Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка імітаційної програми для моделювання процесу ремонту рухомого складу на потокових лініях»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Богдан БАЛУШКІН/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: « Development of a simulation program for modeling the rolling stock repair process on production lines »
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1812: /Bohdan BALUSHKIN/

Scientific Supervisor: /Olena KUROPIATNYK/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Балушкіну Богдану Віталійовичу
1. Тема роботи: «Розробка імітаційної програми для моделювання процесу ремонту рухомого складу на потокових лініях»
Керівник роботи: Куроп'ятник Олена Сергіївна, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Під час виконання цієї роботи необхідно зібрати вимоги, а саме, ознайомитися з предметною областю, переглянути аналоги та визначити, що саме необхідно розробити.
Спроектувати систему, що буде виконувати поставлену задачу, визначити сутності необхідні для реалізації програмного додатка, розробити інтерфейс користувача.
Обрати мову програмування на котрій буде реалізований додаток та розробити програму, що реалізує спроектовану систему.
Протестувати та налагодити розроблений додаток.
Висновки щодо роботи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
Титульний слайд, зображення та схеми методів ремонту вагонів, зображення студентів і викладача, діаграма прецедентів, діаграма класів, ескіз форми, знімки процесу виконання додатку, висновки.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір та аналіз вимог	08.12.21	
2	Зовнішнє проектування	01.02.21	
3	Внутрішнє проектування	01.03.21	
4	Розробка алгоритмів	20.03.21	
5	Розробка програмного забезпечення	01.05.22	
6	Тестування програмного забезпечення	11.05.22	
7	Подання кваліфікаційної роботи до кафедри	13.06.22	
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	22.06.22	

Студент _____ Богдан БАЛУШКІН
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олена КУРОП'ЯТНИК
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 8 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області, а також проводиться опитування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 8 сторінок;

– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи. Складається з 20 сторінок;

– розробка програми – включає в себе вибір мови програмування та розробку алгоритмів необхідних для реалізації проекту. Складається з 3 сторінок;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методом «білої» скриньки. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 11 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складає 3 сторінки;

– додатки – містить текст програми.

Кількість таблиць: 18 штук.

Кількість рисунків: 15 штук.

Ключові слова: трансбордер, модуль, платформа, вагон.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Позиція – блок модулів, що відповідають за виконання одного виду ремонтних робіт;

Модуль – одна платформа, на котру надходять вагони, для виконання ремонтних робіт;

Трансбордер — пристрій для переміщення залізничного вагону з одного рейкового шляху до іншого. Являє собою платформу з залізничною колією, яка рухається по рейках, розташованих поперек.

ВСТУП

Ця робота спрямована на програмне моделювання гнучких потокових виробництв для ремонту рухомого складу. Використана модель складніша за більшість інших, бо у ній кожен вагон окремо проходить по усім модулям. Так при моделі з жорстким зв’язуванням, кожен вагон повинен очікувати поки усі інші виконають необхідні роботи, щоб перейти далі, необхідний лише один тягач, що буде тягнути увесь вагонний склад, а при моделі з напівжорстким зв’язуванням, де усі вагони повинні очікувати поки наступний виконає роботи необхідно по одному тягачу на кожний вагон, котрі будуть перетягувати свої вагони до наступних позицій, коли ті звільняться. То для гнучкої моделі слід передбачати кілька модулів на кожну позицію, щоб декілька вагонів могли виконувати відповідні роботи й не сповільнювати весь потік.

Мета роботи полягає у створенні додатку, який виконує моделювання процесу ремонту на гнучких потокових виробництвах. А саме, показувати модулі на позиціях, вагони різних типів, що надходять до вільних модулів, де їм необхідно пройти процес ремонту та трансбордер котрий має переміщати ці вагони між модулями.

Різним вагонам необхідний різний час для виконання роботи на кожному модулі. Наприклад, помити досить чистий вагон буде швидше ніж заварити важкодоступний елемент на старому вагоні, але й не бажано, щоб вагони, робота на котрих була вже виконана, попусту простоювали на модулі чим затримували весь останній потік. Тобто кожний модуль, на кожній позиції повинен працювати з максимальною ефективністю, не повинно бути зайвих простоїв, тому необхідно щоб модулів на позиціях де робота займає більше часу – було більше, а де процес ремонту проходить швидше – було менше, щоб не займати зайвий простір у будівлі та люди з обладнанням на цих модулях не простоювали.

Практичне призначення розроблюваного програмне забезпечення (ПЗ) полягає в тому, щоб показати студентам як проходить процес ремонту на гнучких потокових лініях та чому слід балансувати кількість модулів на позиціях.

1 ЗБІР ТА АНАЛІЗ ВИМОГ

Аналіз вимог [5] пов’язаний з цілями й потребами замовника. Під час аналізу вимог слід визначити:

- спосіб використання ПЗ;
- вхідні та вихідні дані;
- спосіб зображення інформації;
- наскільки продуктивним повинен бути додаток;
- надійність додатка;
- інструменти для реалізації проекта.

Тобто результатом виконання збору та аналізу вимог, повинно бути вирішено, що повинен виконувати програмний продукт, як це повинно виглядати та за допомогою яких інструментів виконано.

1.1 Огляд аналогів

В ході дослідження виявлено лише один аналог для моделювання гнучких потокових виробництв для ремонту рухомого складу – **Conveyers Model**.

Інформацію щодо функціонала додатка було надано замовником. Далі розглянемо отриману інформацію.

Conveyers Model – перша версія додатку, яка була розроблена з метою моделювання гнучких потокових виробництв для ремонту рухомого складу. Але він дозволяє лише прорахувати коефіцієнти використання й навантаження модулів.

На рис. 1.1. зображена форма для введення конфігурації моделі, а саме: обрати які методу будуть використані, обрати кількість позицій, кількість модулів на цих позиціях, час переміщення до них та коефіцієнт надійності(передбачається можливість виходу зі строю). Також є можливість розгляду моделі з більше ніж одним трансбордером і щоб вони обслуговували різні позиції.

На рис. 1.2 зображені результати обчислення коефіцієнтів використання та навантаження для усіх модулів.

А на вкладці “Программа ремонта” (рис 1.3) є можливість переглянути номери вагону коли він надійшов на ремонт та під яким номером був відремонтований, тип вагону, вид та час ремонту, час простою та знаходження у цеху.

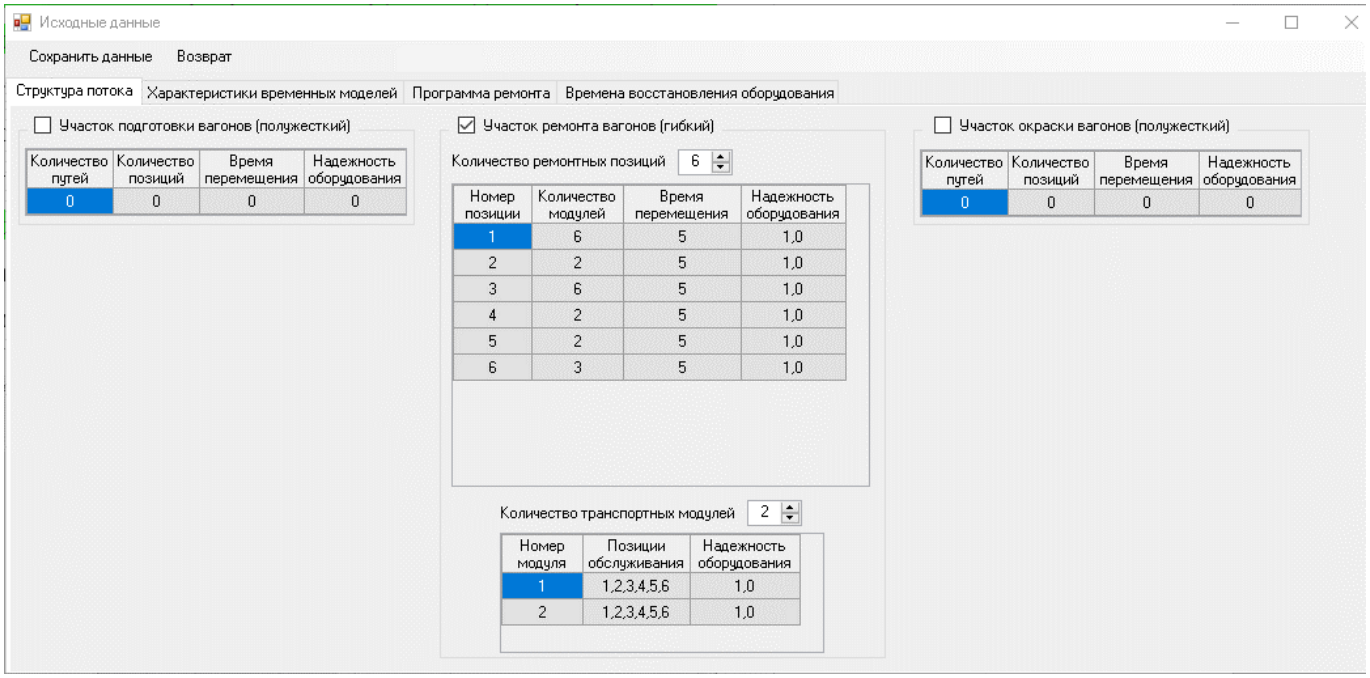

Рисунок 1.1 – Введення значень

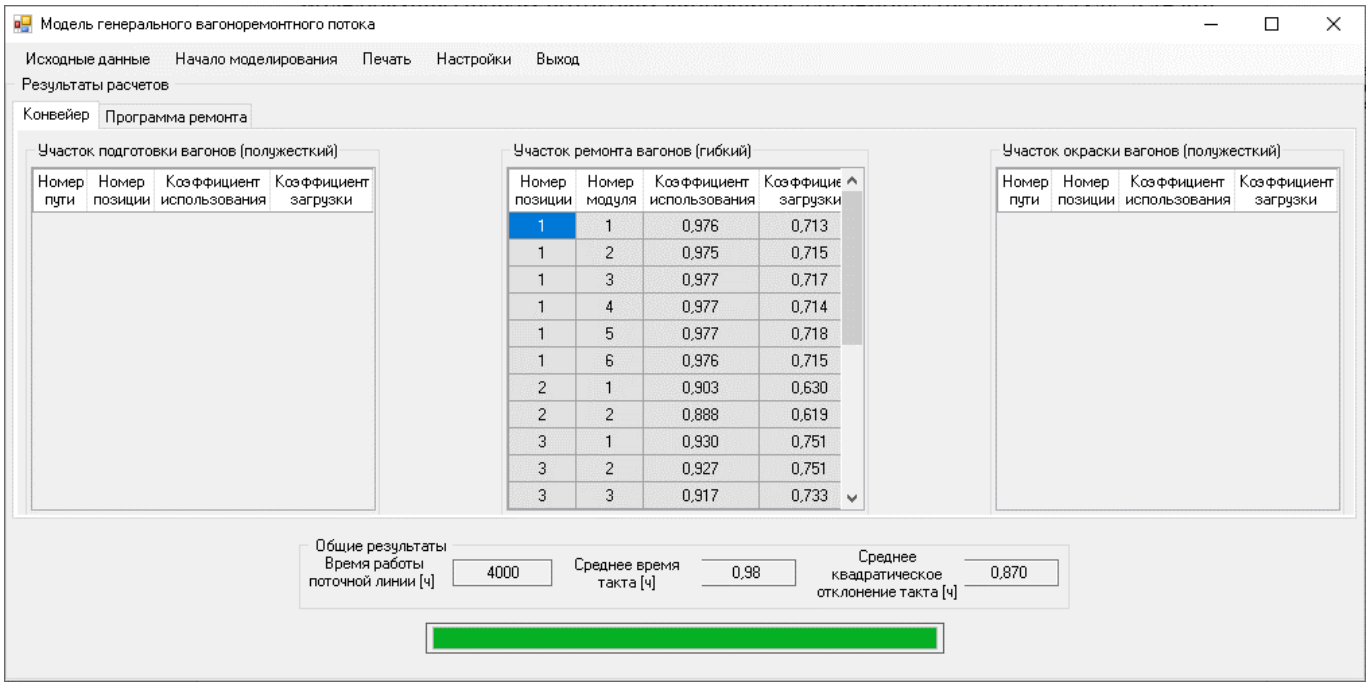

Рисунок 1.2 – Результат розрахунків для конвеєру

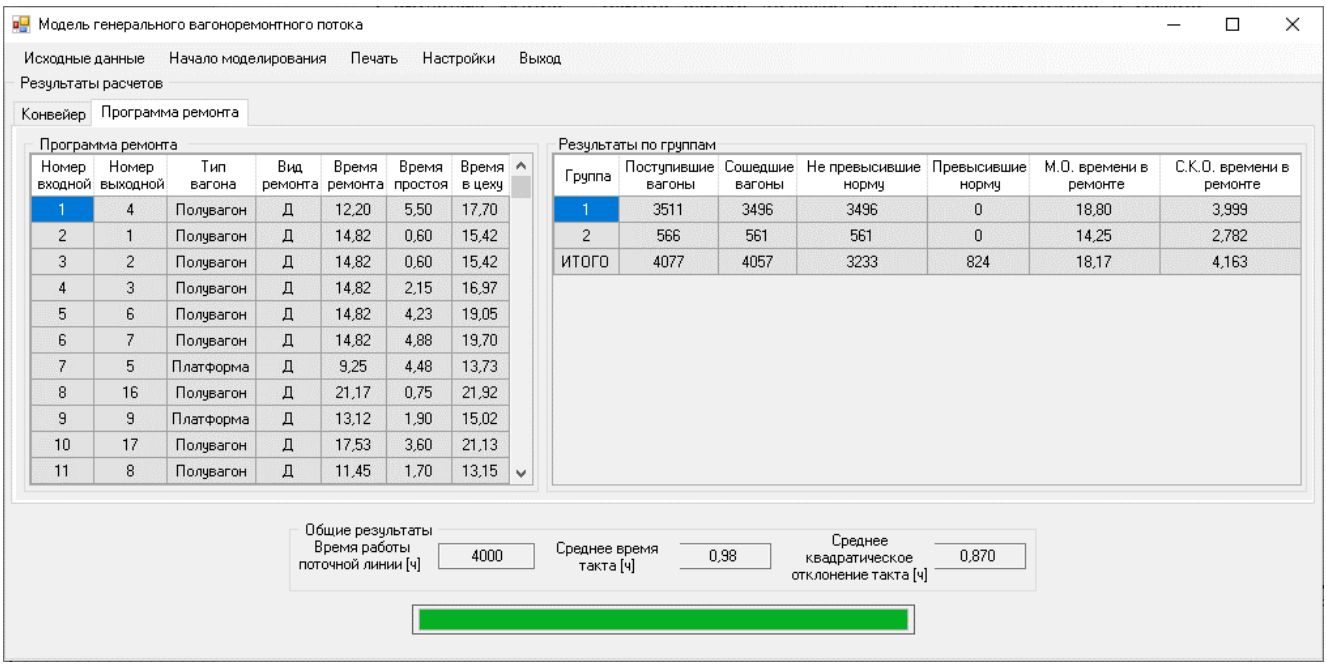

Рисунок 1.3 – Результати розрахунків програми ремонту

1.2 Огляд літератури

Далі розглянемо які бувають методи ремонту рухомих складів за матеріалами [4].

1.2.1 Стаціонарний метод ремонту

При стаціонарному методі ремонту увесь комплекс робіт виконується на одному робочому місці. Поза цього місця виконуються тільки ті операції, для виконання котрих необхідне спеціальне обладнання. Цей метод характеризується низькою продуктивністю, у наслідок чого великою тривалістю ремонтного циклу. Можна виділити два різновиди стаціонарного методу ремонту: стаціонарно-бригадний та стаціонарно-вузловий.

Стаціонарно-бригадний метод заснований на принципі концентрації операцій процесу, виконуваних на одному місці. При цьому методі весь цикл робіт по ремонту вагонів і його частин виконується послідовно на одній позиції однією бригадою універсальних виконавців. Усі вузли та деталі, зняті з вагона, після ремонту встановлюються на той самий вагон.

Стаціонарно-вузловий метод заснований на діленні операції, тобто диференціації процесу на окремі операції за технічними вузлами. Тому в цьому випадку увесь цикл ремонтно-збиральних операцій розділяється на вузлову й загальну збірку. Це дозволяє, шляхом ущільнення й паралельності операцій,

скоротити тривалість робіт. У зв'язку з цим, такий метод ремонту отримав більше визнання ніж стаціонарно-бригадний.

Однак обидва різновиди стаціонарного методу ремонту мають недоліки, пов'язані з потребою висококваліфікованих виконавців й спеціальні засоби механізації.

При стаціонарному методі ремонту вагони непотрібно переміщати, через те, що усі роботи виконуються в одному місці. Стаціонарний метод дуже простий в реалізації, однак він не є продуктивним, бо не дозволяє в одній позиції використовувати весь необхідний комплекс технічного обладнання: від мийної машини до фарбувальної й сушильної камер. Єдиною позитивною якістю є повна незалежність від ремонту останніх вагонів, тобто об'єкт ремонту буде ремонтуватися рівно стільки часу, скільки необхідно. Хоча в умовах єдиного залізничного шляху для подачі-видачі вагонів, саме так спроектовані усі вагонні депо, вказана вище "незалежність" отримує залежний характер. На рис. 1.4 представлена структурна схема вагоно збірної ділянки з використанням стаціонарного методу ремонту вагонів.

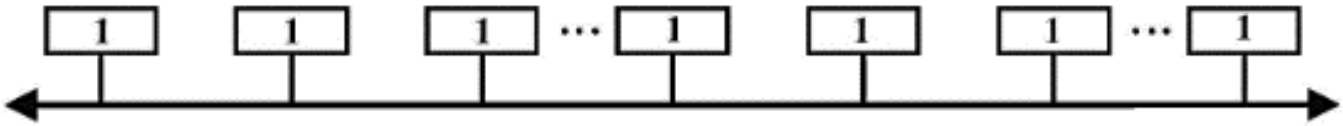

Рисунок 1.4 – Структурна схема розміщення стійл при стаціонарному методі ремонту вагонів

1.2.2 Жорсткий потік

На відміну від стаціонарної форми виробництва, потокова форма має цілий ряд незаперечних переваг. Особливо це видно при масових виробництвах нових виробів.

Як правило, потоковою називається така форма організації виробництва, при якій спеціалізовані позиції розташовуються у строгій послідовності з прийнятим технічним процесом, а предмети праці постійно переміщаються між цими позиціями до повного завершення циклу. Такий метод також називається "поточно-предметним". Можливий варіант, коли засоби праці знаходяться в одному місці, а переміщуються тільки виконавці. Тут ми маємо справу з "поточно-бригадним" методом організації виробництва. Будь-який потоковий метод, навіть у самій простій

реалізації, являється більш ефективним у порівнянні з не потоковим, оскільки дозволяє шляхом спеціалізації позицій й оснащення їх необхідним технологічним обладнанням підвищити продуктивність праці.

На рис. 1.5 представлена структурна схема потокової лінії з жорстким зв'язуванням між позиціями.

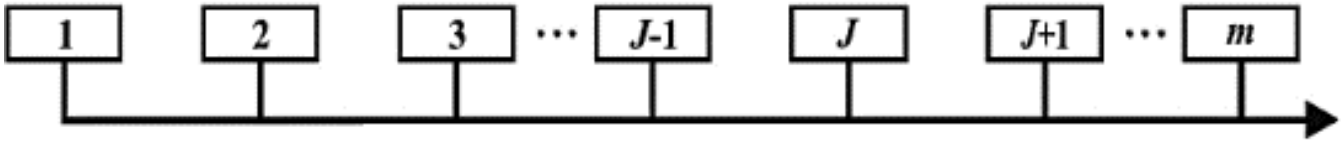

Рисунок 1.5 – Структурна схема потокової лінії з жорстким зв'язуванням між позиціями

При жорсткій структурі потоку, коли шлях руху предметів праці заздалегідь зумовлений, тому, що вони рухаються в "одній зв'язці" строго по одному й тому самому маршруту, звісно, що архіважлива роль відводиться дотриманню регламенту часу виконання ремонтних операцій на кожній позиції.

З теорії надійності добре відомо, що найменш надійною являється система та, що складається з послідовно з'єднаних елементів. Відказ будь-якого елементу в такому ланцюзі призводить до відказу всієї системи в цілому. Порушення регламентованого такту на одній з позицій, по суті і є відказом всього потоку.

1.2.3 Напівжорсткий потік

Окрім "жорсткого" потоку, котрий був прийнятий повсюдно за основу при організації вагоноремонтного виробництва, існують також й інші різновиди потокового виробництва. У якості одного з рішень, підвищуючи пропускну здатність потокової лінії й не що потребує великих капітальних вкладів, було запропоновано, наприклад, перейняти від "жорсткого" потоку до "напівжорсткого". Суть цієї пропозиції зводилась до того, щоб замість одного вантажепровідного конвеєру, виконуваного одночасну перестановку усіх вагонів між позиціями, перейти до "ланцюжку" окремих конвеєрів, кожен з котрих зв'язував би тільки дві сусідні позиції (рис 1.6). Таке рішення дозволяло впровадити деякі елементи гнучкості, хоча й не розв'язує усю проблему в цілому.

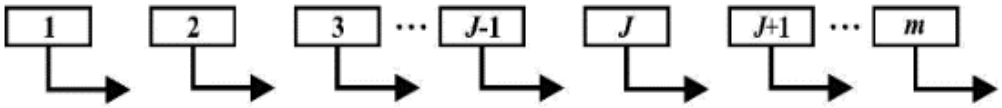

Рисунок 1.6 – Структурна схема потокової лінії з напівжорстким зв’язуванням між позиціями

1.2.4 Гнучкий варіант

Головна вагоноремонтна ділянка компактно розміщена у трьох паралельних будівельних прольотах. Два крайні прольоти – ремонтні й середній – транспортний. Перестановка вагонів між позиціями виконується за допомогою транспортного агрегату. Місцю для розташування вагонів (модулі) розташовані не вздовж прольотів будівлі, а поперек. Ремонтні прольоти розташовані з обох сторін від транспортного прольоту. Таке розташування пов’язане насамперед з тим, що трансбордер мав можливість одночасно обслуговувати ремонтні позиції, знаджувані з обох сторін від нього.

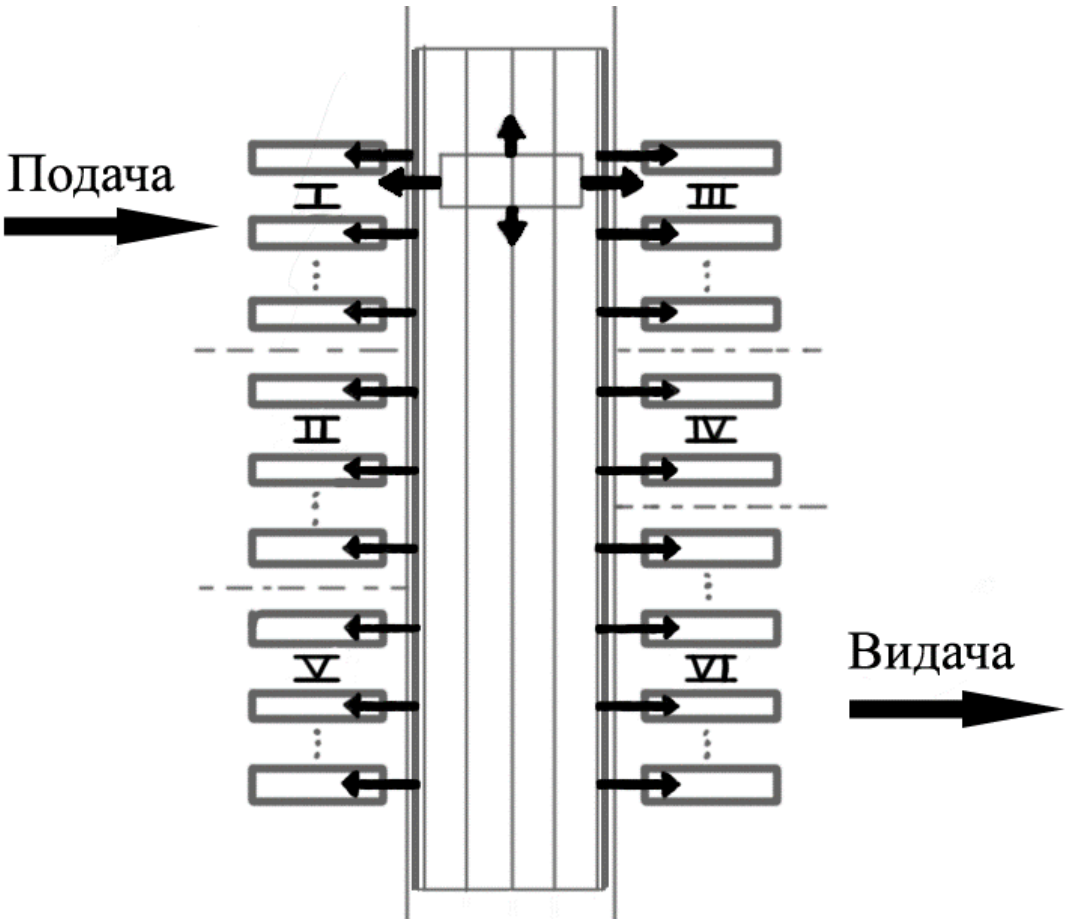

Рисунок 1.7 – Структурна схема потокової лінії з гнучкими зв’язками між позиціями

Використання окремого транспортного прольоту для переміщення вагонів між ремонтними позиціями дозволяє не тільки убезпечити працю виконавців у ремонтних

прольотах, але також й не відривати їх від виконання задач на сусідніх модулях при виконанні між позиційного переміщення. Крім того, така схема переміщення вагонів не чіпає роботу транспортних й вантажопідіймальних засобів, що безпосередньо обслуговують ремонтні позиції. Крім того, така архітектурно-транспортно-технологічна компонування зі спеціалізацією ремонтних позицій буде просто змушує підприємства використовувати виключно потоковий метод ремонту, бо при стаціонарному методі просто не можливо пройти повний ремонтний цикл.

1.3 Опитування зацікавлених сторін

Метод опитування – психологічний вербально-комунікативний метод, що полягає в здійсненні взаємодії між інтерв'юером і опитуваними (респондентами) з метою одержання від суб'єкта відповідей на заздалегідь сформульовані запитання. Іншими словами, опитування являє собою спілкування інтерв’юера і респондента, у якому головним інструментом виступає заздалегідь сформульоване питання [10].

Респондент – замовник.

1.3.1 Перелік питань

Далі представлено перелік питань, що необхідні для конкретизації задачі:

1. Який повинен бути тип додатку настільний чи веб?
2. За яким методом ремонту необхідно виконати моделювання?
3. Що необхідно відобразити на формі додатку?
4. Чи необхідна якісь додаткові можливості під час програвання анімації?
5. Як повинні відображатися об’єкти на анімації?

1.3.2 Відповіді респондента

Далі представлено перелік відповідей респондента на поставлені питання:

1. Настільний у вигляді додатку на комп’ютер;
2. За методом гнучких потокових виробництв;
3. Повинна бути можливість введення значень для моделювання та необхідно відобразити анімацію процесу ремонту;

4. Так, необхідна можливість зміни програвання анімації, а ще щоб під час виконання моделювання, на фоні, грала музика, а також можливість цю музику зупинити;
5. Вагони повинні мати різні типи та кольори, тобто вони повинні бути у вигляді зображень на формі. Також повинні бути модулі, на котрих рейки для вагонів та трансбордер, котрий буде переміщати вагони між модулями. А для трансбордеру необхідно 5 вертикальних рейок

1.4 Постановка задачі

Необхідно розробити програмний додаток з графічним інтерфейсом, що буде зображати анімацію процесу виконання ремонту відповідно до моделі описаній у розділі 1.2.4. Також необхідна можливість задавати конфігурацію цієї моделі, можливість перемикати програвання музики та змінювати швидкість програвання анімації.

Висновки до розділу 1

Під час збору вимог було оглянуто й обрано необхідну схему виконання моделювання, визначено тип додатка та затверджено все, що необхідно прийняти на вхід та зобразити на екрані.

Найбільш інформативними джерелами інформації були огляд літератури та опитування зацікавлених сторін, саме за цими методами приймалося рішення, щодо того, який додаток повинен бути та який функціонал він повинен виконувати.

2 ЗОВНІШНЄ І ВНУТРІШНЄ ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Програмний продукт, що розробляється, призначений для моделювання процесу ремонту рухомого складу на гнучких потокових виробництвах на основі введених даних. Повинен дозволяти додавати та видаляти позиції, а також вводити кількість модулів на відповідні позиції, середній час необхідний для робіт на них та нормальний відступ часу від середнього значення. Також надавати можливість задання швидкості демонстрування, та кількість вагонів, що повинні пройти через усі позиції.

2.1.2 Експлуатаційне призначення

За допомогою даного програмного продукту у викладачів з’явиться можливість наочно продемонструвати, студентам та абітурієнтам, взаємодію платформ та трансбордеру за використаною моделлю, що сприятиме кращому розумінню студентами процесів на гнучких потокових виробництвах. Викладачі зможуть, простим для розуміння чином, продемонструвати, що кількість модулів на позицію повинна змінюватися відповідно часу необхідного для робіт на них.

2.1.3 Вимоги до функціональних характеристик

Повинна бути можливість задавати:

- кількість модулів на позиції;
- середній час необхідний для виконання робіт на позиції;
- нормальний відступ від середнього часу;
- кількість вагонів, що повинні пройти через усі позиції;
- швидкість відображення анімації.

Також необхідна можливість додавати та видаляти позиції, а також спосіб зупинити/почати програвання музики.

Програмний продукт повинен зображати на формі анімацію процесу ремонту, а саме надходження вагонів до модулів, очікування на цих самих модулях (виконання ремонтних робіт) та перехід між модулями за допомогою трансбордеру. Процес

виконання ремонтних робіт повинен супроводжуватись відповідним світловим сигналом (коло червоного/зеленого кольору на платформі).

На формі необхідно відобразити кількість вагонів, що надійшло в ремонт та скільки вже відремонтовано.

Схема моделювання для кожного вагону:

1. Прибуває на першу позицію;
2. Задається випадковий час виконання робіт (відповідно заданим значенням);
3. Світловий сигнал модуля змінюється на червоний;
4. Очікується заданий час;
5. Світловий сигнал модуля змінюється на зелений;
6. Якщо трансбордер і наступна позиція вільні – перехід до наступного кроку;
7. Якщо трансбордер не знаходиться біля модуля з вагоном – переміщення його до цього модуля;
8. Переміщення вагону на трансбордер;
9. Вагон з трансбордером переміщаються до необхідного модуля;
10. Вагон переміщаються на модуль;
11. Повторити кроки 2-10 поки не будуть пройдені усі позиції;
12. Звільнення вагону з останньої позиції.

2.1.4 Вхідні дані

Вхідними даним є:

1) Position – структура, що зберігає обмеження для конфігурації моделі, значення отримуються з файлу(“config.json”):

- NUMBER – граничні значення для номеру позиції, список із 2 елементів типу даних int;
- COUNT_MODULES – граничні значення для кількості модулів на позицію, список із 2 елементів типу даних int;
- AVG_TIME – граничні значення для середнього часу робіт на позиції, список із 2 елементів типу даних float;

- STEP_TIME – граничні значення для відступу від середнього часу на позиції, список із 2 елементів типу даних float;

2) Module – структура, що структура, що зберігає розміри модуля, значення отримуються з файлу(“config.json”) – SIZE– ширина та висота, для кожного модулю, список із 2 елементів типу даних int;

3) Transborder – структура, що зберігає розміри трансбордеру, значення отримуються з файлу(“config.json”) – SIZE– ширина та висота трансбордеру, список із 2 елементів типу даних int;

4) Train – структура, що зберігає розміри вагону, значення отримуються з файлу (“config.json”) – SIZE– ширина та висота для кожного вагону, список із 2 елементів типу даних int;

5) MAX_COUNT_TRAINS – зберігає максимальну кількість вагонів, що повинні виконати ремонт, значення отримуються з файлу(“config.json”);

6) musicFile – музикальний файл, що задається неявним чином (“music.mp3” файл у директорії з виконуваним файлом) – тип даних QUrl, зберігає шлях до файлу;

7) isMusicPlaying – перемикач для програвання музики – може приймати один з 2 значень, тип даних bool;

8) speed – швидкість програвання анімації – може приймати дробове число у межах від 0 до 1, тип даних float;

9) таблиця з конфігуруванням схеми гнучкого поточного виробництва для ремонту рухомих складів, як мінімум повинно бути 2 рядки. Поля таблиці:

- type – номер позиції – ціле натуральне число відповідно обмеженням заданим у Position, тип даних int;
- count – кількість модулів на цій позиції – ціле натуральне число відповідно обмеженням заданим у Position, тип даних int;
- avg – середній час, у хвилинах, необхідний для виконання робіт на цій позиції – дробове число відповідно обмеженням заданим у Position, тип даних float;

- step – нормальний відступ від середнього часу, у хвилинах (необхідне для генерації випадкових чисел за нормальним розподілом) – дробове число відповідно обмеженням заданим у Position, тип даних float;

10) trainsCount – кількість вагонів – ціле натуральне число від 0 до MAX_COUNT_TRAINS, тип даних int.

2.1.5 Вихідні дані

Вихідними даними є:

1) музика – програється musicFile файл;

2) значення скільки надійшло у ремонт та скільки вже відремонтовано – числові значення у межах від 0 до trainsCount;

3) графічне зображення анімації на формі:

- вагони – графічні зображення, що рухаються від модуля до модуля;
- модулі – сірі прямокутники з рейками, номером та світловим сигналом зеленого чи червоного кольорів;
- трансбордер – темно-сірий прямокутник з рейками, що рухається вертикально по 5 рейках.

Виконаємо специфікацію функціональних вимог у вигляді прецедентів (рис. 2.1).

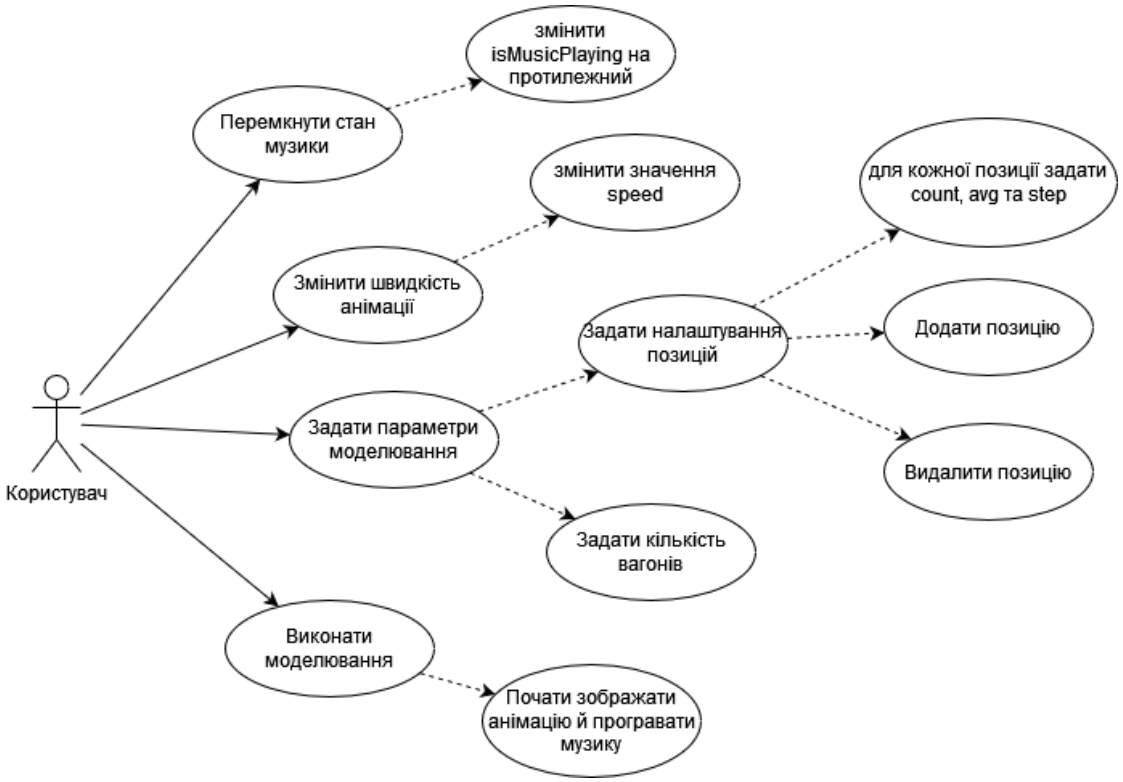

Рисунок 2.1 – Діаграма прецедентів

2.1.6 Опис зовнішнього інформаційного середовища

Музика задається у неявному вигляді, а саме заміною файлу “music.mp3” у директорії з виконуваним файлом;

Перемикання стану музики виконується за допомогою натискання лівою клавішою миші на кнопку на формі;

Швидкість програвання анімації задається зміною положення слайдеру на формі за допомогою переміщення повзунка лівою клавішою миші;

Дані type, count, avg та step вводяться з клавіатури у таблицю, за необхідності додати нову позиції слід за допомогою миші (ЛКМ) натиснути на кнопку “+” на формі, а якщо необхідно – кнопку “-“ відповідно;

Кількість вагонів вводиться з клавіатури у контрольоване поле вводу;

Результат моделювання виводиться на формі у вигляді графічної анімації.

Для функціонування ПЗ необхідно мати: встановлену операційну систему Windows 7 чи вище, у директорії з виконуваним файлом повинні бути встановлені усі необхідні бібліотеки QT, та файл “music.mp3”.

2.2 Внутрішнє проектування

2.2.1 Проектування архітектури системи

2.2.1.1 Моделювання словника системи

Ідентифіковані сутності: трансбордер, вагон, модуль, позиція, менеджер, анімація, зображення, налаштування. Додатково зазначимо інтерфейс користувача у вигляді екранної форми.

Ідентифіковані обов’язки:

1) Трансбордер – виконує задачу пересування вагонів між модулями. Зберігає інформацію про свій стан, а саме свої розміри, який вагон на ньому знаходиться, звідки й куди йому необхідно рухатись;
2) Вагон – зберігає розміри та тип вагону;
3) Модуль – збереження чи виконуються на ньому роботи, який вагон на ньому розташований, свій номер та розміри;
4) Позиція – збереження типу позиції, модулів відповідного типу, виконує задачі отримання вільного модуля та часу необхідного для виконання робіт;

5) Менеджер – управління усіма процесами відповідно моделі. Зберігає усі вагони, що повинні пройти по усім позиція та усі позиції, що були задані користувачем;
6) Анімація – змінення стану об’єктів відповідно часу(з ходом часу оновлює отриманий об’єкт);
7) Зображення – відображення усіх діючих об’єктів(трансбордер, вагони, модулі) на формі;
8) Налаштування – збереження та отримання обмежень заданих у файлі(“config.json”);
9) Форма – введення даних для моделювання(конфігурація позицій, кількість вагонів), задання швидкості відображення, перемикання стану програвання музики, запуск та перегляд процесу моделювання.

Атрибути та операції, необхідні для виконання обов'язків кожної сутності-класу наведемо у табл. 2.1.

Таблиця 2.1 – Сутності, атрибути та методи

Сутність	Атрибути	Методи
1	2	3
трансбордер	localePlace – відносна позиція куди необхідно розташувати вагон – 2 значення позиції (“x” та “y”), тип даних QPoint; train – вказівник на вагон, що знаходиться на трансбордері, тип даних Train*; from – вказівник на платформу з якої необхідно забрати вагон, тип даних Platform*; width – ширина трансбордеру – ціле додатне число, тип даних int;	render – відобразити трансбордер на формі; isFree – повертає значення, чи вільний трансбордер у момент запиту; getPlacePos – отримати абсолютну позицію, куди необхідно розташувати вагон;

Продовження таблиці 2.1

1	2	3
	height – висота трансбордеру – ціле додатне число, тип даних int;	
вагон	image – зображення вагону конкретного типу, тип даних QImage; width – ширина вагону – ціле додатне число, тип даних int; height – висота вагону – ціле додатне число, тип даних int;	render – відобразити вагон на формі;
модуль	isWorking – чи виконуються роботи на модулі – може приймати одне з 2 значень, тип даних bool; localePlace – відносна позиція куди необхідно розташувати вагон – 2 значення позиції (“x” та “y”), тип даних QPoint; train – вказівник на вагон, що знаходиться на модулі, тип даних Train*; number – номер модуля – ціле додатне число, тип даних int; width – ширина модуля – ціле додатне число, тип даних int; height – висота модуля – ціле додатне число, тип даних int;	render – відобразити модуль на формі; free – звільнити модуль; isFree – повертає значення, чи вільний модуль у момент запиту; set – задає вагон, що на ньому буде розташований; get – отримати вагон, що на ньому розташований; getPlacePos – отримати абсолютну позицію, куди необхідно розташувати вагон;

Продовження таблиці 2.1

1	2	3
позиція	type – номер типу позиції – ціле додатне число, тип даних int; avgTime – середній час виконання робіт на позиції – дробове число, тип даних float; step – нормальний відступ від середнього часу виконання робіт на позиції – дробове число, тип даних float; currentHeight – висота позиції з урахуванням усіх модулів на ній – ціле додатне число, тип даних int; platforms – масив платформ, що відповідають цій позиції, тип даних std::vector<Platform>;	render – відобразити усі модулі, що відповідають цій позиції на формі; addPlatform – додати модуль до позиції; getFreePlatform – отримати вільний модуль; getWaitingTime – отримати час виконання робіт на цій позиції;
менеджер	x_leftColumn – висота лівої колонки модулів – ціле додатне число, тип даних int; x_rightColumn – висота правої колонки модулів – ціле додатне число, тип даних int; startTP – масив вагонів у цеху, тип даних std::vector<Task*>; activeTP – масив вагонів, що виконують роботи, тип даних std::vector<Task*>;	update – оновити стан усієї моделі; render – відобразити усі об’єкти моделі на формі; addTrain – додати вагон до цеху; addPlatform – додати модуль; getCountAll – отримати кількість усіх вагонів; getCountOnStart – отримати кількість вагонів у цеху;

Продовження таблиці 2.1

1	2	3
	trains – масив усіх вагонів, тип даних std::vector<Task*>; deadPlatform – пуста платформа, що означає завершення виконання усіх задач, тип даних Platform; positions – масив позицій з довільним ключем, тип даних std::map<int, Position>; leftColumnHeight – висота лівої колонки модулів – ціле додатне число, тип даних int; rightColumnHeight – висота правої колонки модулів – ціле додатне число, тип даних int; transborder – об'єкт трансбордеру, тип даних Transborder; animations – масив анімацій, що необхідно виконати, тип даних std::vector<Animation*>;	getCountActive – отримати кількість вагонів, що виконують ремонт; addMoveAnimation – додати анімацію об'єкту для його переміщення; addWaitAnimation – додати анімацію очікування для об'єкта; updateTasks – оновити стан кожного вагону; updateTrainTask – оновити стан для конкретного вагону; tryAddMoreTrains – спробувати додати вагон із цеху;
анімація	obj – об'єкт, що анімується, тип даних VObject*; time – час, що залишився на анімацію – дробове число, тип даних float; to – позиція, куди необхідно перемістити об'єкт, тип даних QPoint;	update – оновити стан об'єкту відповідно анімації

Продовження таблиці 2.1

1	2	3
зображення	speed – швидкість програвання анімації – дробове число, тип даних float; world – вказівник на об’єкт, що необхідно зобразити, тип даних RObject*;	paintEvent – відобразити world; animate – оновити стан world;
налаштування	position – структура, що зберігає обмеження кількостей позицій, модулів на позиціях, обмеження середнього часу та часу відступу від середнього, тип даних Position; platform – структура, що зберігає розміри модуля, тип даних Platform; transborder – структура, що зберігає розміри трансбордеру, тип даних Transborder; train – структура, що зберігає розміри вагону, тип даних Train; MAX_COUNT_TRAINS – максимальна кількість вагонів – ціле додатне число, тип даних int;	parse – зчитати налаштування з файлу; set – отримати об’єкт (Config), для зміни значень; get – отримати об’єкт (Config), для отримання значень;
форма	ui – зберігає усі елементи інтерфейсу, тип даних Ui::Widget; world – об’єкт моделі, що демонструється, тип даних Manager*;	setWorld – задає модель для демонстрації; run – виконує налаштування моделі і починає виконання демонстрації;

Закінчення таблиці 2.1

1	2	3
	player – музикальний програвач, тип даних QMediaPlayer*; playlist – список пісень, необхідний для player, тип даних QMediaPlaylist*; isMusicPlaying – перемикач стану музики, тип даних bool	speedChanged – змінює швидкість анімації; addRowToTable – додає рядок до таблиці з конфігуруванням моделі; removeRowFromTable – видаляє виділений рядок з таблиці конфігурування моделі; updateCounters – оновлює значення лічильників (скільки вагонів виконують ремонт, а скільки вже відремонтовано); switchMusicStatement – перемикає стан програвання музики; setupConfiguration – застосовує конфігурації з сутності Config; generateTrains – додає до моделі вагони випадкового типу; parseTable – зчитує значення з таблиці конфігурування моделі;

В ході аналізу, сутність “анімація” була розділена на дві, а саме сутність “очікування” та “переміщення”.

2.2.1.2 Моделювання розподілу обов’язків

Множини класів, які працюють спільно для досягнення деякої поведінки:

- Форма, зображення – відображення процесу моделювання;
- Форма, налаштування – отримання обмежень на значення, що вводяться;
- Форма, менеджер – налаштування моделі, що повинна відобразитися;
- Анімація, трансбордер – зміна положення трансбордеру;
- Анімація, вагон – зміна положення чи оновлення часу очікування для вагону;
- Трансбордер, налаштування – отримання розмірів трансбордеру;
- Трансбордер, вагон – переміщення вагону від одного модуля до іншої;
- Вагон, налаштування – отримання розмірів вагону;
- Модуль, налаштування – отримання розмірів модуля;
- Менеджер, анімація – створює і викликає оновлення стану анімації;
- Менеджер, трансбордер – встановлення вагону на трансбордер та задання поведінки;
- Менеджер, вагон – створення та додання вагонів до моделі;
- Менеджер, позиція – створення та додавання модулів до позицій;
- Позиція, модуль – створення модулів;

2.2.1.3 Моделювання непрограмних сутностей

Ідентифіковані непрограмні сутності:

1) RObject – абстрактний інтерфейс, що призначений для зображення об’єктів на формі;

2) VObject – абстрактний інтерфейс, що призначений для зображення рухомих об’єктів на формі.

2.2.1.4 Моделювання примітивних типів, простих залежностей, наслідування та структурних зв’язків

Результат моделювання різних видів зв’язків подано у табл. 2.2.

Таблиця 2.2 – Моделювання залежностей

Клас, який зв’язується	Клас, з яким зв’язуються	Тип зв’язку
Animation	VObject	Агрегація
AMove	Animation	Узагальнення
Manager	RObject	Реалізація
	Config	Залежність
	Train	Композиція
	Position	Композиція
	Animation	Композиція
	Transborder	Композиція
Platform	VObject	Реалізація
	Config	Залежність
	Train	Агрегація
Position	VObject	Реалізація
	Config	Залежність
	Platform	Композиція
VObject	RObject	Узагальнення
Train	VObject	Реалізація
	Config	Залежність
Transborder	VObject	Реалізація
	Config	Залежність
	Train	Агрегація
	Platform	Агрегація
GLWidget	RObject	Асоціація
Widget	GLWidget	Асоціація
	Manager	Композиція
	Config	Залежність

У таблицях 2.3-2.14 зображені CRC картки для усіх класів.

Таблиця 2.3 – CRC-картка для класу “Config”

Базовий клас	Похідні класи
Відсутній	Відсутні
Обов’язки	Зв’язки
Збереження та отримання обмежень заданих у файлі(“config.json”)	Manager, Platform, Position, Train, Transborder, Widget

Таблиця 2.4 – CRC-картка для класу “RObject”

Базовий клас	Похідні класи
Відсутній	VObject, Manager, GLWidget
Обов’язки	Зв’язки
Інтерфейс об’єктів, що можуть оновлювати свій стан та зображатися на формі	VObject, Manager, GLWidget, Animation, AMove, Platform, Position, Train, Transborder

Таблиця 2.5 – CRC-картка для класу “VObject”

Базовий клас	Похідні класи
RObject	Animation, Platform, Position, Train, Transborder
Обов’язки	Зв’язки
Інтерфейс об’єктів, що можуть оновлювати свій стан та зображатися на формі, а також мають свою позицію на екрані	RObject, Animation, AMove, Platform, Position, Train, Transborder

Таблиця 2.6 – CRC-картка для класу “Animation”

Базовий клас	Похідні класи
Відсутній	AMove
Обов’язки	Зв’язки
Оновлення часу виконання анімації	AMove, VObject, Manager

Таблиця 2.7 – CRC-картка для класу “AMove”

Базовий клас	Похідні класи
Animation	відсутні
Обов’язки	Зв’язки
Оновлення позиції об’єкту	Animation, VObject, Manager

Таблиця 2.8 – CRC-картка для класу “Manager”

Базовий клас	Похідні класи
RObject	Відсутні
Обов’язки	Зв’язки
Управління моделлю. Утримання усіх об’єктів та створення й оновлення анімацій	RObject, Config, Train, Position, Transborder, Animation, AMove

Таблиця 2.9 – CRC-картка для класу “Platform”

Базовий клас	Похідні класи
VObject	Відсутні
Обов’язки	Зв’язки
Збереження чи виконуються на ньому роботи, який вагон на ньому розташований, свій номер та розміри	VObject, Config, Train

Таблиця 2.10 – CRC-картка для класу “Position”

Базовий клас	Похідні класи
VObject	Відсутні
Обов’язки	Зв’язки
Збереження типу позиції, модулів відповідного типу, виконує задачі отримання вільного модуля та часу необхідного для виконання робіт	VObject, Config, Platform

Таблиця 2.11 – CRC-картка для класу “Train”

Базовий клас	Похідні класи
VObject	Відсутні
Обов’язки	Зв’язки
Зберігає розміри та тип вагону	VObject, Config

Таблиця 2.12 – CRC-картка для класу “Transborder”

Базовий клас	Похідні класи
VObject	Відсутні
Обов’язки	Зв’язки
Виконує задачу пересування вагонів між модулями. Зберігає інформацію про свій стан, а саме свої розміри, який вагон на ньому знаходиться, звідки й куди йому необхідно рухатись	VObject, Config, Train, Platform

Таблиця 2.13 – CRC-картка для класу “GLWidget”

Базовий клас	Похідні класи
Відсутній	Відсутні
Обов’язки	Зв’язки
Відображення усіх діючих об’єктів(трансбордер, вагони, модулі) на формі	RObject

Таблиця 2.14 – CRC-картка для класу “Widget”

Базовий клас	Похідні класи
RObject	Відсутні
Обов’язки	Зв’язки
Введення даних для моделювання(конфігурація позицій, кількість вагонів), задання швидкості відображення, перемикання стану програвання музики, запуск та перегляд процесу моделювання	Config, GLWidget, Manager

Заключний результат моделювання у вигляді діаграми класів (рис. 2.2).

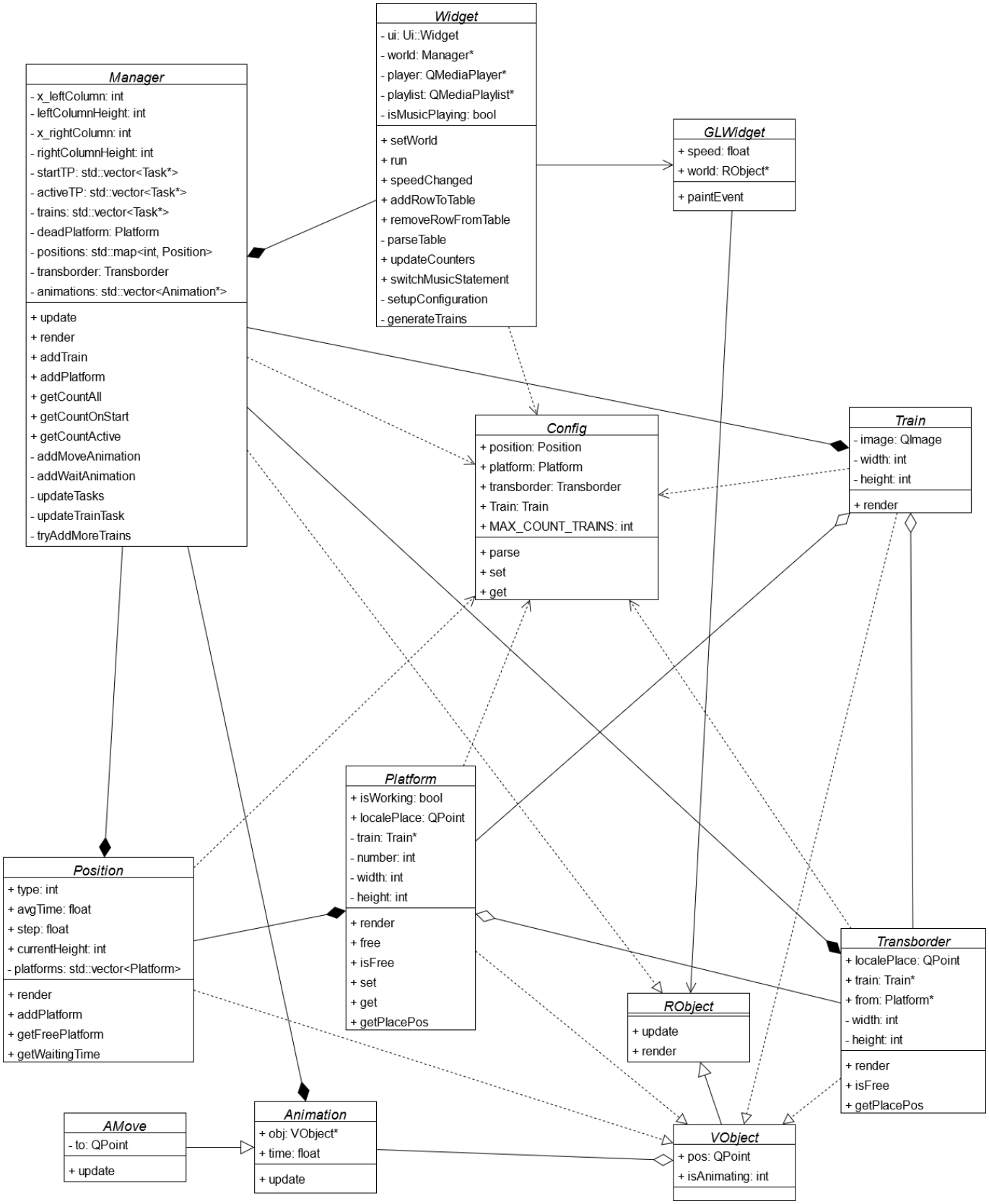

Рисунок 2.2 – Діаграма класів

2.2.2 Проектування інтерфейсу користувача

На формі необхідно відобразити:

- зображення;
- таблицю для введення конфігурації моделі;
- дві кнопки для додавання/видалення рядку таблиці;
- поле для введення кількості вагонів;
- слайдер для зміни швидкості програвання;
- кнопку для початку/зупинки програвання музики;
- кнопку для початку процесу моделювання;
- два поля для відображення кількості вагонів, що надійшло/завершило процес ремонтування.

Визначимо взаємне розташування даних та елементів інтерфейсу на екрані (рис. 2.3). На формі основна увага приділена саме зображенню, де буде демонструватися процес моделювання. Після розробки ескізу був створений макет (рис. 2.4), що демонструє розміщення елементів інтерфейсу.

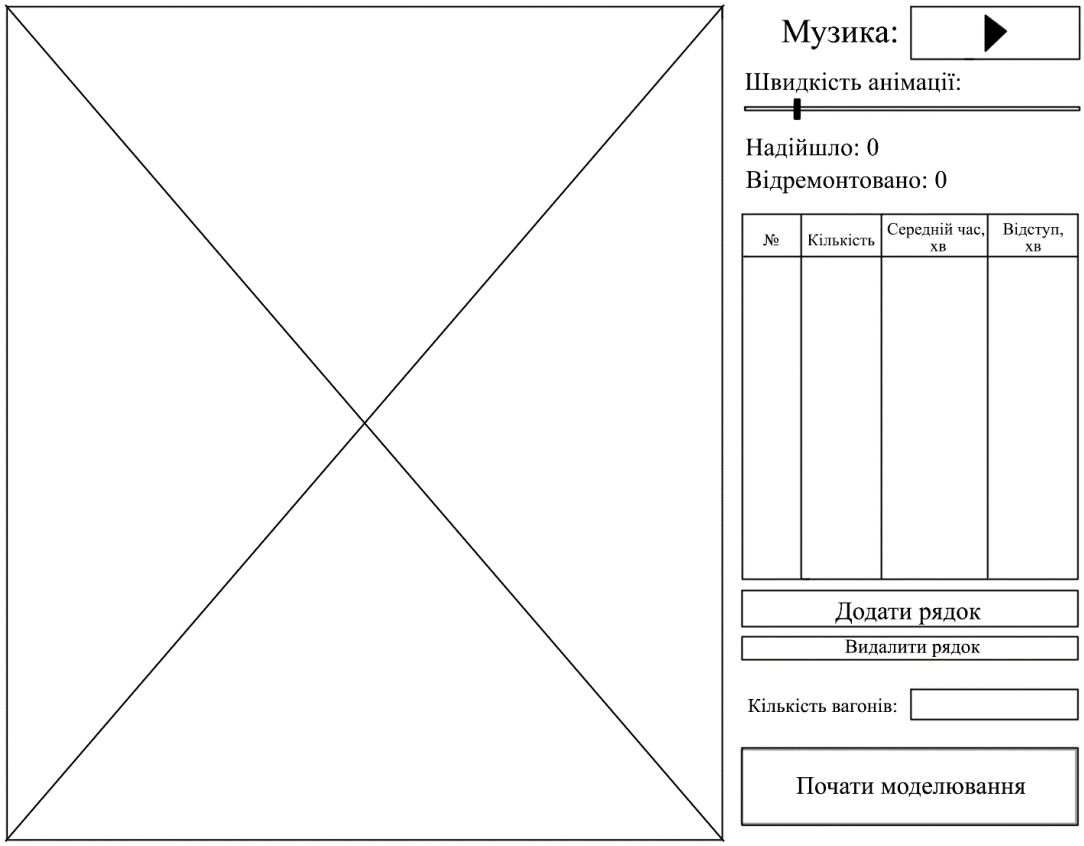

Рисунок 2.3 – Ескіз форми

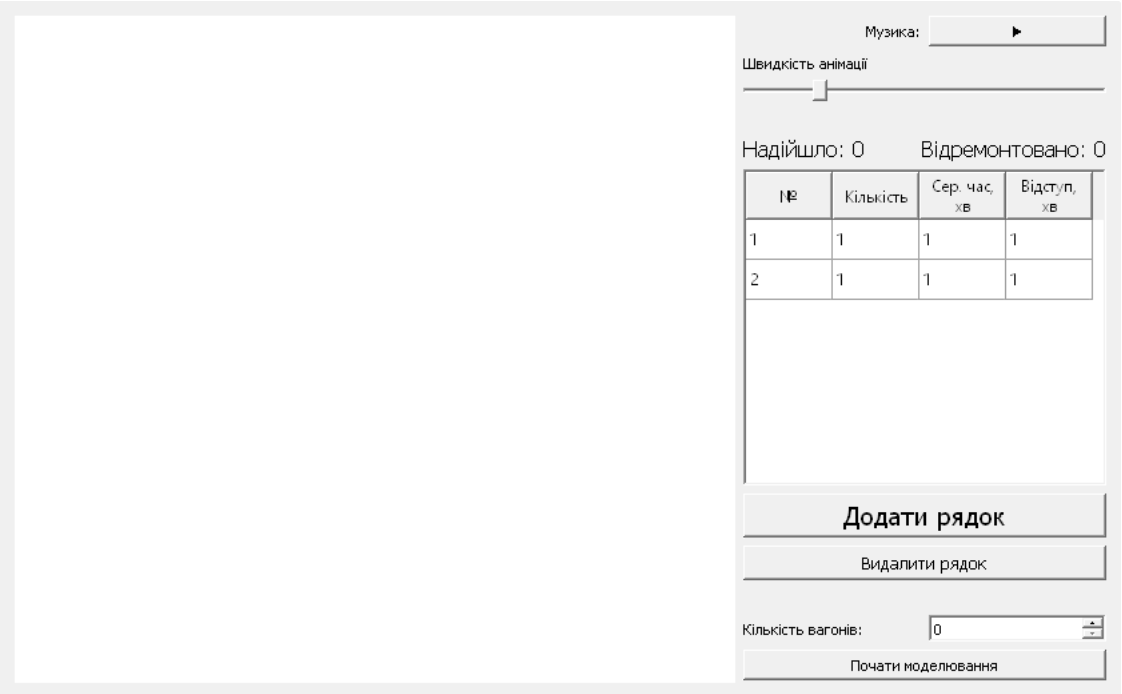

Рисунок 2.4 – Макет розміщення даних

При спробі видалити рядок не виділивши його за допомогою миші, повинне з’явитися вікно, що повідомить користувача, як необхідно видаляти рядки.

2.2.3 Проектування динаміки системи

На рис. 2.5 зображена діаграма послідовностей для ініціалізації найпростішої моделі й виконання переміщення вагона до першого модуля. Методи “render” та “update” викликаються постійно й асинхронно, тому на діаграмі були зображені тільки декілька випадків коли вони викликались. Для реалізації переміщення вагона між модулями будуть виконані аналогічні операції, але додається створення анімацій руху вагона на трансбордер, руху вагона й трансбордеру до необхідного модуля та вивантаження вагона до цього модуля.

Висновки до розділу 2

Під час зовнішнього проектування було визначено, що повинне робити ПЗ, а саме, воно повинне приймати на вхід конфігураційні значення та повертати результат моделювання.

А під час внутрішнього проектування – розроблено архітектуру системи, визначено сутності проекту, інтерфейс користувача та як ці сутності повинні взаємодіяти між собою.

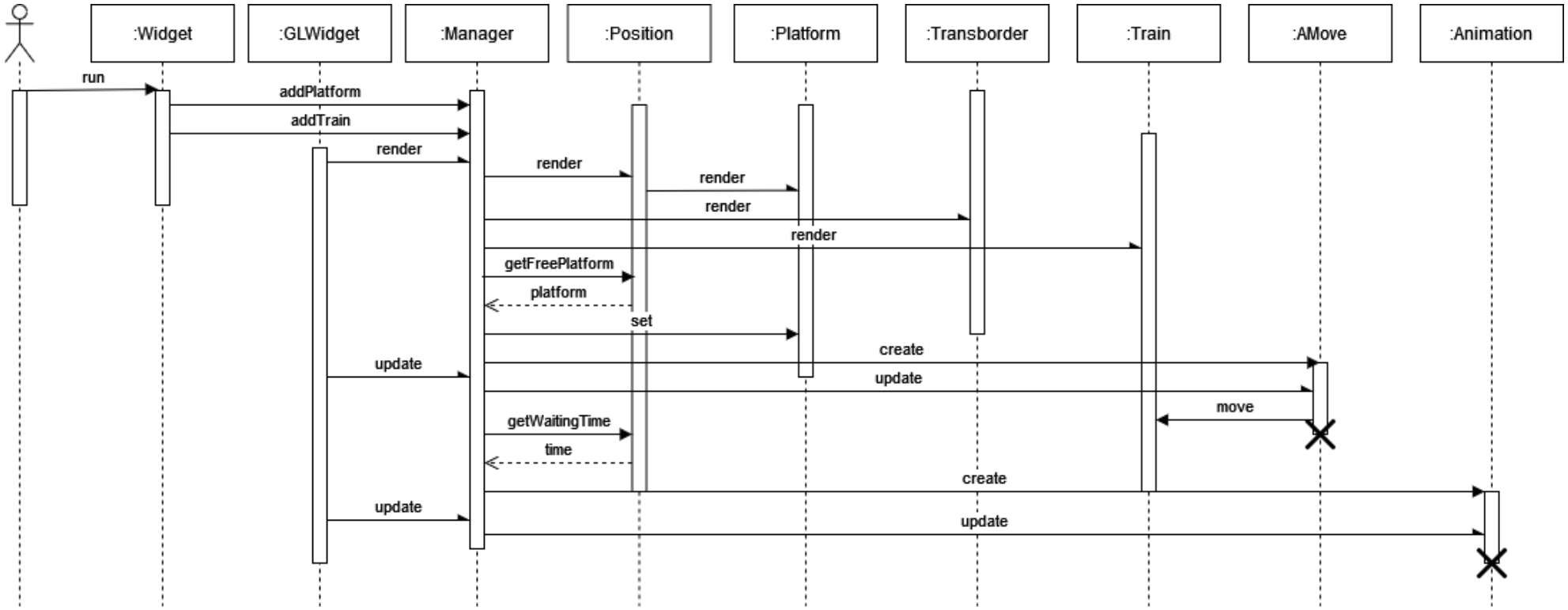

Рисунок 2.5 – Діаграма послідовностей для першого модуля

3 РОЗРОБКА ПРОГРАМИ

3.1 Вибір мови програмування

Для реалізації розроблюваного продукту було обрано мову програмування C++ з використанням фреймворку Qt. Оскільки стоїть задача моделювання, тому найважливішою є задача стабільності й швидкості виконання, а саме у цьому є перевага обраної мови. Також досить важливим фактором вибору мови є те, що під час навчання я вже використовував цю мову при вивчанні комп’ютерної графіки, тобто вже маю досвід у розв'язання подібних задач.

Для створення вікна й реалізації інтерфейсу був обраний фреймворк Qt, через те, що його я також вже використовував і він дозволяє створити кросплатформовий додаток з усіма, необхідними для розв'язання поставленої задачі, елементами інтерфейсу.

3.2 Розробка алгоритмів

Основним є алгоритмом, що повинен ініціювати виконання усіх робіт, відповідно моделі, а для спрощення цього алгоритму й уникнення дублювання коду – необхідні й другорядні алгоритми, а саме, котрі виконують одну конкретну задачу, таку як, постачання вагонів до першого модуля, чи оброблення переходів між модулями.

Далі описані необхідні алгоритми:

1) спробувати додати вагони (виконати першу задачу) рис. 3.1. Якщо усі модулі у позиції зайняті – залишити без змін. При виконанні алгоритму заповнюються усі модулі на першій позиції й вагони додаються до списку активних.

2) оновити задачі вагона (рис. 3.2). Під час виконання цього алгоритму конкретний вагон проходить усі етапи виконання задач, починаючи з першої позиції й закінчуючи останньою задачею.

3) оновлення задач (рис 3.3). Цей алгоритм пов’язує два минулих і звільнює вагони зі списку активних.

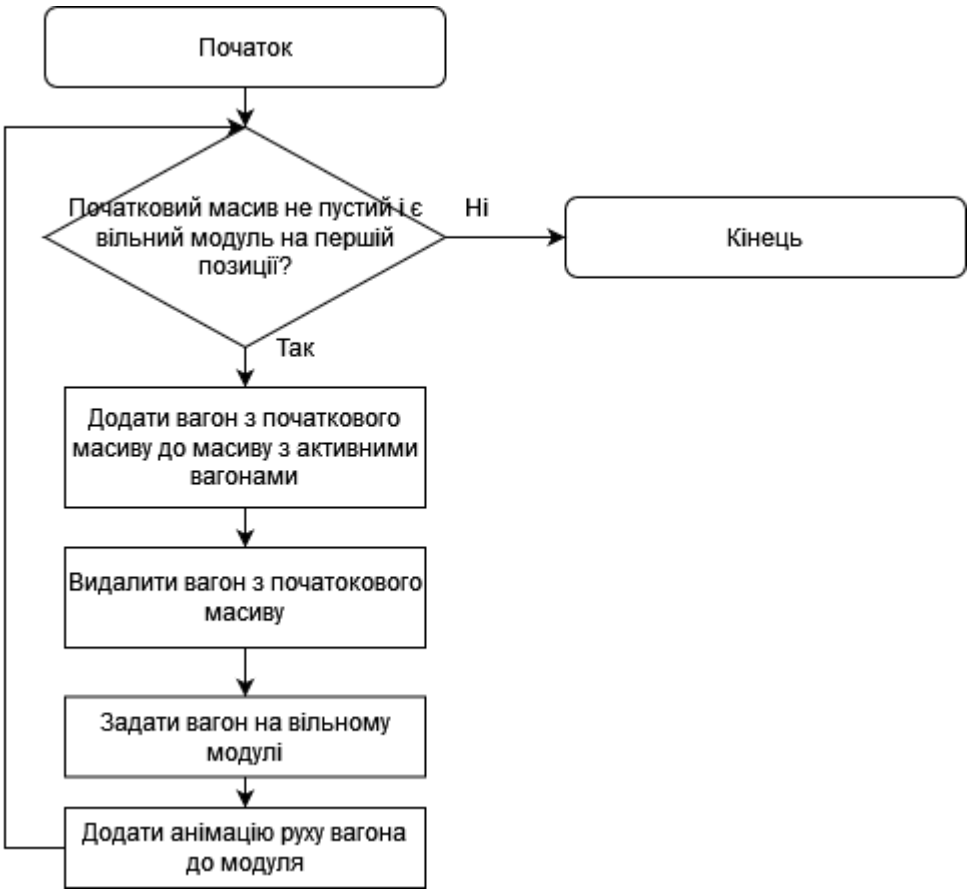

Рисунок 3.1 – Додавання вагону до активного списку

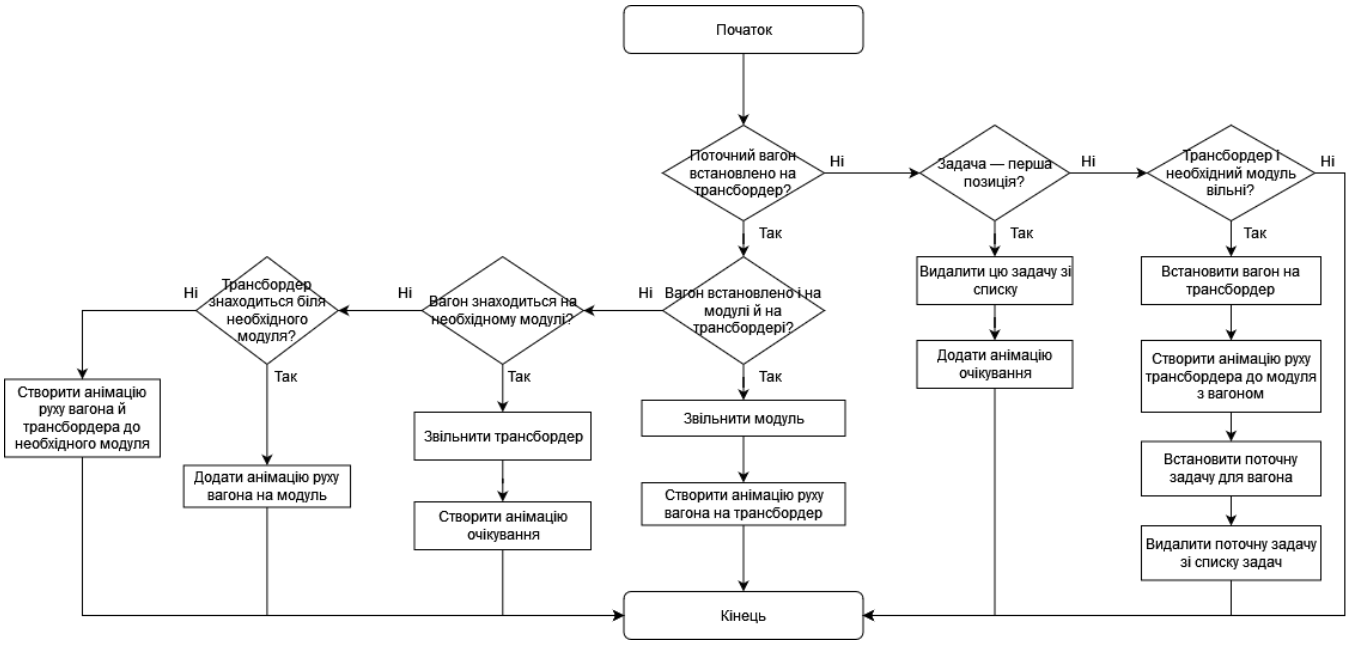

Рисунок 3.2 – Оновлення задачі конкретного вагона

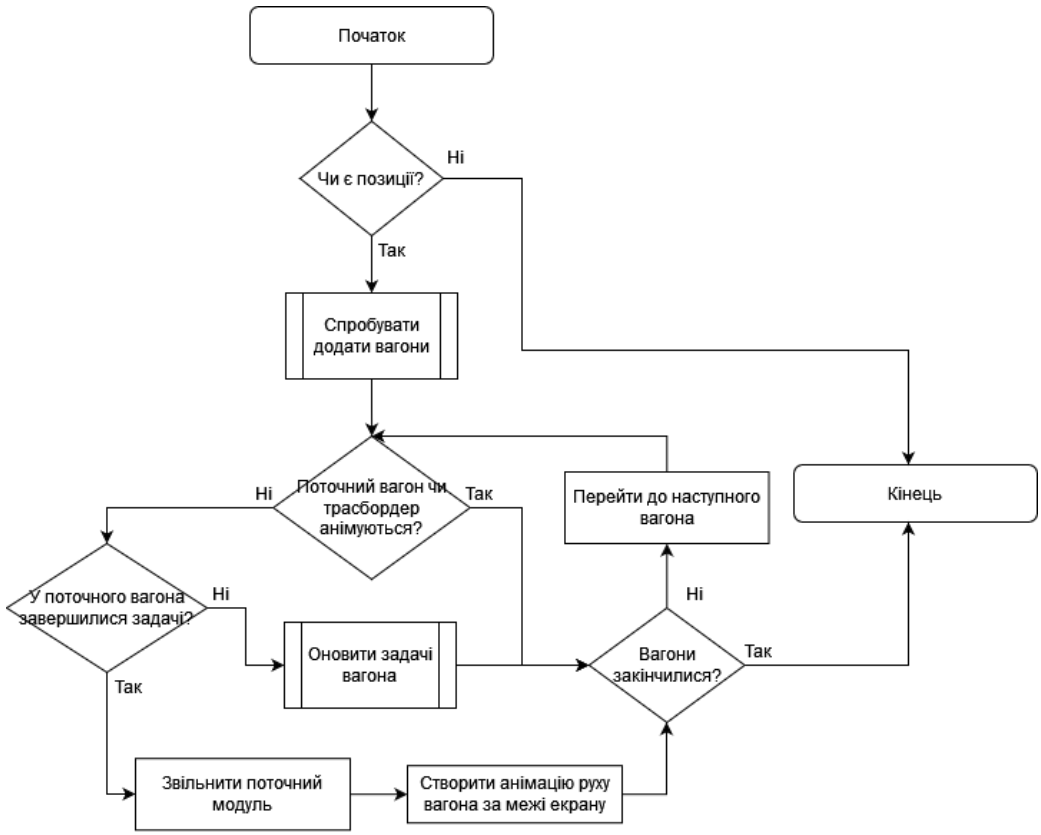

Рисунок 3.3 – Алгоритм менеджеру

Висновки до розділу 3

Було обрано мову програмування на котрій буде розроблюватись проект, а саме C++ й необхідну бібліотеку для розробки інтерфейсу – QT, також було розроблено й використано три алгоритми, що виконують поставлену задачу.

Оскільки використовується готова бібліотека для розробки інтерфейсів – є можливість приділити більше уваги до розробки саме логіки роботи ПЗ. Необхідно розмістити необхідні елементи на формі, а далі змінювати стан цих елементів.

Усі інші елементи, що використовуються у створених алгоритмах – лише зберігають інформацію.

4 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Щоб впевнитись в коректності специфікації функцій їх необхідно протестувати. Для тестування було обрано найважливіші для проекту функції, а саме ті, що виконують саме процес направлення вагонів до необхідних модулів, тобто керують усім процесом. Через те, що ці функції не планується використовувати у якості бібліотек для інших проектів – тестування чорної скринькою не є необхідним, бо інтерфейс користувача перевіряє усі вхідні значення на коректність.

4.1 Специфікація функцій

Далі наведено перелік функцій, що будуть тестуватись:

1) void Manager::updateTrainTask(Task *train, const int task)

Оновлює стан заданого вагону.

Приймає на вхід:

у явному вигляді:

- вказівник на структуру даних з вказівником на вагон, запланованим списком задач, номером поточної задачі та вказівником на поточну платформу, тип даних – Task*;
- номер позиції, на котрій необхідно виконати роботи, тип даних – int;

у неявному вигляді:

- вказівник на трансбордер, тип даних – Transborder;
- список позицій, тип даних – std::map<int, Position>;

Повертає на вихід:

у явному вигляді не повертає значень;

у неявному вигляді:

- додання анімацій до списку;
- змінює значення для платформ, а саме вагон, що знаходиться на ній;
- змінює значення для трансбордеру, а саме вагон, що знаходиться на ньому та платформи з якої необхідно отримати вагон;
- оновлює список завдань, поточну задачу та поточний модуль в отриманій структурі;

Текст функції:

```
void Manager::updateTrainTask(Task *train, const int task)
{
    const auto& trainOnTransborder = transborder.train;
    // 1
```

```
    if (&train->train == trainOnTransborder && train->currentPlatform != nullptr) {
        // 2
        if (transborder.from != nullptr && &train->train == transborder.from->get())
{
            transborder.from->free();
            transborder.from = nullptr;
            addMoveAnimation(&train->train, transborder.getPlacePos());
            // 3
        } else if (&train->train == train->currentPlatform->get()) {
            transborder.train = nullptr;
            // время ожидания на платформе(кроме первой)
            addWaitAnimation(&train->train, train->currentTask);
            // isWorking true
            train->currentPlatform->isWorking = true;
            // 4
            if (task == -1)
                train->tasks.push_back(-2);
        } else {
            auto transborderPos = QPoint(transborder.pos.x(), train->currentPlatform-
>pos.y());
            // 5
            if (transborder.pos == transborderPos) {
                train->currentPlatform->set(&train->train);
                addMoveAnimation(&train->train, train->currentPlatform-
>getPlacePos());
            } else {
                addMoveAnimation(transborder.train,
QPoint(transborderPos.x()+transborder.localePlace.x(),

transborderPos.y()+transborder.localePlace.y()));
                addMoveAnimation(&transborder, transborderPos);
            }
        }
        // 6
    } else if (task == Config::get().position.MIN_NUMBER) {
        train->tasks.pop_back();
        addWaitAnimation(&train->train, task);
        // isWorking true
        train->currentPlatform->isWorking = true;
        // 7
    } else if (transborder.isFree() && train->currentPlatform != nullptr) {
        auto platformToGo = positions[task].getFreePlatform();
        // 8
        if (platformToGo != nullptr)
        {
            transborder.train = &train->train;
            transborder.from = train->currentPlatform;
            addMoveAnimation(&transborder, QPoint(transborder.pos.x(), train-
>currentPlatform->pos.y()));
            train->currentTask = task;
            train->currentPlatform = platformToGo;
            train->tasks.pop_back();
        }
    }
}
```

2) void Manager::updateTasks()

Оновлює стан усіх вагонів.

Приймає на вхід:

– у явному вигляді не приймає значень;

– у неявному вигляді:

- список вагонів у цеху, тип даних – std::vector<task*>;
- список активних вагонів, тип даних – std::vector<task*>;
- вказівник на трансбордер, тип даних – transborder;
- список позицій, тип даних – std::map<int, Position>;

Повертає на вихід:

– у явному вигляді не повертає значень;

– у неявному вигляді:

- зміна списку вагонів у цеху
- зміна список активних вагонів
- додання анімацій до списку;
- змінює значення для платформ, а саме вагон, що знаходиться на ній;
- змінює значення для трансбордеру, а саме вагон, що знаходиться на ньому та платформи з якої необхідно отримати вагон;
- оновлює список завдань, поточну задачу та поточний модуль в отриманій структурі;

Текст функції:

```
void Manager::updateTasks()
{
    // 1
    if (positions.empty())
        return;

    tryAddMoreTrains();

    // освободить уже отработавшие вагоны
    activeTP.erase(std::remove_if(activeTP.begin(), activeTP.end(),
                        [&](Task *t) {
                            return t->currentPlatform == &deadPlatform && t-
>train.isAnimating == 0;
                        }), activeTP.end());

    // 2
    for (auto& train: activeTP) {
        int task = train->tasks.empty() ? -1 : train->tasks.back();
        // если анимируется либо тележка, либо данный вагон
        // 3
        if (train->train.isAnimating != 0 || transborder.isAnimating != 0) {
            continue;
            // 4
```

```
        } else if (task == -2)
            // если вагон уже завершил задачи и не анимируется
        {
            // isWorking false
            train->currentPlatform->isWorking = false;
            train->currentPlatform->free();
            train->currentPlatform = &deadPlatform;
            const auto widthScene =
Config::get().platform.WIDTH*2+Config::get().transborder.WIDTH+Config::get().train.WI
DTH;
            addMoveAnimation(&train->train, QPoint(widthScene, train-
>train.pos.y()));
        } else {
            // isWorking false
            train->currentPlatform->isWorking = false;
            updateTrainTask(train, task);
        }
    }
}
```

4.2 Тестування методом білої скриньки

4.2.1 Розроблення тестів

Функція 1(updateTrainTask):

1) WorkOnFirstPosition

   Вхідні дані:

   Train – {currentTask = -1, tasks = [2,1], currentPlatform = Вказівник на модуль з першої позиці};

   Task – 1;

   Transborder – не важливо у якому стані (не використовується у цьому випадку);

   Positions – на першій позиції знаходиться отриманий вагон;

   Вихідні дані:

   Train – {currentTask = -1, tasks = [2], currentPlatform = Вказівник на модуль з першої позиці};

   Animations – додано анімацію очікування виконання робіт для вагону;

2) MoveTransborderToPlatformWithTrain

   Вхідні дані:

   Train – {currentTask = -1, tasks = [2], currentPlatform = Вказівник на перший модуль};

   Task – 2;

   Transborder – {Train – відсутній, from – відсутній};

   Positions – на першій позиції знаходиться отриманий вагон;

Вихідні дані:

Train – {currentTask = 2, tasks = [], currentPlatform = Вказівник на модуль з другої позиці};

Transborder – {Train – вказівник на отриманий вагон, from – вказівник на платформу з цим вагоном};

Animations – додано анімацію руху трансбордеру до модуля з вагоном;

3) MoveTrainOnTransborder

Вхідні дані:

Train – {currentTask = 2, tasks = [], currentPlatform = Вказівник на модуль з другої позиці};

Task – -1;

Transborder – {Train – вказівник на вагон, from – вказівник на платформу з цим вагоном};

Positions – на першій позиції, першому модулі знаходиться вагон;

Вихідні дані:

Positions - на першій позиції, першому модулі відсутній вагон;

Animations – додано анімацію руху вагона на трансбордер;

Transborder – {Train – вказівник на вагон, from – відсутній};

4) Idle

Вхідні дані:

Train – {currentTask = -1, tasks = [2], currentPlatform = Вказівник на модуль з першої позиці};

Task – 2;

Transborder – {Train – вказівник інший вагон, from – відсутній};

Positions – на першій позиції, першому модулі знаходиться вагон;

Вихідні дані – все залишилось без змін;

5) MoveTrainOnPlatform

Вхідні дані:

Train – {currentTask = 2, tasks = [], currentPlatform = Вказівник на модуль з другої позиці};

Task – -1;

Transborder – {Train – вказівник на отриманий вагон, from – відсутній};

Positions – на першій позиції, першому модулі знаходиться вагон;

Вихідні дані:

Positions - на другій позиції, першому модулі встановлено вагон;

Animations – додано анімацію руху вагона на необхідний модуль;

6) WorkOnOtherPlatforms

Вхідні дані:

Train – {currentTask = 2, tasks = [], currentPlatform = Вказівник на модуль з другої позиці};

Task – -1;

Transborder – {Train – вказівник на отриманий вагон, from – відсутній};

Positions – на першій та другій позиціях, першому модулі знаходяться вагони;

Вихідні дані:

Train – {currentTask = 2, tasks = [-2], currentPlatform = Вказівник на модуль з другої позиці};

Transborder – {Train – відсутній, from – відсутній};

Positions - на першій та другій позиціях, першому модулі знаходяться вагони;

Animations – додано анімацію очікування виконання робіт для вагону;

7) MoveTrainAndTransborder

Вхідні дані:

Train – {currentTask = 3, tasks = [], currentPlatform = Вказівник на модуль з третьої позиці};

Task – -1;

Transborder – {Train – вказівник на отриманий вагон, from – відсутній};

Positions – на першій позиції, першому модулі знаходиться вагон;

Вихідні дані:

Positions - на першій та другій позиціях, першому модулі знаходяться вагони;

Animations – додано анімації руху вагона і трансбордера до необхідного модуля;

8) WorkOnMidleTask

Вхідні дані:

Train – {currentTask = 2, tasks = [3], currentPlatform = Вказівник на модуль з другої позиці};

Task – 3;

Transborder – {Train – вказівник на отриманий вагон, from – відсутній};

Positions – на другій та третій позиціях, першому модулі знаходяться вагони;

Вихідні дані:

Train – {currentTask = 2, tasks = [3], currentPlatform = Вказівник на модуль з другої позиці};

Transborder – {Train – відсутній, from – відсутній};

Positions - на другій та третій позиціях, першому модулі знаходяться вагони;

Animations – додано анімацію очікування виконання робіт для вагону;

9) WaitingForNextTask

Вхідні дані:

Train – {currentTask = 2, tasks = [3], currentPlatform = Вказівник на модуль з другої позиці};

Task – 3;

Transborder – {Train – відсутній, from – відсутній};

Positions – на другій та третій позиціях, першому модулі знаходяться вагони;

Вихідні дані – все залишилось без змін;

Функція 2(updateTasks):

1) CanNotDoAnything

Вхідні дані:

Positions – не має елементів;

startTP – не важливо;

activeTP – не важливо;

Transborder – не важливо;

Вихідні дані – все залишилось без змін;

2) ManyTasks

Вхідні дані:

Positions – є три позиції з 1 модулем у кожному;

activeTP – має три вагони:

1) Вагон, що анімується(переміщається з трансбордеру на другий модуль);
2) Вагон, що завершив роботи на останньому модулі;
3) Вагон, що починає виконання на першому модулі;

startTP – відсутні вагони;

Transborder – {Train – перший вагон у списку activeTP, from – відсутній};

Вихідні дані :

activeTP – має три вагони:

1) Залишився без змін;
2) Звільнений модуль, додано анімацію руху цього вагону за межі екрану;
3) Оновлений список задач, додано анімацію очікування для виконання робіт;

3) IsAnimating

Вхідні дані:

Positions – є три позиції з 1 модулем у кожному;

activeTP – має два вагони:

- що рухається з трансбордером до третьої позиції;
- що завершив виконання робіт на першій позиції;

startTP – відсутні вагони;

Transborder – {Train – перший вагон у списку activeTP, from – відсутній};

Вихідні дані - все залишилось без змін;

4.2.2 Покриття рішень

Тестування методом покриття рішень для функції 1 (updateTrainTask) представлено у табл. 4.1.

Тестування методом покриття рішень для функції 2 (updateTasks) представлено у табл. 4.2.

Таблиця 4.1 – Покриття рішень для функції 1

Номер тесту	Номер рішення															
	1		2		3		4		5		6		7		8	
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
1		*									*					
2		*										*	*		*	
3	*		*													
4		*										*		*		
5	*			*		*			*							
6	*			*	*		*									
7	*			*						*						
8	*			*	*			*								
9		*										*	*			*

Таблиця 4.2 – Покриття рішень для функції 2

Номер тесту	Номер рішення							
	1		2		3		4	
	+	-	+	-	+	-	+	-
1	*							
2		*	*	*	*	*	*	*

4.2.3 Покриття умов

Тестування методом покриття умов для функції 1 (updateTrainTask) представлено у табл. 4.3. Номер умови береться з тексту функції, якщо рішення складається з декількох умов – кожна буде позначатись у вигляді “<номер рішення>.<номер умови>”.

Таблиця 4.3 – Покриття умов для функції 1

Номер тесту	Номер умови																					
	1.1		1.2		2.1		2.2		3		4		5		6		7.1		7.2		8	
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
1		*													*							
2		*														*	*		*		*	
3	*		*		*		*															
4		*														*		*				
5	*		*			*				*			*									
6	*		*			*			*		*											
7	*		*			*								*								
8	*		*			*			*			*										
9		*														*	*		*			*

Тестування методом покриття умов для функції 2 (updateTasks) представлено у табл. 4.4. Номер умови береться з тексту функції, якщо рішення складається з декількох умов – кожна буде позначатись у вигляді “<номер рішення>.<номер умови>”.

Таблиця 4.4 – Покриття умов для функції 2

Номер тесту	Номер умови									
	1		2		3.1		3.2		4	
	+	-	+	-	+	-	+	-	+	-
1	*									
2		*	*	*	*	*	-	*	*	*
3		*	*	*	*	*	*			

4.3 Налагодження ПЗ

Існують такі види помилок, як:

- синтаксичні – виникають через порушення правил мови програмування. Такі помилки зазвичай виявляються під час компіляції. Можуть бути виключені порівняно легко. Навіть якщо не переглядати текст програми, то можна бути впевненим, що компілятор на стадії трансляції знайде помилки й видають відповідні попередження [19];
- семантичні (логічні) – ті, що призводять до некоректних обчислень або помилок під час виконання (run-time error). Семантичні помилки усувають зазвичай за допомогою виконання програми з ретельно підібраними перевірковими даними, для яких відома правильна відповідь [19].

При розробці ПЗ було виявлено й усунені обидва види помилок.

Синтаксичні були виявлені за допомогою компілятора, він одразу показує усі синтаксичні помилки, а метод їх усунення полягає лише у виправленні тексту команд.

А з семантичними помилками не все так просто, їх виявлення зображується у некоректній (не запланованій) поведінці ПЗ, після чого необхідно локалізувати такі помилки за допомогою точок зупинки чи логуванні ходу виконання програми, коли помилки були локалізовані – необхідно їх усунути, а це вже залежить будови алгоритму.

Висновки до розділу 4

Під час тестування білою скринькою була виявлена зайва умова у першій функції, а саме 2.2, також є умови 1.2 та 7.2, що виконуються завжди, бо якщо ця умова не буде виконана – додаток може завершиться з критичною помилкою, тому задля гарантії стабільності роботи, було вирішено їх залишити. А друга функція залишилась без змін.

Тестування методом чорної скриньки – не є необхідним, бо ПЗ не планується використовувати у якості бібліотеки, а усі можливі значення, що задає користувач, обмежуються елементами графічного інтерфейсу.

Саме для цього проекту, найбільш ефективним методом тестування виявився “Метод покриття умов”, він дозволив більш детально переглянути які умови насправді необхідні саме для виконання задачі, а які – для безпеки при виконанні..

ВИСНОВКИ

В результаті виконання даної роботи було розроблено програму, що моделює процес ремонту рухомого складу на гнучких потокових.

Під час розробки даного проекта був використаний архітектурний шаблон «Модель-Вигляд-Контролер», завдяки цьому, у майбутньому, буде легше удосконалювати програму, внаслідок того, що будь-яку частину цієї архітектури можна модифікувати, якщо буде необхідність, можливо швидко, не роблячи багато змін, зобразити моделювання в іншому інтерфейсі. Також, якщо необхідно змінити модель (наприклад, з жорстким чи напівжорстким зчепленням), достатньо замінити лише один клас (Manager) та інтерфейс відповідно.

Конфігураційний файл, що знаходиться у директорії з виконуваним файлом, дозволяє викладачам змінювати обмеження значень для необхідних для конфігурації моделі та змінювати розміри об’єктів для візуалізації.

Шляхом до подальшого удосконалення проекта є полегшення процесу додання нових типів та зображень вагонів, для цього необхідно розробити систему зберігання типів вагонів та їх зображень, а також зчитування цих значень при запуску програмного додатка. Також можливим шляхом є додання налаштувань для генерації випадкових вагонів за типами, тобто, щоб можливо було задати план робіт зі значеннями скільки яких вагонів повинно пройти ремонт.

Завдяки курсу по комп’ютерній графіці, виконання цього проекта було легше, бо вже було вивчено, як подібні проекти необхідно розроблювати, які сутності необхідні та як їх пов’язувати. Також завдяки курсу з тестування, навіть під час розробки, були знання як перевіряти себе, щоб через введені користувачем, некоректні значення, виконання програми продовжувалось, а при написанні звіту, вже знав, що необхідно демонструвати й у якому вигляді.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Sinha A. Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry [Електронний ресурс] / Abhiup Sinha, Pallabi Das // 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 24–26 верес. 2021 р. – [Б. м.], 2021. – Режим доступу: https://doi.org/10.1109/iementech53263.2021.9614779
2. Lazar G. Mastering Qt 5: Create stunning cross-platform applications / Guillaume Lazar, Robin Penea. – [Б. м.] : Packt Publishing, 2016. – 526 с.
3. Stroustrup B. C++ Programming Language / Bjarne Stroustrup. – [Б. м.] : Addison Wesley, 2013. – 1368 с.
4. Sergievskiy M. Optimizing UML Class Diagrams [Електронний ресурс] / Maxim Sergievskiy, Ksenia Kirpichnikova // ITM Web of Conferences. – 2018. – Т. 18. – С. 03003. – Режим доступу: https://doi.org/10.1051/itmconf/20181803003.
5. Gunawardhana L. K. P. D. Process of Requirement Analysis Link to Software Development [Електронний ресурс] / L. K. P. Dhananjaya Gunawardhana // Journal of Software Engineering and Applications. – 2019. – Т. 12, № 10. – С. 406–422. – Режим доступу: https://doi.org/10.4236/jsea.2019.1210025.
6. Eng L. Z. Qt5 C++ GUI Programming Cookbook: Design and build a functional, appealing, and user-friendly graphical user interface / Lee Zhi Eng. – [Б. м.] : Packt Publishing - ebooks Account, 2016. – 300 с.
7. Malevris N. Software Testing [Електронний ресурс] / Nicos Malevris // Journal of Computer Engineering & Information Technology. – 2013. – Т. 02, № 01. – Режим доступу: https://doi.org/10.4172/2324-9307.1000e105.
8. Suriya Sundaramoorthy, UML Diagramming: A Case Study Approach [Текст] / Suriya Sundaramoorthy – New York : Auerbach Publications, 2022. – 430 с.
9. UML 2. (. D. G. UML 2002-- the Unified Modeling Language: Model engineering, concepts, and tools : 5th International Conference, Dresden, Germany, September 30-October 4, 2002 : proceedings / 2002 (2002 Dresden Germany) UML. – Berlin : Springer, 2002. – 447 с.

10. Алгоритми: побудова й аналіз, 3-е видання, том 2 [Текст] / Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. – Вид-во «Диалектика-Вильямс», 2019. – 664 с.
11. ДСТУ ГОСТ 7.1-2006. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : чинний з 2007-07-01. – К. : Держспоживстандарт України, 2007. – 47 с. (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
12. Нікандров В.В.. Експериментальна психологія [Текст]: Навчальний посібник / В. В. Нікандров – СПб.: Вид-во «Мова», 2003. – 480 с.
13. Основні принципи ООП: інкапсуляція, успадкування, поліморфізм [Електронний ресурс] – Режим доступу: www.gos-it.wikia.com/wiki/osnovi_opp/
14. Івченко, Ю. М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн.трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с
15. Мямлін, В. В. Развитие научных основ создания гибких поточных технологий ремонта подвижного состава [Текст]: диссертация / В. В. Мямлін. – Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервис», 2015. – 380с.
16. Бараш Ю. С. Рациональные пути развития технической базы для деповского ремонта грузовых вагонов [Електронний ресурс] : автореф. thesis / Бараш Юрий Савельевич ; Днепропетровский институт инженеров железнодорожного транспорта. – [Б. м.], 1982. – Режим доступу: http://eadnurt.diit.edu.ua/jspui/handle/123456789/9255.
17. Стратегії тестування [Електронний ресурс] – Режим доступу: www.4stud.info/software-construction-and-testing/lecture10.html
18. Мямлін, В. В. Теоретические основы создания гибких поточных производств для ремонта подвижного состава [Текст]: монографія / В. В. Мямлін. – Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервис», 2014. – 380с.

19. Тестування програмного забезпечення [Текст]: навчальний посібник / Авраменко А.С., Авраменко В.С. Косенюк Г.В. – Черкаси : ЧНУ імені Богдана Хмельницького, 2017. – 284 с.
20. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп’ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с.

ДОДАТКИ

Додаток А

ТЕКСТ ПРОГРАМИ

animation.cpp

```
#include "animation.h"

Animation::Animation(VObject *obj) :
obj(obj)
{

}

Animation::Animation(VObject *obj,
float time) : obj(obj), time(time)
{

}

void Animation::update(float
deltaSeconds)
{
   time -= deltaSeconds;
}

void Animation::updateTime(float time)
{
   this->time = time;
}

bool Animation::isAlive()
{
   return obj != nullptr && time > 0.0;
}

AMove::AMove(VObject *obj, QPoint
to) : Animation(obj), to(to)
{

}

AMove::AMove(VObject *obj, QPoint
to, float time) : Animation(obj, time),
to(to)
{

}

AMove::AMove(VObject *obj, QPoint
to, float speed, bool):AMove(obj, to)
{
   const auto x = to.x() - obj->pos.x();
   const auto y = to.y() - obj->pos.y();
   this->time = sqrt(x*x + y*y)/speed;
}

void AMove::update(float deltaSeconds)
{
   auto to_obj = to-obj->pos;
   if (time <= 0.0)
      obj->pos = to;
   else
      obj->pos +=
to_obj*(deltaSeconds/time);

   Animation::update(deltaSeconds);
}
```

animation.h

```
#ifndef ANIMATION_H
#define ANIMATION_H

#include "robject.h"
#include "math/vecmath.h"


struct Animation
{
   VObject *obj = nullptr;
```

```
    float time = 0.0;

    explicit Animation(VObject *obj);
    Animation(VObject *obj, float time);
    virtual ~Animation() = default;

    virtual void update(float deltaSeconds);

    void updateTime(float time);
    bool isAlive();
};

struct AMove: public Animation
{
     AMove(VObject *obj, QPoint to);
     AMove(VObject *obj, QPoint to, float
time);
     AMove(VObject *obj, QPoint to, float
speed, bool);
    virtual ~AMove() = default;

    void update(float deltaSeconds)
override;

private:
    QPoint to;
};

#endif // ANIMATION_H
```

config.cpp

```
#include "config.h"
#include <QDebug>

Config::Config()
{

}

void Config::parse(QFile& file)
{
    if (!file.open(QIODevice::ReadOnly |
QIODevice::Text))
        return;

    QJsonDocument doc =
QJsonDocument::fromJson(file.readAll())
;
    QJsonObject JON = doc.object();
    if
(JON["Position"].toObject().value("NUM
BER").isArray())
    {
        position.MIN_NUMBER =
JON["Position"].toObject().value("NUM
BER")[0].toInt();
        position.MAX_NUMBER =
JON["Position"].toObject().value("NUM
BER")[1].toInt();
    }
    if
(JON["Position"].toObject().value("COU
NT_MODULES").isArray())
    {
        position.MIN_COUNT_MODULES
=
JON["Position"].toObject().value("COU
NT_MODULES")[0].toInt();

position.MAX_COUNT_MODULES =
JON["Position"].toObject().value("COU
NT_MODULES")[1].toInt();
    }
    if
(JON["Position"].toObject().value("AVG
_TIME").isArray())
    {
```

```
        position.MIN_AVG_TIME =
JON["Position"].toObject().value("AVG_
TIME")[0].toDouble();
        position.MAX_AVG_TIME =
JON["Position"].toObject().value("AVG_
TIME")[1].toDouble();
    }
    if
(JON["Position"].toObject().value("STEP
_TIME").isArray())
    {
        position.MIN_STEP_TIME =
JON["Position"].toObject().value("STEP
_TIME")[0].toDouble();
        position.MAX_STEP_TIME =
JON["Position"].toObject().value("STEP
_TIME")[1].toDouble();
    }

    if
(JON["Module"].toObject().value("SIZE"
).isArray())
    {
        platform.WIDTH =
JON["Module"].toObject().value("SIZE")
[0].toInt();
        platform.HEIGHT =
JON["Module"].toObject().value("SIZE")
[1].toInt();
    }

    if
(JON["Transborder"].toObject().value("SI
ZE").isArray())
    {
        transborder.WIDTH =
JON["Transborder"].toObject().value("SI
ZE")[0].toInt();
        transborder.HEIGHT =
JON["Transborder"].toObject().value("SI
ZE")[1].toInt();
    }

    if
(JON["Train"].toObject().value("SIZE").i
sArray())
    {
        train.WIDTH =
JON["Train"].toObject().value("SIZE")[0
].toInt();
        train.HEIGHT =
JON["Train"].toObject().value("SIZE")[1
].toInt();
    }

    if
(JON["MAX_COUNT_TRAINS"].isDou
ble())
        MAX_COUNT_TRAINS =
JON["MAX_COUNT_TRAINS"].toInt();
}

Config& Config::set()
{
    static Config instance;
    return instance;
}

Config const& Config::get()
{
    return set();
}
```

config.h

```
#ifndef CONFIG_H
#define CONFIG_H

#include <QFile>
```

```
#include <QJsonObject>
#include <QJsonDocument>

struct Config
{
public:
   struct Position {
      int MIN_NUMBER = 1;
      int MAX_NUMBER = 10;
      int MIN_COUNT_MODULES = 1;
      int MAX_COUNT_MODULES =
10;
      float MIN_AVG_TIME = 1.0f;
      float MAX_AVG_TIME = 1000.0f;
      float MIN_STEP_TIME = 1.0f;
      float MAX_STEP_TIME = 1000.0f;
   } position;

   struct Platform {
      int WIDTH = 180;
      int HEIGHT = 60;
   } platform;

   struct Transborder {
      int WIDTH = 180;
      int HEIGHT = 50;
   } transborder;

   struct Train {
      int WIDTH = 120;
      int HEIGHT = 40;
   } train;

   int MAX_COUNT_TRAINS = 1000;

public:
   void parse(QFile& file);
public:
   static Config& set();
   static Config const& get();

private:
   Config();
   Config(const Config&);
   Config& operator=(const Config&) =
delete;

};


#endif // CONFIG_H
```

delegate.cpp

```
#include "delegate.h"
#include <QSpinBox>

QWidget*
Delegate::createEditor(QWidget *parent,
const QStyleOptionViewItem & option,
                       const QModelIndex &
index) const
{
   QSpinBox *editor = new
QSpinBox(parent);
   editor->setFrame(false);
   const auto& column = index.column();
   if (column <
static_cast<int>(columnMinAndMax.size
()))
   {
      editor-
>setMinimum(columnMinAndMax[colu
mn].first);
      editor-
>setMaximum(columnMinAndMax[colu
mn].second);
   }

   return editor;
```

```
}
```

delegate.h

```
#ifndef DELEGATE_H
#define DELEGATE_H

#include <QLineEdit>
#include <QItemDelegate>

class Delegate : public QItemDelegate
{
public:
   std::vector<std::pair<int, int>>
columnMinAndMax;
   QWidget* createEditor(QWidget
*parent, const QStyleOptionViewItem &
option,
                  const QModelIndex &
index) const;
};

#endif // DELEGATE_H
```

glwidget.cpp

```
#include "glwidget.h"
GLWidget::GLWidget(QWidget *parent)
   : QOpenGLWidget(parent)
{
   setAutoFillBackground(false);
}

void GLWidget::animate()
{
//    elapsed = (elapsed +
qobject_cast<QTimer*>(sender())-
>interval()) % 1000;
   update();
   if (world != nullptr)
   {
      world->update(speed);
   }
}

void GLWidget::paintEvent(QPaintEvent
*event)
{
   QPainter painter;
   painter.begin(this);

painter.setRenderHint(QPainter::Antialias
ing);
   painter.fillRect(event->rect(),
Qt::white);
   QFont font;
   font.setPixelSize(24);
   painter.setFont(font);
   if (world != nullptr)
   {
      world->render(&painter);
   }
   painter.end();
}
```

glwidget.h

```
#ifndef GLWIDGET_H
#define GLWIDGET_H

#include <QTimer>
#include <QPainter>
#include <QPaintEvent>
#include <QOpenGLWidget>

#include "model/robject.h"

class GLWidget : public
QOpenGLWidget
{
   Q_OBJECT

public:
   float speed = 0.2f;
```

```
    RObject *world = nullptr;

public:
    explicit GLWidget(QWidget *parent);

public slots:
    void animate();

protected:
    void paintEvent(QPaintEvent *event)
override;

};

#endif // GLWIDGET_H
```

main.cpp

```
#include "widget.h"

#include <QApplication>
#include "model/manager.h"
#include "config.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QFile configFile("./config.json");
    Config::set().parse(configFile);
    configFile.close();

    Widget w;
    w.show();

    return a.exec();
}
```

manager.cpp

```
#include "manager.h"

Manager::Manager():
    x_rightColumn{
Config::get().platform.WIDTH+Config::g
et().transborder.WIDTH },

transborder(QPoint(Config::get().platfor
m.WIDTH, 0))
{

}

Manager::~Manager()
{
    for(size_t i = 0; i < trains.size(); i++)
        delete trains[i];

    // Должен быть пуст, но если было
экстренное завершение
    for(size_t i = 0; i < animations.size();
i++)
        delete animations[i];
}

void Manager::update(float deltaSeconds)
{
    updateTasks();
    for (auto& animation: animations)
    {
        animation->update(deltaSeconds);
        if (!animation->isAlive()) {
            animation->update(deltaSeconds);
            animation->obj->isAnimating--;
            delete animation;
            animation = nullptr;
        }
    }

animations.erase(std::remove(animations.
begin(), animations.end(), nullptr),
                 animations.end());
```

```
}

void Manager::render(QPainter *painter)
{
   for (auto& [_, position]: positions)
      position.render(painter);

   // Рельсы для трансбордера
   const auto& x_transborder =
transborder.pos.x();
   for (int i = 1; i < 6; i++)
      painter->drawRect(QRect(

x_transborder+i*Config::get().transborder
.WIDTH/6, 0,
                        1,
leftColumnHeight>rightColumnHeight?le
ftColumnHeight:rightColumnHeight));

   transborder.render(painter);

   for (auto& train: activeTP)
      train->train.render(painter);

}

void Manager::addPlatform(int type, float
avgTime, float step, bool leftColumn)
{
   // Если этот модуль ещё не создан
   if (positions.emplace(type,
Position(type, avgTime, step)).second)
      // emplace returns pair<iter,bool>
where bool means whether the element
was created or not
   {

      if (leftColumn)
      {

positions[type].pos.setX(x_leftColumn);

positions[type].pos.setY(leftColumnHeig
ht);
      }
      else
      {

positions[type].pos.setX(x_rightColumn);

positions[type].pos.setY(rightColumnHei
ght);
      }
   }
   if (leftColumn)
      leftColumnHeight +=
Config::get().platform.HEIGHT;
   else
      rightColumnHeight +=
Config::get().platform.HEIGHT;

   positions[type].addPlatform();
}

void Manager::addTrain(QImage image,
std::vector<int> tasks)
{
   std::reverse(tasks.begin(), tasks.end());
   auto train = new Task{ Train(image),
tasks };
   trains.push_back(train);
   startTP.push_back(train);
}

int Manager::getCountAll()
{
   return static_cast<int>(trains.size());
}
```

```
int Manager::getCountOnStart()
{
   return static_cast<int>(startTP.size());
}

int Manager::getCountActive()
{
   return
static_cast<int>(activeTP.size());
}

void
Manager::addMoveAnimation(VObject*
obj, QPoint pos)
{
   // magic number 20.0
   animations.push_back(new
AMove(obj, pos, 20.0, true));
   obj->isAnimating++;
}

void
Manager::addWaitAnimation(VObject*
obj, int position)
{
   animations.push_back(new
Animation(obj,
positions[position].getWaitingTime()));
   obj->isAnimating++;
}

// если у поезда task:
//              -1: он находится в
неопределенном состоянии(ожидает)
//              -2: он завершил
выполнение всех задач(свободен)
void Manager::updateTrainTask(Task
*train, const int task)
{
   const auto& trainOnTransborder =
transborder.train;
   // 1
   if (&train->train ==
trainOnTransborder && train-
>currentPlatform != nullptr) {
      // 2
      if (transborder.from != nullptr /*&&
&train->train == transborder.from-
>get()*/) {
         transborder.from->free();
         transborder.from = nullptr;
         addMoveAnimation(&train-
>train, transborder.getPlacePos());
         // 3
      } else if (&train->train == train-
>currentPlatform->get()) {
         transborder.train = nullptr;
         // время ожидания на
платформе(кроме первой)
         addWaitAnimation(&train->train,
train->currentTask);
         // isWorking true
         train->currentPlatform-
>isWorking = true;
         // 4
         if (task == -1)
            train->tasks.push_back(-2);
      } else {
         auto transborderPos =
QPoint(transborder.pos.x(), train-
>currentPlatform->pos.y());
         // 5
         if (transborder.pos ==
transborderPos) {
            train->currentPlatform-
>set(&train->train);
```

```
            addMoveAnimation(&train->train, train->currentPlatform->getPlacePos());
        } else {

addMoveAnimation(transborder.train, QPoint(transborderPos.x()+transborder.localePlace.x(),

transborderPos.y()+transborder.localePlace.y()));

addMoveAnimation(&transborder, transborderPos);
        }
      }
      // 6
   } else if (task == Config::get().position.MIN_NUMBER) {
      train->tasks.pop_back();
      addWaitAnimation(&train->train, task);
      // isWorking true
      train->currentPlatform->isWorking = true;
      // 7
   } else if (transborder.isFree() && train->currentPlatform != nullptr) {
      auto platformToGo = positions[task].getFreePlatform();
      // 8
      if (platformToGo != nullptr)
      {
         transborder.train = &train->train;
         transborder.from = train->currentPlatform;

addMoveAnimation(&transborder, QPoint(transborder.pos.x(), train->currentPlatform->pos.y()));
         train->currentTask = task;
         train->currentPlatform = platformToGo;
         train->tasks.pop_back();
      }
   }
}

void Manager::tryAddMoreTrains()
{
   Platform *startPosition;
   while (!startTP.empty() && (startPosition = positions[Config::get().position.MIN_NUMBER].getFreePlatform()) != nullptr)
   {
      auto& currentTrain = startTP.back();
      activeTP.push_back(currentTrain);
      startTP.pop_back();

      currentTrain->train.pos.setY(startPosition->getPlacePos().y());
      startPosition->set(&currentTrain->train);
      currentTrain->currentPlatform = startPosition;
      addMoveAnimation(&currentTrain->train, startPosition->getPlacePos());
   }
}

// если у поезда task:
//          -1: он находится в неопределенном состоянии(ожидает)
//          -2: он завершил выполнение всех задач(свободен)
```

```
void Manager::updateTasks()
{
   // 1
   if (positions.empty())
      return;

   tryAddMoreTrains();

   // освободить уже отработавшие вагоны

activeTP.erase(std::remove_if(activeTP.begin(), activeTP.end(),
                  [&](Task *t) {
                     return t->currentPlatform == &deadPlatform && t->train.isAnimating == 0;
                  }), activeTP.end());

   // 2
   for (auto& train: activeTP) {
      int task = train->tasks.empty() ? -1 : train->tasks.back();
      // если анимируется либо тележка, либо данный вагон
      // 3
      if (train->train.isAnimating != 0 || transborder.isAnimating != 0) {
         continue;
         // 4
      } else if (task == -2)
         // если вагон уже завершил задачи и не анимируется
      {
         // isWorking false
         train->currentPlatform->isWorking = false;
         train->currentPlatform->free();
         train->currentPlatform = &deadPlatform;
         const auto widthScene = Config::get().platform.WIDTH*2+Config::get().transborder.WIDTH+Config::get().train.WIDTH;
         addMoveAnimation(&train->train, QPoint(widthScene, train->train.pos.y()));
      } else {
         // isWorking false
         train->currentPlatform->isWorking = false;
         updateTrainTask(train, task);
      }
   }
}
```

manager.h

```
#ifndef MANAGER_H
#define MANAGER_H

#include "../config.h"
#include "train.h"
#include "platform.h"
#include "position.h"
#include "transborder.h"
#include "animation.h"

struct Task
{
   Train train;
   std::vector<int> tasks;
   int currentTask = -1;
   Platform* currentPlatform = nullptr;
};

class Manager: public RObject
{
public:
```

```
    Manager();
    ~Manager() override;

    void update(float deltaSeconds)
override;
    void render(QPainter *painter)
override;

    void addTrain(QImage image,
std::vector<int> tasks);
    void addPlatform(int type, float
avgTime, float step, bool leftColumn);

    int getCountAll();
    int getCountOnStart();
    int getCountActive();

private:
    const int x_leftColumn = 0;
    const int x_rightColumn = 0+180+180;

private:
    // start trains
    std::vector<Task*> startTP;
    // active trains
    std::vector<Task*> activeTP;
    // all trains
    std::vector<Task*> trains;

    Platform deadPlatform;
    std::map<int, Position> positions;
    int leftColumnHeight = 0;
    int rightColumnHeight = 0;

    Transborder transborder; // 140, 0

    std::vector<Animation*> animations;

private:
    void addMoveAnimation(VObject*
obj, QPoint pos);
    void addWaitAnimation(VObject* obj,
int position);

    void updateTasks();
    void updateTrainTask(Task *train,
const int task);

    void tryAddMoreTrains();
};

#endif // MANAGER_H
```

platform.cpp

```
#include "platform.h"

Platform::Platform()
{
}

Platform::Platform(int number, QPoint
pos)
{
    VObject::pos = pos;
    this->number = number;
    width =
Config::get().platform.WIDTH;
    height =
Config::get().platform.HEIGHT;

localePlace.setX((Config::get().platform.
WIDTH-Config::get().train.WIDTH)/2);

localePlace.setY((Config::get().platform.
HEIGHT-
Config::get().train.HEIGHT)/2);
}

void Platform::render(QPainter *painter)
```

```
{
    painter->setBrush(QBrush(QColor(100, 100, 100)));
    painter->drawRect(QRect(pos.x(), pos.y(), width, height));
    // Рельсы
    const auto magicNumber = Config::get().platform.HEIGHT/3;
    painter->drawRect(QRect(pos.x(), pos.y()+magicNumber, width, 2));
    painter->drawRect(QRect(pos.x(), pos.y()+magicNumber*2, width, 2));

    // Сигнал
    if (isWorking)
        painter->setBrush(QBrush(Qt::red));
    else
        painter->setBrush(QBrush(Qt::green));

    painter->drawRect(
            pos.x()+width/2, pos.y(),
            10, 10
    );
    painter->setBrush(QBrush(Qt::black));

    // Номер
    painter->setPen(Qt::white);
    painter->drawText(
            pos.x()+width/2-10,pos.y()+height/2,
            QString::number(number)
    );
    painter->setPen(Qt::black);
}

bool Platform::set(Train *train)
{
    if (!isFree())
        return false;
    this->train = train;
    return true;
}

Train* Platform::get()
{
    return train;
}

void Platform::free()
{
    train = nullptr;
}

bool Platform::isFree()
{
    return train == nullptr;
}

QPoint Platform::getPlacePos()
{
    return QPoint(pos.x()+localePlace.x(), pos.y()+localePlace.y());
}
```

platform.h

```
#ifndef PLATFORM_H
#define PLATFORM_H

#include "../config.h"
#include "robject.h"
#include "train.h"

class Platform : public VObject
{
public:
    bool isWorking = false;
    QPoint localePlace; // 30 10
```

```
public:
    Platform();
    explicit Platform(int number, QPoint
position);

    void render(QPainter *painter)
override;

    void free();
    bool isFree();

    bool set(Train *train);
    Train *get();

    QPoint getPlacePos();

private:
    Train *train = nullptr;
    int number = -1;

    int width = 180; // 140
    int height = 60; // 90
};

#endif // PLATFORM_H
```

position.cpp

```
#include "position.h"

Position::Position()
{

}

Position::Position(int type, float avgTime,
float step)
    : type(type), avgTime(avgTime),
step(step)
{

}

Position::~Position()
{

}

void Position::render(QPainter *painter)
{
    for (auto& platform: platforms)
        platform.render(painter);
}

void Position::addPlatform()
{
    const auto number =
(QString::number(type)+QString::number
(platforms.size()+1)).toInt();
    platforms.emplace_back(number,
QPoint(pos.x(), pos.y()+currentHeight));
    currentHeight +=
Config::get().platform.HEIGHT;
}

Platform* Position::getFreePlatform()
{
//    for (auto& platform: platforms)
//        if  (platform.isFree())
//            return &platform;
    auto rv = std::find_if(platforms.begin(),
platforms.end(), [](auto p){ return
p.isFree(); });
    return (rv != platforms.end()) ? &*rv:
nullptr;

//    return nullptr;
}
```

```
float Position::getWaitingTime()
{
    std::random_device rd{};
    std::mt19937 gen{rd()};
    std::normal_distribution<float>
d{avgTime, step};
    return d(gen);
}
```

position.h

```
#ifndef POSITION_H
#define POSITION_H

#include <random>
#include "platform.h"

class Position: public VObject
{
public:
    int type = -1;
    float avgTime = 20.0;
    float step = 5.0;
    int currentHeight = 0;

public:
    Position();
    explicit Position(int type, float
avgTime, float step);
    ~Position() override;

    void render(QPainter *painter)
override;

    void addPlatform();
    Platform *getFreePlatform();
    float getWaitingTime();

private:
    std::vector<Platform> platforms;
};

#endif // POSITION_H
```

robject.cpp

```
#include "robject.h"

RObject::RObject()
{

}

void RObject::update(float)
{

}
```

robject.h

```
#ifndef ROBJECT_H
#define ROBJECT_H

#include <QPainter>

// Renderable Object
struct RObject
{
    RObject();
    virtual ~RObject() = default;

    virtual void update(float deltaTime);
    virtual void render(QPainter *painter)
= 0;
};

// Visible Object
struct VObject: public RObject
{
    QPoint pos;
    // if it isn't 0, it means that it is
animating
    int isAnimating = 0;
};
```

```
#endif // ROBJECT_H
```

train.cpp

```
#include "train.h"

Train::Train(QImage image) :
    image(image)
{
    width = Config::get().train.WIDTH;
    height = Config::get().train.HEIGHT;

    pos = QPoint(-width-10,10); // -200 20
}

void Train::render(QPainter *painter)
{
    painter->drawImage(QRect(pos.x(),
pos.y(), width, height), image);
}
```

train.h

```
#ifndef TRAIN_H
#define TRAIN_H

#include "../config.h"
#include "robject.h"
#include "math/vecmath.h"

class Train : public VObject
{
public:
    Train(){};
    explicit Train(QImage image);
    ~Train() = default;

    void render(QPainter *painter)
override;

private:
    QImage image;
    int width = 120; // 100
    int height = 40; // 50
};

#endif // TRAIN_H
```

transborder.cpp

```
#include "transborder.h"

Transborder::Transborder()
{

}

Transborder::Transborder(QPoint
position)
{
    VObject::pos = position;

    width =
Config::get().transborder.WIDTH;
    height =
Config::get().transborder.HEIGHT;


localePlace.setX((Config::get().transbord
er.WIDTH-
Config::get().train.WIDTH)/2);

localePlace.setY((Config::get().platform.
HEIGHT-
Config::get().train.HEIGHT)/2);
}

void Transborder::render(QPainter
*painter)
{
    painter->setBrush(QBrush(QColor(80,
80, 80)));
```

```
    painter->drawRect(QRect(pos.x(),
pos.y()+localePlace.y()/2, width, height));

    // Рельсы
    const auto magicNumber =
Config::get().platform.HEIGHT/3;
    painter->drawRect(QRect(pos.x(),
pos.y()+magicNumber, width, 2));
    painter->drawRect(QRect(pos.x(),
pos.y()+magicNumber*2, width, 2));
}

bool Transborder::isFree()
{
    return train == nullptr;
}

QPoint Transborder::getPlacePos()
{
    return QPoint(pos.x()+localePlace.x(),
pos.y()+localePlace.y());
}
```

transborder.h

```
#ifndef TRANSBORDER_H
#define TRANSBORDER_H

#include "../config.h"
#include "robject.h"
#include "train.h"
#include "platform.h"
#include "math/vecmath.h"

class Transborder: public VObject
{
public:
    QPoint localePlace; // 30 10

    Train *train = nullptr;
    Platform *from = nullptr;
//    QPoint needed_pos;

public:
    Transborder();
    explicit Transborder(QPoint position);

    void render(QPainter *painter)
override;

    bool isFree();
    QPoint getPlacePos();

private:
    int width = 180; // 140
    int height = 50; // 90
};

#endif // TRANSBORDER_H
```

vecmath.cpp

```
#include "vecmath.h"


float len(const QPoint &point) {
    const auto x = point.x();
    const auto y = point.y();
    return sqrt(x*x + y*y);
}

QPoint to_direction(QPoint point) {
    return point/len(point);
}
```

vecmath.h

```
#ifndef VECMATH_H
#define VECMATH_H
#include <QPoint>
#include <math.h>

float len(const QPoint &point);
QPoint to_direction(QPoint point);
```

```
#endif // VECMATH_H
```

widget.cpp

```
#include "widget.h"
#include "./ui_widget.h"
#include "config.h"

#define repeat(what, howManyTimes) do
{    \
   for (int i = 0; i < howManyTimes; i++)
\
      what;
                  \
} while (false)


Widget::Widget(QWidget *parent)
   : QWidget(parent)
   , ui(new Ui::Widget)
{
   ui->setupUi(this);
   setupConfiguration();
   auto file =
QUrl::fromLocalFile(QDir::currentPath()
+"/music.mp3");
   playlist = new QMediaPlaylist();
   playlist->addMedia(file);
   playlist-
>setPlaybackMode(QMediaPlaylist::Loo
p);

   player = new QMediaPlayer();
   player->setPlaylist(playlist);
   QTimer *timer = new QTimer(this);
   connect(timer, &QTimer::timeout, ui-
>openGLW, &GLWidget::animate);
   connect(timer, &QTimer::timeout, this,
&Widget::updateCounters);
   timer->start(50);
}

Widget::~Widget()
{
   delete ui;
   delete player;
   delete playlist;
   if (world != nullptr)
      delete world;
}

void Widget::setupConfiguration()
{
   auto delegator = new Delegate;
   ui->tableWidget->item(0,0)-
>setText(QString::number(Config::get().p
osition.MIN_NUMBER));
   ui->tableWidget->item(1,0)-
>setText(QString::number(Config::get().p
osition.MIN_NUMBER+1));
   delegator-
>columnMinAndMax.push_back({

Config::get().position.MIN_NUMBER,

Config::get().position.MAX_NUMBER
                              });
   delegator-
>columnMinAndMax.push_back({

Config::get().position.MIN_COUNT_M
ODULES,

Config::get().position.MAX_COUNT_M
ODULES
                              });
   ui->tableWidget->item(0,1)-
>setText(QString::number(Config::get().p
osition.MIN_COUNT_MODULES));
```

```
    ui->tableWidget->item(1,1)-
>setText(QString::number(Config::get().p
osition.MIN_COUNT_MODULES));
    delegator-
>columnMinAndMax.push_back({

Config::get().position.MIN_AVG_TIME,

Config::get().position.MAX_AVG_TIME
                                });
    ui->tableWidget->item(0,2)-
>setText(QString::number(Config::get().p
osition.MIN_AVG_TIME));
    ui->tableWidget->item(1,2)-
>setText(QString::number(Config::get().p
osition.MIN_AVG_TIME));
    delegator-
>columnMinAndMax.push_back({

Config::get().position.MIN_STEP_TIME,

Config::get().position.MAX_STEP_TIM
E
                                });
    ui->tableWidget->item(0,3)-
>setText(QString::number(Config::get().p
osition.MIN_STEP_TIME));
    ui->tableWidget->item(1,3)-
>setText(QString::number(Config::get().p
osition.MIN_STEP_TIME));
    ui->tableWidget-
>setItemDelegate(std::move(delegator));
    ui->trainsCount-
>setMaximum(Config::get().MAX_COU
NT_TRAINS);
}

void Widget::setWorld(Manager *world)
{
    this->world = world;
    ui->openGLW->world = world;
}

void Widget::generateTrains(int count,
const std::vector<int>& tasks)
{
    for (int i = 0; i < count; i++)
    {
        QImage type;

switch(QRandomGenerator::global()-
>bounded(9))
        {
        case 0:
            type =
QImage(":/images/tanker.png");
            break;
        case 1:
            type =
QImage(":/images/tankerR.png");
            break;
        case 2:
            type =
QImage(":/images/tankerG.png");
            break;
        case 3:
            type =
QImage(":/images/half.png");
            break;
        case 4:
            type =
QImage(":/images/halfB.png");
            break;
        case 5:
            type =
QImage(":/images/halfG.png");
            break;
        case 6:
```

```
            type =
QImage(":/images/hopper.png");
            break;
        case 7:
            type =
QImage(":/images/hopperB.png");
            break;
        case 8:
            type =
QImage(":/images/hopperG.png");
            break;
        }
        world->addTrain(type, tasks);
    }
}

void
Widget::parseTable(std::vector<positionI
nitList>& input)
{
    for (int i = 0; i < ui->tableWidget-
>rowCount(); ++i) {
        const auto type = ui->tableWidget-
>item(i, 0)->text().toInt();
        const auto count = ui->tableWidget-
>item(i, 1)->text().toInt();
        const auto avg = ui->tableWidget-
>item(i, 2)->text().toFloat();
        const auto step = ui->tableWidget-
>item(i, 3)->text().toFloat();
        input.push_back({type, count, avg,
step});
    }
}

void Widget::run()
{
    if (ui->tableWidget->rowCount() < 2)
        return;
    // TODO: Убрать на релизе
    QFile configFile("./config.json");
    Config::set().parse(configFile);

    isMusicPlaying = false;
    switchMusicStatement();

    if (world != nullptr)
        delete world;
    setWorld(new Manager());

    std::vector<positionInitList> input;
    parseTable(input);

    std::sort(input.begin()+1, input.end()-1,
          [](const auto &l, const auto &r)
          {
              if (l.count != r.count) {
                  return l.count > r.count;
              }

              return l.type > r.type;
          });

    // Добавляю первый модуль в модель
    repeat(world-
>addPlatform(input[0].type, input[0].avg,
input[0].step, true), input[0].count);

    // Распределяю модули поровну
между двумя половинами
    int weight = input[input.size()-
1].count-input[0].count;
    for (auto position = input.begin()+1;
position != input.end()-1; ++position) {
        if(weight > 0) {
            repeat(world-
>addPlatform((*position).type,
```

```
(*position).avg, (*position).step, true),
(*position).count);
            weight -= (*position).count;
        } else {
            repeat(world-
>addPlatform((*position).type,
(*position).avg, (*position).step, false),
(*position).count);
            weight += (*position).count;
        }
    }

    // Добавляю последний модуль
    repeat(world-
>addPlatform(input[input.size()-1].type,
input[input.size()-1].avg,
input[input.size()-1].step, false),
input[input.size()-1].count);

    std::vector<int> allPositions;
    std::reverse(input.begin()+1,
input.end()-1);
    std::transform(input.begin(),
input.end(),
std::back_inserter(allPositions),
                   [](const positionInitList& p) -
> int { return p.type; });


    generateTrains(ui->trainsCount-
>text().toInt(), allPositions);
}

void Widget::speedChanged(int speed)
{
    ui->openGLW->speed =
float(speed)/10.0;
}
```

```
void Widget::addRowToTable()
{
    const auto count = ui->tableWidget-
>rowCount();
    if
(Config::get().position.MIN_NUMBER+
count >
Config::get().position.MAX_NUMBER)
        return;
    ui->tableWidget->insertRow(count);
    ui->tableWidget->setItem(count, 0,
new
QTableWidgetItem(QString::number(Con
fig::get().position.MIN_NUMBER+count
)));
    ui->tableWidget->setItem(count, 1,
new
QTableWidgetItem(QString::number(Con
fig::get().position.MIN_COUNT_MODU
LES)));
    ui->tableWidget->setItem(count, 2,
new
QTableWidgetItem(QString::number(Con
fig::get().position.MIN_AVG_TIME)));
    ui->tableWidget->setItem(count, 3,
new
QTableWidgetItem(QString::number(Con
fig::get().position.MIN_STEP_TIME)));
    auto item = ui->tableWidget-
>item(count, 0);
    item->setFlags(item->flags() ^
Qt::ItemIsEditable);
}

void Widget::removeRowFromTable()
{
    QModelIndexList selection = ui-
>tableWidget->selectionModel()-
>selectedRows();
```

```
    if (selection.size() > 0)
    {
        const auto row = selection[0].row();
        ui->tableWidget->removeRow(row);
        for (int i = row; i < ui->tableWidget->rowCount(); ++i)
        {
            auto item = ui->tableWidget->item(i,0);
            item->setText(QString::number(item->text().toInt()-1));
        }
    }
    else
    {
        QMessageBox messageBox;
        messageBox.critical(0,"Не вдалося видалити рядок","Для видалення рядку, його необхідно виділити за допомогою миші, після чого натиснути кнопку «Видалити рядок»");
    }
}

void Widget::updateCounters()
{
    if (world != nullptr)
    {
        // TODO: заменить текст(должно быть что-то типа Надійшло до ремонту и Відремонтовано или типа того
        ui->comeIn->setText("Надійшло: "+QString::number(world->getCountAll()-world->getCountOnStart()));
        ui->comeOut->setText("Відремонтовано: "+QString::number(world->getCountAll()-world->getCountActive()-world->getCountOnStart()));
    }
}

void Widget::switchMusicStatement()
{
        if (isMusicPlaying)
        {
            player->stop();
            ui->playOrStopBtn->setText("►");
        }
        else
        {
            player->play();
            ui->playOrStopBtn->setText("||");
        }
        isMusicPlaying = !isMusicPlaying;
}
```

widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QDir>
#include <QPainter>
#include <QWidget>
#include <QMessageBox>
#include <QMediaPlayer>
#include <QMediaPlaylist>
#include <QRandomGenerator>

#include "model/manager.h"
#include "delegate.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
```

```
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT

public:
    Widget(QWidget *parent = nullptr);
    ~Widget();

    void setWorld(Manager *world);

public slots:
    void run();
    void speedChanged(int value);
    void addRowToTable();
    void removeRowFromTable();
    void updateCounters();
    void switchMusicStatement();

private:
    Ui::Widget *ui;
    Manager *world = nullptr;
    QMediaPlayer *player;
    QMediaPlaylist *playlist;
    bool isMusicPlaying = false;

    struct positionInitList{
        int type;
        int count;
        float avg;
        float step;
    };

private:
    void setupConfiguration();
    void generateTrains(int count, const
std::vector<int>& tasks);
    void
parseTable(std::vector<positionInitList>
& input);
};
#endif // WIDGET_H
```

CMqmake.pro

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4):
QT += widgets
QT += multimedia

CONFIG += c++17

# You can make your code fail to compile
if it uses deprecated APIs.
# In order to do so, uncomment the
following line.
#DEFINES +=
QT_DISABLE_DEPRECATED_BEFOR
E=0x060000    # disables all the APIs
deprecated before Qt 6.0.0
SOURCES += \
    config.cpp \
    delegate.cpp \
    main.cpp \
    widget.cpp \
    glwidget.cpp \
    model/train.cpp \
    model/robject.cpp \
    model/manager.cpp \
    model/position.cpp \
    model/platform.cpp \
    model/animation.cpp \
    model/transborder.cpp \
    model/math/vecmath.cpp

HEADERS += \
    config.h \
```

```
    delegate.h \
    widget.h \
    glwidget.h \
    model/train.h \
    model/robject.h \
    model/manager.h \
    model/position.h \
    model/platform.h \
    model/animation.h \
    model/transborder.h \
    model/math/vecmath.h

FORMS += \
    widget.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path =
/opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS +=
target

RESOURCES += \
    resources.qrc
```

resources.qrc

```
<RCC>
    <qresource prefix="/">
        <file>images/half.png</file>
        <file>images/tanker.png</file>
        <file>images/hopper.png</file>
        <file>images/hopperB.png</file>
        <file>images/tankerR.png</file>
        <file>images/halfB.png</file>
        <file>images/halfG.png</file>
        <file>images/hopperG.png</file>
        <file>images/tankerG.png</file>
    </qresource>
</RCC>
```

widget.ui

```
<?xml version="1.0" encoding="UTF-
8"?>
<ui version="4.0">
 <class>Widget</class>
 <widget class="QWidget"
name="Widget">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>840</width>
    <height>520</height>
   </rect>
  </property>
  <property name="minimumSize">
   <size>
    <width>840</width>
    <height>520</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>1024</width>
    <height>16777215</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>Widget</string>
  </property>
  <layout class="QHBoxLayout"
name="horizontalLayout_3">
   <item>
    <widget class="GLWidget"
name="openGLW">
     <property name="minimumSize">
      <size>
       <width>540</width>
       <height>500</height>
```

```
     </size>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QHBoxLayout"
name="horizontalLayout">
    <item>
     <layout class="QVBoxLayout"
name="verticalLayout">
      <item>
       <layout class="QHBoxLayout"
name="horizontalLayout_4">
        <item>
         <widget class="QLabel"
name="label">
          <property
name="layoutDirection">
           <enum>Qt::LeftToRight</enum>
          </property>
          <property name="text">
           <string>Музика:</string>
          </property>
          <property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt
::AlignVCenter</set>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton"
name="playOrStopBtn">
          <property name="text">
           <string>►</string>
          </property>
         </widget>
        </item>
       </layout>
      </item>
      <item>
       <widget class="QLabel"
name="label_2">
        <property name="text">
         <string>ШВидкість
анімації</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QSlider"
name="speedSlider">
        <property name="maximum">
         <number>10</number>
        </property>
        <property name="pageStep">
         <number>2</number>
        </property>
        <property name="value">
         <number>2</number>
        </property>
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
       </widget>
      </item>
      <item>
       <spacer name="horizontalSpacer">
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint"
stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
```

```
      </spacer>
     </item>
     <item>
      <layout class="QHBoxLayout"
name="horizontalLayout_5">
       <item>
        <widget class="QLabel"
name="comeIn">
         <property name="font">
          <font>
           <pointsize>12</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Надійшло: 0</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLabel"
name="comeOut">
         <property name="font">
          <font>
           <pointsize>12</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Відремонтовано:
0</string>
         </property>
        </widget>
       </item>
      </layout>
     </item>
     <item>
      <widget class="QTableWidget"
name="tableWidget">
       <property
name="horizontalScrollBarPolicy">
<enum>Qt::ScrollBarAsNeeded</enum>
       </property>
       <property
name="sizeAdjustPolicy">

<enum>QAbstractScrollArea::AdjustToC
ontentsOnFirstShow</enum>
       </property>
       <property name="rowCount">
        <number>2</number>
       </property>
       <property name="columnCount">
        <number>4</number>
       </property>
       <attribute
name="horizontalHeaderDefaultSectionSi
ze">
        <number>65</number>
       </attribute>
       <attribute
name="horizontalHeaderHighlightSection
s">
        <bool>true</bool>
       </attribute>
       <attribute
name="verticalHeaderVisible">
        <bool>false</bool>
       </attribute>
       <row/>
       <row/>
       <column>
        <property name="text">
         <string notr="true">№</string>
        </property>
        <property name="font">
         <font>
          <pointsize>8</pointsize>
         </font>
```

```
        </property>
       </column>
       <column>
        <property name="text">
         <string>Кількість</string>
        </property>
        <property name="font">
         <font>
          <pointsize>8</pointsize>
         </font>
        </property>
       </column>
       <column>
        <property name="text">
         <string>Сер. час,
хв</string>
        </property>
        <property name="font">
         <font>
          <pointsize>8</pointsize>
         </font>
        </property>
       </column>
       <column>
        <property name="text">
         <string>Відступ,
хв</string>
        </property>
        <property name="font">
         <font>
          <pointsize>8</pointsize>
         </font>
        </property>
       </column>
       <item row="0" column="0">
        <property name="text">
         <string>1</string>
        </property>
        <property name="flags">
<set>ItemIsSelectable|ItemIsDragEnabled
|ItemIsDropEnabled|ItemIsUserCheckabl
e|ItemIsEnabled</set>
        </property>
       </item>
       <item row="0" column="1">
        <property name="text">
         <string>1</string>
        </property>
       </item>
       <item row="0" column="2">
        <property name="text">
         <string>1</string>
        </property>
       </item>
       <item row="0" column="3">
        <property name="text">
         <string>1</string>
        </property>
       </item>
       <item row="1" column="0">
        <property name="text">
         <string>2</string>
        </property>
        <property name="flags">
<set>ItemIsSelectable|ItemIsDragEnabled
|ItemIsDropEnabled|ItemIsUserCheckabl
e|ItemIsEnabled</set>
        </property>
       </item>
       <item row="1" column="1">
        <property name="text">
         <string>1</string>
        </property>
       </item>
       <item row="1" column="2">
        <property name="text">
```

```
        <string>1</string>
       </property>
      </item>
      <item row="1" column="3">
       <property name="text">
        <string>1</string>
       </property>
      </item>
     </widget>
    </item>
    <item>
     <widget class="QPushButton"
name="addRowBtn">
      <property name="font">
       <font>
        <pointsize>14</pointsize>
       </font>
      </property>
      <property name="text">
       <string>Додати рядок</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton"
name="removeRowBtn">
      <property name="font">
       <font>
        <pointsize>10</pointsize>
       </font>
      </property>
      <property name="text">
       <string>Видалити рядок</string>
      </property>
     </widget>
    </item>
    <item>
     <spacer
name="horizontalSpacer_2">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeHint"
stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item>
     <layout class="QHBoxLayout"
name="horizontalLayout_2">
      <item>
       <widget class="QLabel"
name="trainsCountLabel">
        <property name="text">
         <string>Кількість
вагонів:</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QSpinBox"
name="trainsCount"/>
      </item>
     </layout>
    </item>
    <item>
     <widget class="QPushButton"
name="pushButton">
      <property name="text">
       <string>Почати
моделювання</string>
      </property>
     </widget>
    </item>
```

```
    </layout>
   </item>
  </layout>
 </item>
</layout>
</widget>
<customwidgets>
 <customwidget>
  <class>GLWidget</class>
  <extends>QOpenGLWidget</extends>
  <header>glwidget.h</header>
  <slots>
   <slot>animate()</slot>
  </slots>
 </customwidget>
</customwidgets>
<resources/>
<connections>
 <connection>
  <sender>pushButton</sender>
  <signal>clicked()</signal>
  <receiver>Widget</receiver>
  <slot>run()</slot>
  <hints>
   <hint type="sourcelabel">
    <x>818</x>
    <y>508</y>
   </hint>
   <hint type="destinationlabel">
    <x>693</x>
    <y>416</y>
   </hint>
  </hints>
 </connection>
 <connection>
  <sender>speedSlider</sender>
  <signal>valueChanged(int)</signal>
  <receiver>Widget</receiver>
  <slot>speedChanged(int)</slot>
  <hints>
   <hint type="sourcelabel">
    <x>818</x>
    <y>79</y>
   </hint>
   <hint type="destinationlabel">
    <x>701</x>
    <y>252</y>
   </hint>
  </hints>
 </connection>
 <connection>
  <sender>addRowBtn</sender>
  <signal>clicked()</signal>
  <receiver>Widget</receiver>
  <slot>addRowToTable()</slot>
  <hints>
   <hint type="sourcelabel">
    <x>629</x>
    <y>388</y>
   </hint>
   <hint type="destinationlabel">
    <x>701</x>
    <y>381</y>
   </hint>
  </hints>
 </connection>
 <connection>
  <sender>removeRowBtn</sender>
  <signal>clicked()</signal>
  <receiver>Widget</receiver>
  <slot>removeRowFromTable()</slot>
  <hints>
   <hint type="sourcelabel">
    <x>761</x>
    <y>408</y>
   </hint>
   <hint type="destinationlabel">
    <x>824</x>
```

```
     <y>438</y>
    </hint>
   </hints>
  </connection>
  <connection>
   <sender>playOrStopBtn</sender>
   <signal>clicked()</signal>
   <receiver>Widget</receiver>
   <slot>switchMusicStatement()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>763</x>
     <y>20</y>
    </hint>
    <hint type="destinationlabel">
     <x>670</x>
     <y>4</y>
    </hint>
   </hints>
  </connection>
 </connections>
 <slots>
  <slot>run()</slot>
  <slot>speedChanged(int)</slot>
  <slot>addRowToTable()</slot>
  <slot>removeRowFromTable()</slot>
  <slot>switchMusicStatement()</slot>
 </slots>
</ui>
```