**Дніпровський ліцей інформаційних технологій**

**при Дніпровському національному університеті**

**імені Олеся Гончара**

**Випускна робота**

**на тему:**

*Моделювання руху зарядженної частинки біля незарядженної металевої поверхні*

**Виконавець:**

**ліцеїст 11-Г класу**

**Зима Кирило**

**Керівник роботи:**

**Ентін Йосиф Абрамович, викладач Дніпропетровського ліцею інформаційних технологій**

**Зміст**

[Зміст](#cont)

[Вступ](#entry)

* [Актуальність](#entry1)
* [Мета](#entry2)

[Теоретична частина з предмету](#teor1)

[Теоретична частина з інформатики](#teor2)

[Етапи створення роботи](#stages)

* [”back-end”](#st1)
* [”front-end”](#st2)

[Висновки](#res)

[Використані програмні засоби](#soft)

[Джерела](#source)

**Вступ**

**Актуальність:**

*Моделювання руху зарядженої частинки поблизу незарядженої поверхні необхідне, оскільки розуміння цієї взаємодії може дати нам важливе уявлення про поведінку обох сторін. Моделювання руху зарядженої частинки поблизу незарядженої поверхні - це не просто корисна інтелектуальна вправа; вона насправді дуже важлива в ускладнених задач для врахування цього фактору у комплексних задачах з комп’ютерного моделювання, призначених для точних задач мікроскопічного уровня. Це може дозволити нам спостерігати і передбачувати їхній рух за власним бажанням, або, можливо, навіть використати його для відкриття раніше недоступних або недосліджених явищ.*

**Мета:**

*Візуальне зображення више вказаної взаємодії для наукових досліджень в області мікрочастинок (при використанні паралельно з або у складі іншого ПО)*

**Теоретична частина з предмету**

**Моделювання***— це метод дослідження об'єктів пізнання (явищ, пристроїв, процесів), що ґрунтується на заміні конкретного об'єкта досліджень (оригіналу) іншим, подібним до нього (моделлю).*

**Теоретична частина з інформатики**

**Програма***(застосунок, прикладна програма або додаток) — користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача. Поняття введене, щоб ідкреслити відмінність від операційної системи, драйверів, бібліотек, системних утиліт тощо (які забезпечують функціювання власне комп'ютерної системи та підтримують її працездатність) та засобів і середовищ розробки.*

**Фреймворк***(Програмний каркас) — інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду.*

**GUI***(Graphical User Interface або Графічний Інтерфейс Користувача) - це система інтерактивних візуальних компонентів для комп'ютерного програмного забезпечення. Графічний інтерфейс користувача відображає об'єкти, які передають інформацію та представляють дії, які може виконати користувач. Об'єкти змінюють колір, розмір або видимість, коли користувач взаємодіє з ними.*

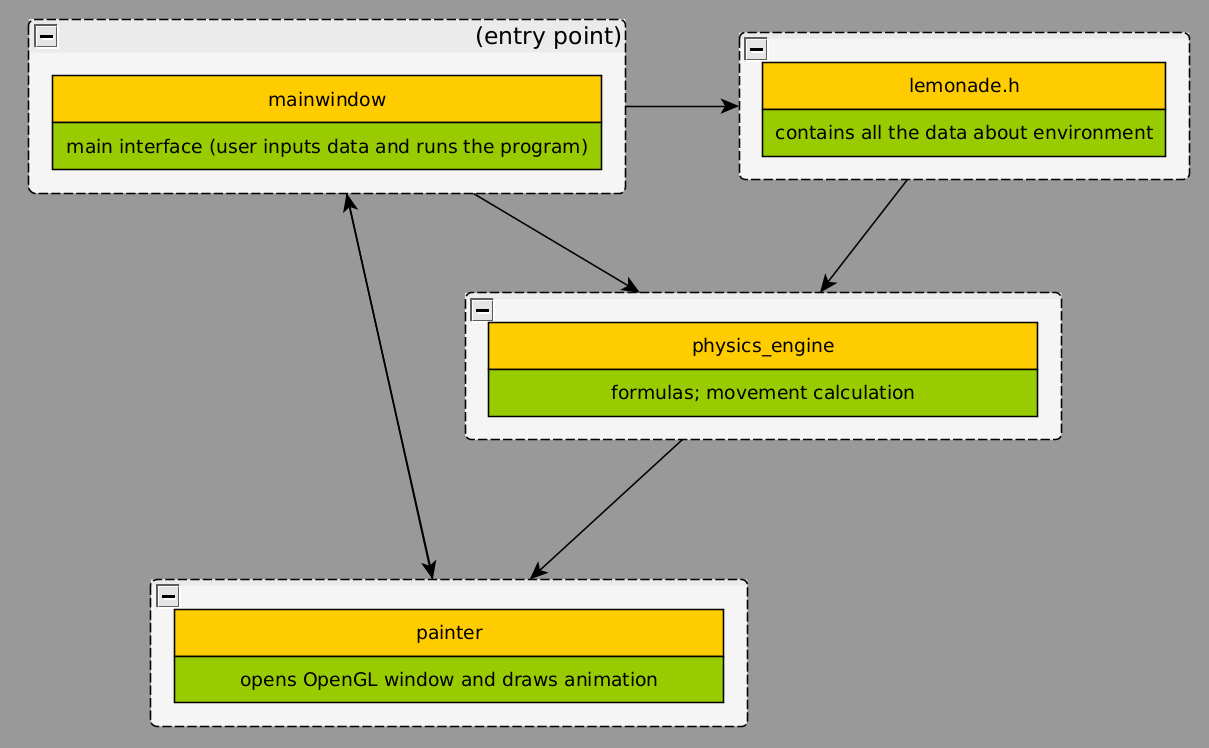
**Етапи створення роботи**

**Створення “back-end” частини проекту**

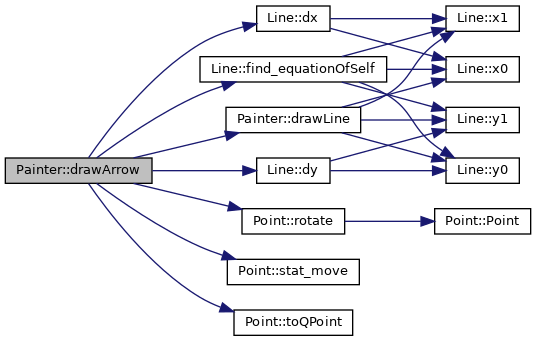
**1.1 Вибір мови, середовища**

*С++ - дуже потужна та швидка мова програмування, в основному, за рахунок прямої компіляції (без використання інтерпретатору), а Qt – досить популярний фреймворк для неї, тому вони і були обрані для цього проекту.*

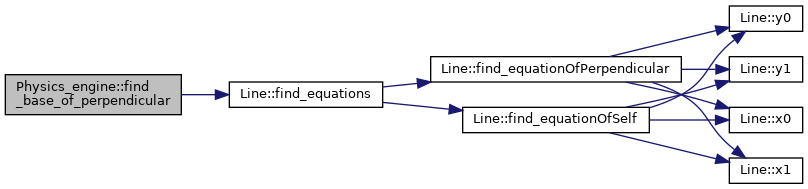
**1.2 Розробка основного логічного каркасу програми (класи, файли, зв’язки)**

*Початкова структура на етапі планування:*

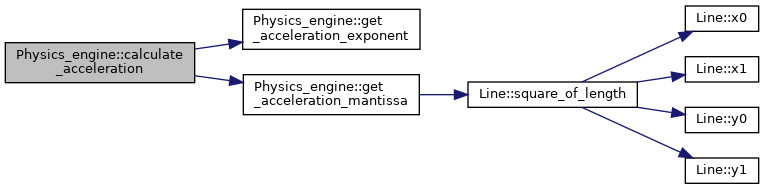
*Теперішня структура (окремі частини):*



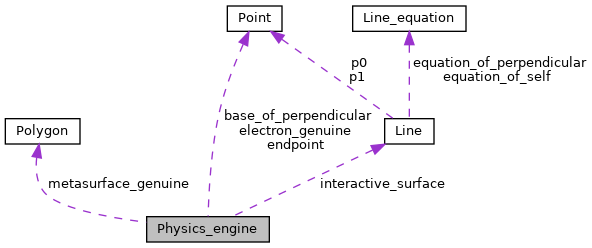
Функції, які функція drawArrow(…) у класі Painter задіює напряму чи побічно



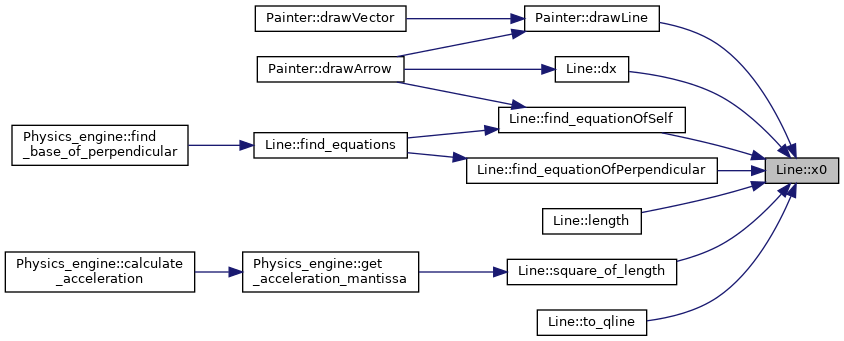
Функції, які функція find\_base\_of\_perpendicular(…) у класі Physics\_engine задіює напряму чи побічно



Функції, які функція calculate\_acceleration(…) у класі Physics\_engine задіює напряму чи побічно



Деякі об’єкти (назви стрілок), які містяться у класі Physics\_engine із вказанням їх типів (у прямокутниках)



Функції, які напряму чи побічно використовують функцію x0() у класі Line

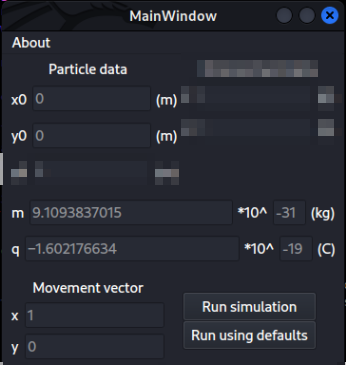
**1.3 Робота з формулами**

*Аналіз поставленної задачі та виведення необхідних формул, внесення їх до проекту, визначення дефолтних даних*

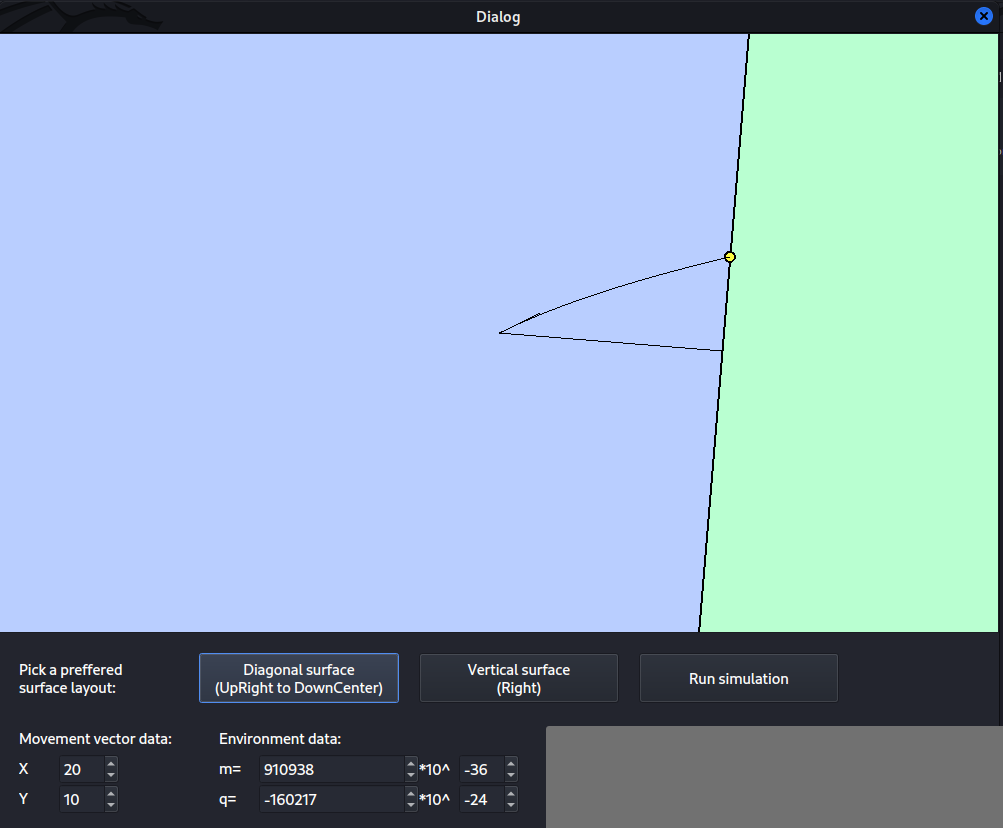
**Створення “front-end” частини проекту**

**2.1 Інтерфейс**

*Створення дизайну головного вікна, встановлення взаємозв’язків між частинами програми, опрацювання вводу даних користувачем*



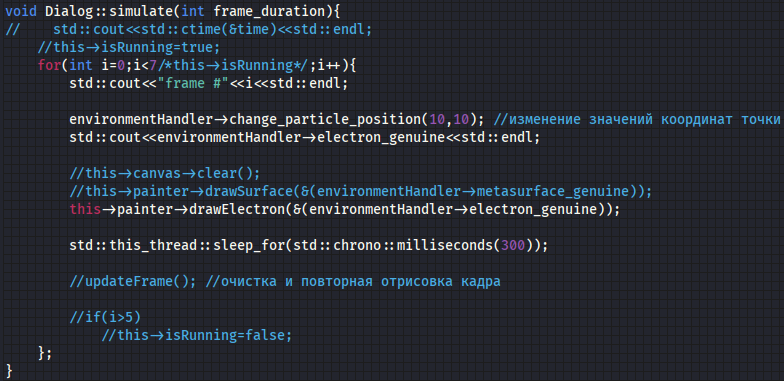
Минулий дизайн



Теперішній дизайн

**2.2 Вибір засобу подання анімації (засоби, класи)**

*Спочатку планувалося використовувати відкриту графічну бібліотеку OpenGL**, проте виявилося, що драйвери під Linux, які постачає компанія Nvidia для моєї графічної карти, не підтримують роботу з OpenGL, яка заявлена виробником. (На жаль, приклади коду не були збережені в архів)*

 *Варіант 2: Циклічна призупинка потоку і попіксельна перемальовка зображення з використанням Qpixmap для зображення та this\_thread::sleep\_for() для зупинки потоку (входить до стандартної бібліотеки С++, детальніше* ***[ТУТ](https://en.cppreference.com/w/cpp/thread/sleep_for)****)*

Один з проміжних варінтів з архіву, намагався дебажити, тому деякий код з коментарів необхіден для функціонування

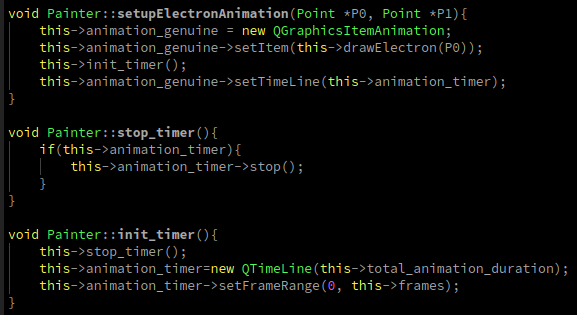
*Цей варіант видався також невдалим, так як особивості архітектури використовуваного фреймворка заважають корректній його роботі при використанні призупинки потока.*

*Варіант 3: Використання вбудованого класу [QGraphicsItemAnimation](https://doc.qt.io/qt-5/qgraphicsitemanimation.html) для створення анімацій. (див. зображення у розділі 2.3)*

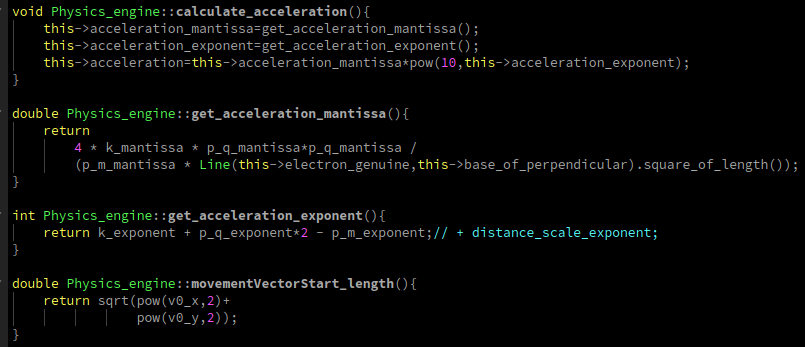
*Демонстраційне відео роботи доступно для перегляду за посиланням: <https://www.youtube.com/watch?v=WoGYTei7aeY>*

**2.3 До-/переробка структури проекту(із врахуванням графіки)**

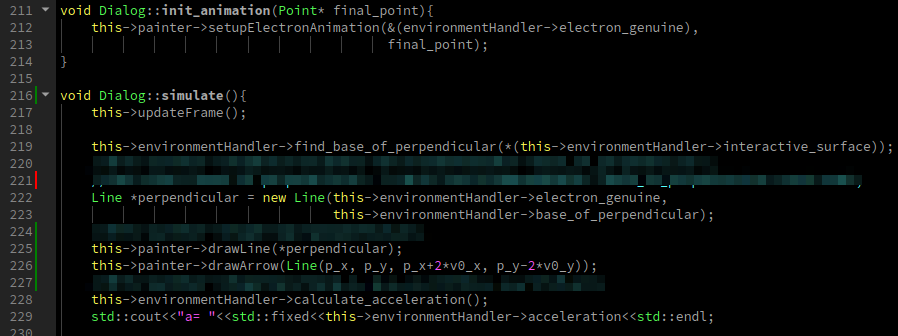
*Спочатку за анімацію повинен був відповідати Physics\_engine, зараз ці функції розподілені між Dialog (створення та оркестрація об’єктів для анімації), Physics\_engine (підрахунок руху за формулами і повернення результату у Dialog) та Painter (відображення графічних елементів у GUI та оперування життєвим циклом об'єкту анімації)*



painter.cpp



physics\_engine.cpp



dialog.cpp

**Висновки**

Під час виконання роботи я зрозумів, що цілком можливо змоделювати рух зарядженої частинки біля незарядженої металевої поверхні, і зробив це за допомогою мови програмування C++ та фреймворку Qt, з яких не мав жодних попередніх знань. Проект ще раз підкреслює потенціал C++ та Qt Framework для розширення можливостей сучасної науки та демонструє їх гнучкість для вирішення нових задач. У процесі роботи я покращив свої знання в області програмування, а саме дізнався про наступні поняття і оволодів ними: вказівники(pointers), «файли заголовків» (header files), vectors (динамічні масиви у С++), lambda-функції, деякі ключові слова, такі як constexpr, extern, explicit та friend. Також я отримав досить повне представлення про синтаксис С++, і, незважаючи на певні недоліки цієї мови, вирішив використовувати її для майбутніх професіональних проектів. Також під час створення проекту я в більшому степені опанував такі програми, як Doxygen, Qt Creator, CherryTree та yEd.

**Використані програмні засоби**

1. **Qt** - крос-платформовий інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни початкового коду. Містить всі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML.
2. **Qt Creator** - інтегроване середовище розробки, призначене для створення крос-платформових застосунків з використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в якій використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками.
3. **Qt Designer** - це інструмент Qt для проектування та побудови графічних інтерфейсів користувача (GUI) за допомогою віджетів Qt. Ви можете створювати та налаштовувати ваші вікна або діалоги за принципом "що бачите - те й отримуєте" (“what-you-see-is-what-you-get”-WYSIWYG), а також тестувати їх за допомогою різних стилів та роздільної здатності вікна.

**Джерела**

[C++ Tutorial](https://pages.cs.wisc.edu/~hasti/cs368/CppTutorial/) - Цей підручник призначений для тих, хто вже знайомий з Java та структурами даних і хоче вивчити C++. Він складається з 5 уроків для самостійного вивчення, які передбачають читання нотаток, відповіді на запитання з декількома варіантами відповідей та написання програм на C++. Найкращий спосіб засвоїти матеріал - спробувати самостійно відповісти на питання, а потім скомпілювати і запустити код.

[C++ Annotations](http://www.icce.rug.nl/documents/cplusplus/) — документ, призначений для досвідчених користувачів C (або будь-якої іншої мови, що використовує C-подібну граматику, наприклад, Perl або Java), які хотіли б дізнатися більше про C++ або здійснити перехід на неї. Цей документ є основним підручником для курсів з програмування на C++, які щорічно організовуються в Університеті Гронінгена. Однак “C++ Annotations” не охоплює усіх аспектів мови C++. Зокрема, базова граматика C++ не розглядається, оскільки вона не відрізняється від граматики C. Для того, щоб освіжити цю частину граматики C++, можна звернутися до будь-якої базової книги з мови C.

[Qt 5.15 documentation](https://doc.qt.io/qt-5/index.html) - це онлайн ресурс, який надає документацію, приклади та підручники для розробників, які використовують інструментарій графічного інтерфейсу користувача (GUI) Qt 5.15. Він є необхідним для всіх, хто бажає створювати графічні програми для настільних комп'ютерів, вбудованих пристроїв або мобільних телефонів, використовуючи Qt фреймворк. Він охоплює теми від початку роботи до створення вашої першої програми з графічним інтерфейсом, а також містить вичерпну довідку про всі API Qt 5.15.

[Черноуцан А.И. Метод электростатических изображений //Квант. — 1987. — № 3. — С. 39-42.](https://www.physbook.ru/index.php/Kvant._Метод_электростат._изображений)