряма не вертикальна, то використовуємо формули,
виведені з таких двох рівнянь:
        //y = k*x + b
        //(x-a)^2 + (y-b)^2 = R^2
        DoubleExention C = -(radius*radius - center.X*center.X -
center.Y*center.Y - line.B*line.B +
        2*line.B*center.Y);
        DoubleExention B = 2*(line.K*line.B - center.X -
line.K*center.Y);
        DoubleExention A = 1 + line.K*line.K;
        DoubleExention D = B*B - 4*A*C;
        if(D == 0) //Якщо пряма дотикається до кола
        {
            DoubleExention X = -B/(2*A);
            DoubleExention Y = line.K*X + line.B;
            ToReturn.Add(new SystemCoordinates(X, Y));
        }
        else if (D > 0) //Якщо пряма пернетає коло
        {
            DoubleExention X1 = (-B + D.sqrt())/(2*A);
            DoubleExention X2 = (-B - D.sqrt())/(2*A);
            ToReturn.Add(new SystemCoordinates(X1, line.K * X1 +
line.B));
            ToReturn.Add(new SystemCoordinates(X2, line.K * X2 +
line.B));
        }
    }
    List<SystemCoordinates> Toreturn = new
List<SystemCoordinates>();
    foreach (var e in ToReturn)
    {
        if(line.IsBetweenEnds(e))
            Toreturn.Add(e);
    } //Перевірка на належність отриманих координат прямиї (бо
пряма може бути променем або відрізком)
    if(ULimitSet && LLimitSet)
    {
        List<SystemCoordinates> a = new
List<SystemCoordinates>();
        foreach(var z in Toreturn)
            if(PointIsBetweenEdges(z)) a.Add(z);
        Toreturn = a;
    }
}

```

```
    } //Перевірка на належність отриманих координат колу (бо коло  
може бути лише частиною кола, а не повним колом)  
    return Treturn;  
}
```