УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ
ТА ТЕХНОЛОГІЙ

Кафедра автоматики та комп'ютерного телекерування
рухом поїздів

МОДЕЛЮВАННЯ РОБОТОТЕХНІЧНИХ СИСТЕМ У V-REP

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт і самостійної роботи

з дисципліни

«ТЕОРЕТИЧНІ ОСНОВИ РОБОТОТЕХНІКИ»

Харків – 2024

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри автоматики та комп'ютерного телекерування рухом поїздів 25 березня 2024 р., протокол № 8.

Наведено короткі теоретичні відомості, завдання і порядок дій для виконання практичних робіт, а також рекомендації до самостійного вивчення відповідних розділів курсу.

Методичні вказівки призначені для здобувачів вищої освіти другого (магістерського) рівня спеціальності 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка», які вивчають дисципліну «Теоретичні основи робототехніки», усіх форм здобуття освіти.

Укладачі:

доценти С. О. Змій,

О. В. Щебликіна,

І. М. Сіроклин,

старш. викл. М. В. Ушаков

Рецензент

проф. В. І. Мойсеєнко

ЗМІСТ

ВСТУП

Сьогодні робототехніка є одним із найзатребуваніших напрямів у розробленні автоматизованих технічних систем і активно використовується в медицині, телекомунікаціях, військовій, промисловій та освітній сферах.

Моделювання вже давно стало невід'ємною частиною розроблення будь-якої технології або продукту. Воно економить час і гроші, а в деяких випадках допомагає запобігти впливу різних небезпек на сам продукт, людей і навколишнє середовище. У галузі робототехніки високий рівень складності технічних систем робить моделювання ще більш поширеним і актуальним. Чи то дослідницький проєкт, чи то новий продукт, призначений для масового виробництва, грамотне моделювання з використанням математичного апарату й обчислювальних потужностей сучасних комп'ютерів знижує як технічний ризик у процесі виробництва, так і ймовірність відмови роботизованої системи в наступних операціях.

Досягнення в галузі робототехніки та технології моделювання призвели до появи пакета програмного забезпечення, так званого симулятора роботів. Слід виокремити особливу важливість симуляторів у навчанні робототехніки. Віртуальні симулятори суттєво відрізняються від загальних середовищ моделювання. У віртуальних симуляторах використовують підхід, за якого в модель завантажують компоненти, що забезпечують ту саму функціональність, що й аналогічні компоненти в реальному роботі. Це дає змогу здобувачам навчитися працювати з електричними приводами, механічними компонентами та системами керування, маніпулюючи віртуальною моделлю, що знижує ризик завдання шкоди під час навчання.

Тестування – найпоширеніша, але не єдина мета розроблення експериментальних платформ. Наприклад, більшість сучасних промислових роботів мають власні системи автономного програмування, зокрема системи експериментального моделювання. Такі системи виконують усі необхідні для моделювання розрахунки та візуалізують процес.

Цілі моделювання залежать від поставленого завдання, а в разі створення нового робототехнічного рішення – в основному від етапу розроблення: і перевірка гіпотез, і оптимізація конструкції, і тестування програмного забезпечення, що реалізує нові алгоритми опрацювання сенсорної інформації та керування рухом, і налагодження виконуваного коду перед його виконанням на контролері робочої станції маніпулятора на наступних етапах.

Процес розроблення нелінійний, і хороше середовище комп'ютерного моделювання дає змогу уникнути багатьох проблем на ранніх етапах, даючи змогу тестувати як окремі компоненти робота, так і спрощені моделі, що реалізують окремі функції.

Існує низка програмних рішень, що забезпечують високоточне моделювання робототехнічних систем. З них найпопулярнішими є Gazebo і V-REP, що пропонують більше можливостей і широкий спектр моделювання різних типів роботів, від левітуючих до літаючих. Ще однією особливістю цих платформ є те, що вони легко масштабуються. Однак інтерфейс розробника V-REP значно кращий, ніж у Gazebo, яка не має інтерфейсу розроблення; програмний пакет Gazebo є повністю безкоштовним, але все розроблення в Gazebo ведеться мовою розмітки xml, що дуже трудомістко. Крім того, V-REP регулярно оновлюється і додає нові функції, тоді як Gazebo погано підтримують, і за останні три роки він не продемонстрував жодних значних поліпшень. Усе це робить V-REP на сьогодні найкращим рішенням для тих, хто хоче вивчити і почати застосовувати на практиці можливості моделювання процесів робототехнічних систем і особливості створення симуляторів роботів.

У методичних вказівках розглянуто хід практичних робіт, тематика яких охоплює питання побудови робототехнічних систем. З урахуванням змісту навчальної дисципліни наведено цілі, короткі теоретичні відомості, порядок виконання та приклади виконання поставлених завдань.

ПРАКТИЧНА РОБОТА 1

Моделювання роботи механічного захвату для промислового маніпулятора

1.1 Мета роботи: розроблення комп'ютерної 3D-моделі простого механічного захвату для маніпулятора та сценаріїв моделювання.

1.2 Завдання

1 Створити 3D-модель простого захвату, що складається з трьох елементів, у будь-якій CAD-системі та імпортувати її у V-REP (або створити модель механічного захвату безпосередньо у V-REP).

2 Визначити структуру моделі (встановити взаємозв'язки).

3 Додати в модель датчики і визначити алгоритм керування захватом. Захват повинен мати два стабільних положення: активне і початкове.

1.3 Порядок виконання

У роботі запропоновано змоделювати роботу механічного захвату на промисловому маніпуляторі. Перш ніж приступити до виконання, необхідно добре зрозуміти опис експериментального дослідження і відповісти на запитання для самоперевірки. Можливості програми наведено в додатку А.

Примітка. 3D-моделі можуть бути створені в будь-якій CAD-системі і не становлять особливої складності для імпорту у V-REP.

Для створення 3D-моделі механічного захвату безпосередньо в програмі V-REP створюють нову сцену, у якій за допомогою інструменту «Примітивна форма» створюють захват, що складається з трьох прямокутників.

Використовуючи команди «Add→Primitive Shape→Cuboid» і вводячи параметри (рисунок 1.1), можна одержати тривимірну модель основи захвату.

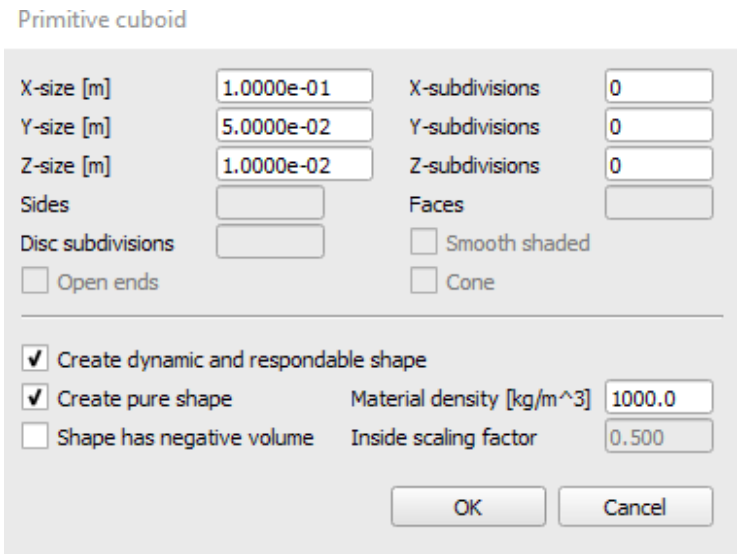

Рисунок 1.1 – Контекстне меню інструменту «Primitive Shape» (проста форма) при створенні тривимірної моделі основного компонента захвату

Далі необхідно створити другий і третій елемент, тут можна їх зробити однаковими, отже, достатньо створити один елемент і скопіювати (рисунок 1.2).

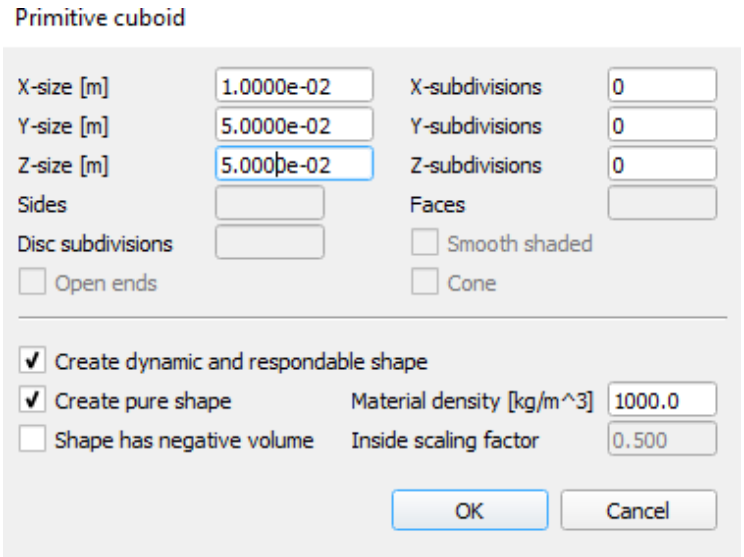

Рисунок 1.2 – Контекстне меню інструменту «Primitive Shape» (проста форма) для другого компонента

Тепер необхідно його скопіювати і задати всім трьом елементам назву відповідно до їхнього призначення. За замовчуванням V-REP задає їм імена, що відповідають назві інструменту, за допомогою якого вони були створені. Щоб перейменувати, достатньо навести курсор на ім'я і виконати подвійне натискання правої кнопки миші комп'ютера, задати нове ім'я і натиснути кнопку введення (Enter) на клавіатурі. *Зверніть увагу*, що V-REP не підтримує кириличні символи. Нові назви мають бути «Gripper_Base», «Right_side» і «Left_side».

Після цього необхідно задати коректне розташування для двох елементів механічного захвату відносно основи, відповідне вихідному положенню. Для цього виділяємо в лівому вікні дерева моделі Left_side і активуємо інструмент «Object/Item shift» з верхнього основного меню інструментів. У вікні інструменту, що з'явилося, вибираємо вкладку Position, потім вводимо нові значення розташування елемента за віссю «*x*» і віссю «*y*», оскільки тільки за цими осями поточні значення не відповідають необхідним (рисунок 1.3).

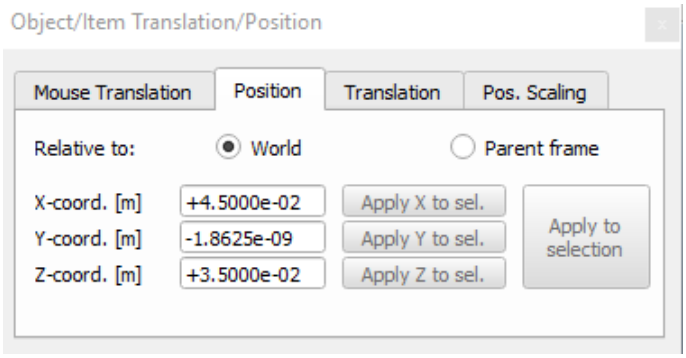

Рисунок 1.3 – Контекстне меню інструменту «Object/Item Position»

Потім необхідно вибрати елемент «Right_side» і задати відповідні значення зі зворотним за знаком значенням для осі «*x*» (рисунок 1.4).

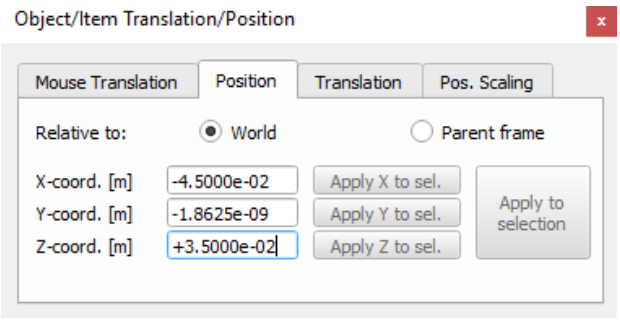

Рисунок 1.4 – Контекстне меню інструменту «Object/Item Position»

Після виконання вищеописаних операцій має вийти результат, як на рисунку 1.5.

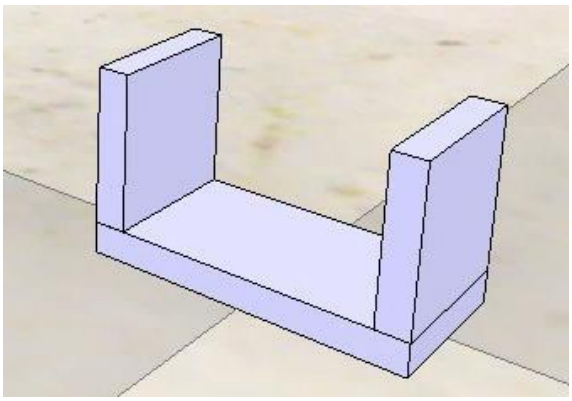

Рисунок 1.5 – Зовнішній вигляд створених компонентів типу «Random Shape»

Далі необхідно задати механічні з'єднання для наявних елементів за допомогою елементів «З'єднання». У цьому випадку необхідно додати два з'єднання типу «призматичний». Додати їх можна через команди «Add → Joint → Prismatic». Поруч із наявними елементами з'явиться новий елемент, який необхідно зменшити, оскільки його розмір задано за замовчуванням, і він не дає нам змоги з достатньою точністю керувати положенням елементів нашого механізму. Розміри елементів із категорії «З'єднання»

використовують тільки для зручності роботи з ними при створенні симуляції і ніяк не впливають на функціональні параметри з'єднання.

Змінимо розміри з'єднання через редагування властивостей, як показано на рисунку 1.6, для цього в дереві моделі вибираємо необхідний елемент з'єднання і відкриваємо вікно властивостей з'єднання за допомогою інструменту «Scene Object Properties» (перший інструмент у лівій панелі).

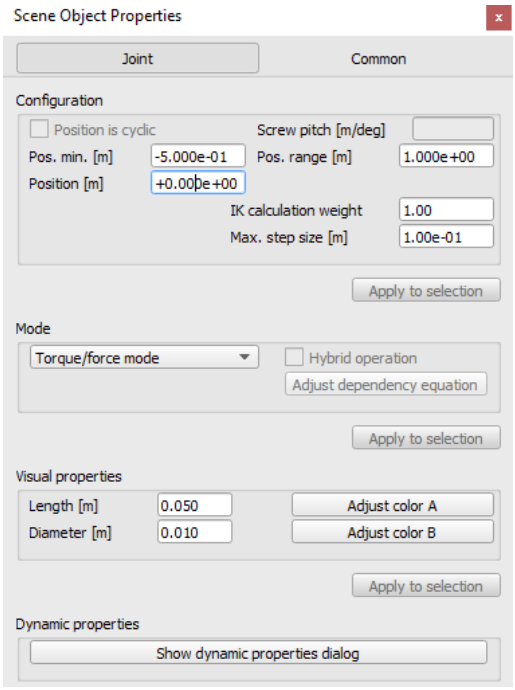

Рисунок 1.6 – Контекстне меню інструменту «Scene Object Properties»

Далі необхідно задати правильно розташування й орієнтацію в просторі для елемента з'єднання, що й можна зробити, скориставшись інструментами для зміни положення й орієнтації.

Опишемо детальніше основні прийоми, які дають змогу виконати вищеописані операції. Щоб задати положення для з'єднання, ми можемо скористатися візуальним інтерфейсом і за допомогою мишки «перетягнути» елемент на потрібне положення, однак такий підхід не рекомендований, оскільки не гарантує високої точності. Рекомендується використовувати більш швидкий спосіб – скопіювати координати іншого елемента і вже задавати нове

розташування. Скопіювати координати іншого елемента можна, виконавши такі операції: спочатку вибираємо в дереві моделі об'єкт, у який хочемо скопіювати координати, і, затиснувши клавішу «Ctrl», вибираємо другий елемент, чиї координати ми й хочемо скопіювати. Потім вибираємо інструмент «Object/Item shift» і у вкладці «Position» натискаємо на кнопку «Apply to selection» (показано на рисунку 1.7).

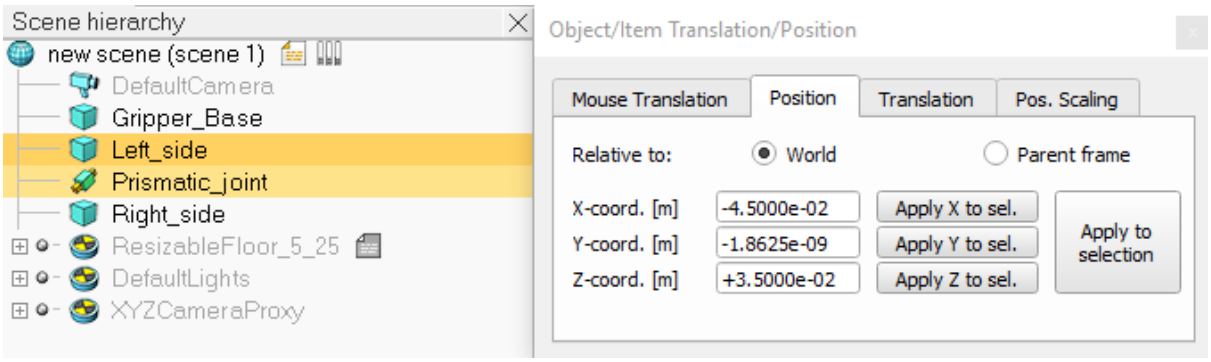

Рисунок 1.7 – Копіювання положення елемента «Left_side» у властивості шарніра

Так само можна скопіювати параметри орієнтації з одного об'єкта в інший, скориставшись інструментом «Object/Item rotate» замість «Object/Item shift».

Однак одного копіювання даних орієнтації та положення буває недостатньо, як і в нашому випадку. У таких випадках необхідно скористатися вкладкою «Translation» для інструменту «Object/Item shift», де можна вказати значення переміщення відносно поточного положення, і після натискання кнопки «Translate» буде застосовано зазначені зміни.

Схожа функція є і в інструменту «Object/Item rotate», де у вкладці «Rotation» можна вказати кути повороту відносно поточної орієнтації, вибравши вісь обертання, і після натискання кнопки «Rotate selection» отримати результат. Під час використання цього інструменту слід звернути увагу, що необхідно вибрати, відносно якої системи координат виконують

обертання. У більшості випадків цю операцію виконують відносно системи координат, пов'язаної з елементом, що обертається, і в такому разі слід у параметрі «Relative to» встановити на «Own Frame».

Отже, скориставшись вищеописаними прийомами, ми можемо отримати результат, поданий на рисунку 1.8.

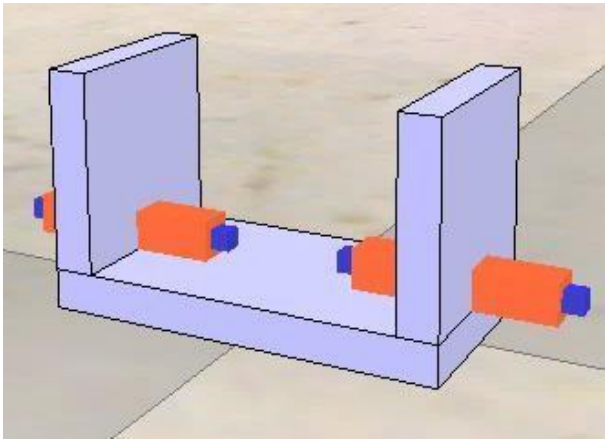

Рисунок 1.8 – Зовнішній вигляд елементів сцени з коректними координатами та орієнтацією

Після того як задали необхідне положення й орієнтацію для всіх елементів, необхідно задати зв'язки цих елементів.

Елемент під назвою «Gripper_Base» є основою нашого механізму і, отже, усі зв'язки мають виходити з нього. У V-REP доступні два способи для визначення зв'язків. Перший спосіб найпростіший і найшвидший – виділити і перетягнути один елемент на інший у дереві моделі. При цьому елементи мають утворювати деревоподібну структуру, де один елемент стоїть в основі (базовий елемент), а наступні елементи можуть з'єднуватися з основою або іншим елементом однією з гілок дерева. Другий спосіб – виділити елемент, який необхідно з'єднати з батьківським елементом, затиснути кнопку «ctrl» і виділити елемент, який хочемо зробити дочірнім, потім натиснути праву кнопку миші і в меню вибрати «Edit», «Make last

selected Object parent». Також під час визначення зв'язків необхідно дотримуватися такого правила: два елементи, що моделюють динаміку, не можна з'єднувати без використання елемента типу «шарнір», тобто щоб з'єднати елементи Gripper_Base і Left_side, необхідно між ними в дереві додати «Left_joint».

Так само необхідно задати зв'язки між «Gripper_Base» і «Right_side». Заключна деревоподібна структура, яку необхідно отримати, подана на рисунку 1.9.

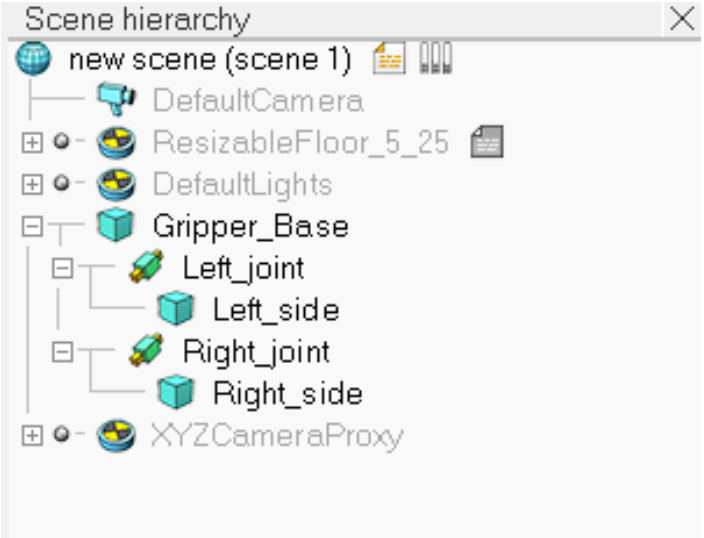

Рисунок 1.9 – Заключна ієрархія сцени з заданими зв'язками компонентів

Далі необхідно увімкнути вбудовані мотори в шарнірах «Left_joint» і «Right_joint», задати необхідні налаштування. Для цього вибрати елемент у дереві моделі і в лівій панелі активувати інструмент «Scene object properties», далі в контекстному меню інструменту, яке з'явиться, вибрати «Show dynamic properties dialog» і ввімкнути мотор, поставивши відповідну позначку навпроти «Motor enabled». Після цього активуються рядки номінальної швидкості обертання і максимального моменту – «Target velocity» і «Maximum force» відповідно. Зважаючи на порівняно малі розміри захвату, максимальну силу можна поставити на $5.0e^{+00}$ Ньютон і

швидкість залишити на 0 м/с, тому що в цій роботі його будемо задавати через скрипт. *Зверніть увагу*, що необхідно враховувати напрямок призматичного шарніра: залежно від його орієнтації потрібно вказати швидкість для кожного з захватів із позитивним або негативним знаком. Визначити знак можна експериментальним способом, провівши симуляцію сценарію і примусово увімкнувши мотори та додавши значення до швидкості. Якщо позитивне значення не збігається з необхідним напрямком, то необхідно розгорнути шарнір на 180°. Приклад налаштування шарніра наведено на рисунку 1.10.

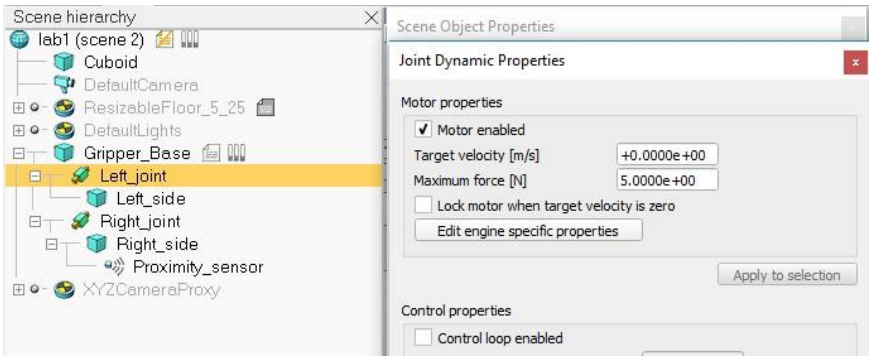

Рисунок 1.10 – Завершальна ієрархія сцени з заданими зв'язками компонентів

Тепер необхідно додати «Proximity Sensor» (датчик наближення). Додати його можна через головне меню «Add→Proximity Sensor→Ray Type». Далі необхідно розташувати (задати координати) сенсор на одному з рухомих елементів захвату так, як показано на рисунку 1.11. Після цього можна задати зв'язок між рухомим елементом захвату і сенсором через ієрархію сцени. Зробити це найпростіше методом «перетягнути і відпустити», перетягуємо сенсор «Proximity_sensor» на назву того елемента захвату, з яким поєднали сенсор, і відпускаємо (рисунок 1.12).

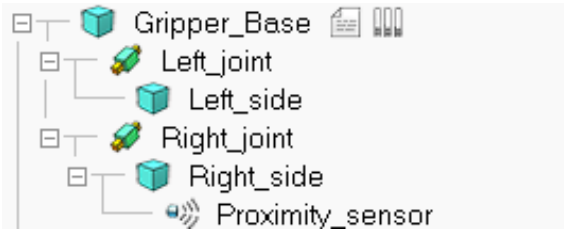

Рисунок 1.11 – Заключна ієрархія сцени з заданою прив'язкою «Proximity sensor» до захвату

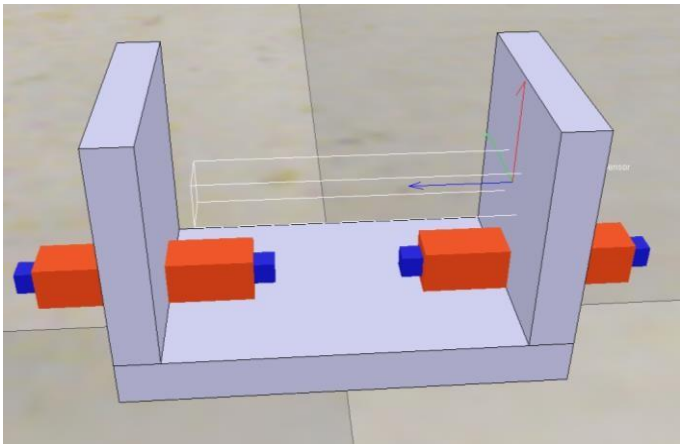

Рисунок 1.12 – Кінцеве розташування всіх об'єктів сцени

У властивостях «Proximity Sensor» необхідно перейти в меню «Show detection parameters» і зробити активним параметр «Don't allow detections if distance smaller than», встановивши значення 0. Це пов'язано з похибкою сенсора й об'єкта в захваті, який також потрібно додати в сценарій симуляції. Для цього скористаємося вже знайомим інструментом у меню «Add → Primitive shape → Cuboid» і задамо розміри, які підходять під масштаб нашого механічного захвату (рисунок 1.13).

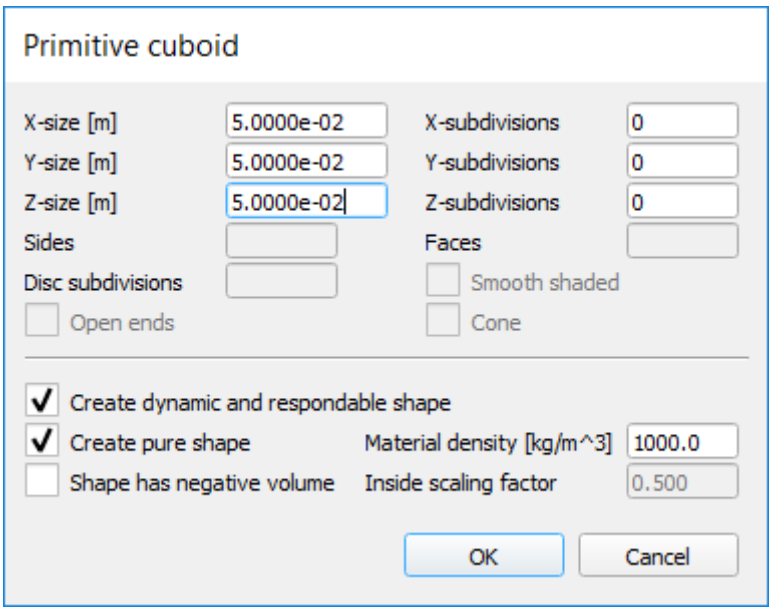

Рисунок 1.13 – Параметри об'єкта, який підходить під розміри захвату

Розмістити можна відразу в захваті, для цього потрібно задати коректне положення. Приклад показано на рисунку 1.14.

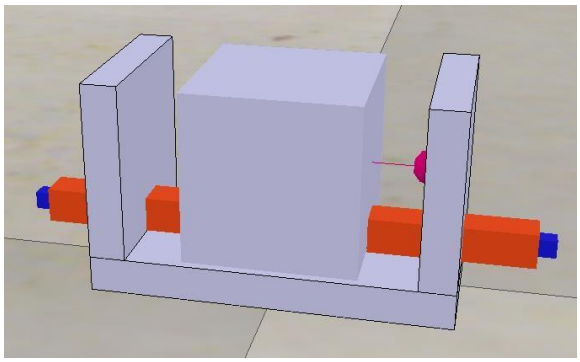

Рисунок 1.14 – Розташування об’єкта в захваті

Далі необхідно перейти до реалізації системи керування через скрипт. Для цього вибираємо елемент «Gripper_Base» і в меню виконуємо «Add→ Associated Child script → Non-threaded child script». Тепер в ієрархії сцени (часто її ще називають деревом моделі) з'явилася нова піктограма праворуч від назви елемента «Gripper_Base». Ця піктограма означає, що дочірній

скрипт прив'язаний до цього елемента. Завжди керуючий скрипт прив'язується саме до базового елемента, тому що в разі перенесення моделі вдасться уникнути проблем з керуванням.

Додати скрипт можна й іншим способом, відкривши в лівій панелі інструмент «Scripts», після активації якого можна додати новий скрипт, натиснувши на «Insert new script» і в контекстному меню обравши «Child script (non-threaded)» зі списку, що випливає. Після цього у списку скриптів з'явиться відповідний запис «Non-threaded child script (unassociated)», який ще потрібно прив'язати: роблять це виділенням скрипту у вікні інструменту «Scripts» і вибору у списку, що випливає, внизу вікна відповідних об'єктів прив'язки (рисунок 1.15).

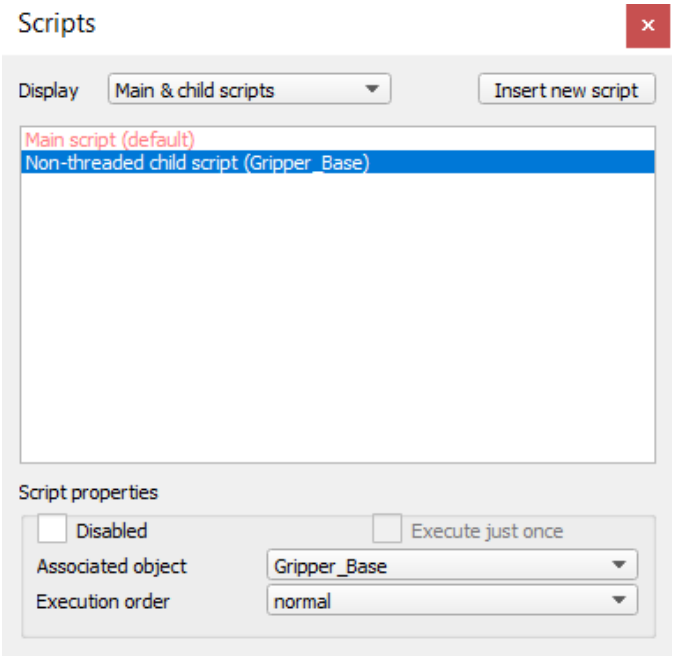

Рисунок 1.15 – Створення нового скрипту і прив'язка до базового елемента через інструмент «Scripts»

Після того як скрипт створено і прив'язано до базового елемента, можна приступити до його редагування. Для цього потрібно виконати подвійне натискання на піктограму праворуч від елемента «Gripper_Base».

Після переходу в режим редагування скрипту можна побачити шаблон скрипту (рисунок 1.16). У цій роботі необхідно написати скрипти у двох розділах: блок ініціалізації та блок керуючого скрипту.

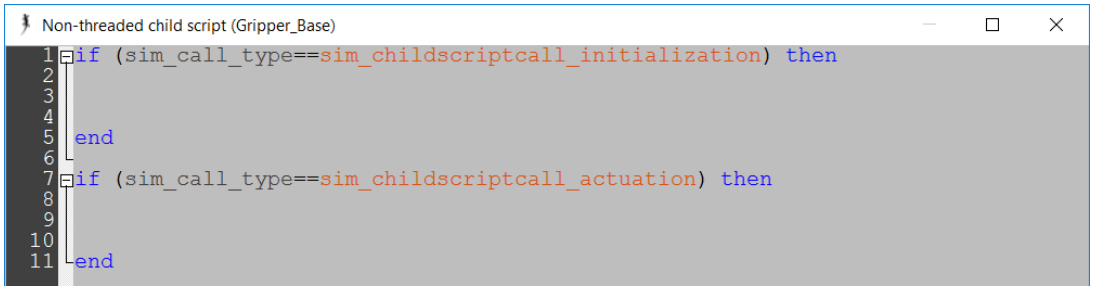

Рисунок 1.16 – Шаблон скрипту

У блоці ініціалізації необхідно оголосити тільки назви моторів і сенсора (а точніше отримати посилання на їхні обробники). Перед захватом об'єкта також необхідно переконатися, що об'єкт перебуває між рухомими елементами захватного пристрою. Щоб реалізувати цю функцію, оголосимо додаткову змінну з таким значенням, що відстань до протилежного захвату дорівнює 1. Тому значення змінної «Detection» можна поставити 0.5, що дорівнюватиме визначенню предмета сенсором до середини захвату (рисунок 1.17).

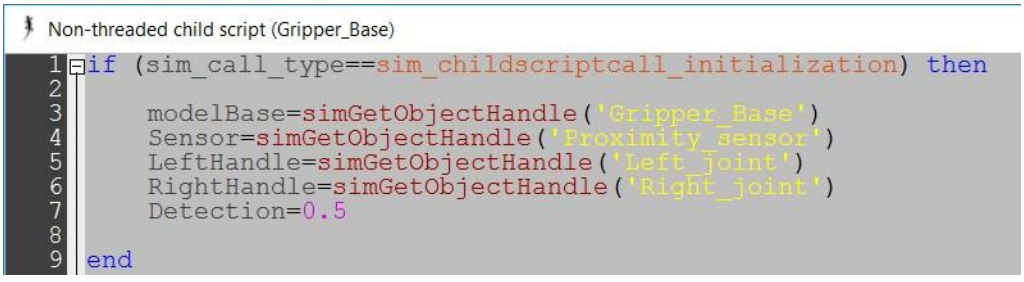

Рисунок 1.17 – Приклад фрагмента скрипту для блока ініціалізації

Переходимо до основного коду керування: вміст цього блока безпосередньо залежить від алгоритму керування. У нашому випадку можна

використовувати такий алгоритм: якщо об'єкт перебуває між рухомими елементами захвату, то активуємо захват, інакше чекаємо. Відповідно в скрипті необхідно реалізувати ухвалення рішення захвату або очікування об'єкта.

Насамперед необхідно зчитувати дані з сенсора під час кожної ітерації. Для зчитування даних із сенсора «Proximity sensor» необхідно скористатися відповідною функцією V-REP (зверніться до довідника регулярних функцій). Ця функція, як видно з документації, на першому місці повертає стан сенсора, що в наведеному прикладі буде записано у змінну «result» (рисунок 1.18, 13 рядок). Також із документації видно, що значення цієї змінної «1» відповідає робочому стану сенсора, тобто в робочій зоні сенсора є якийсь об'єкт, «0» – зворотному. Другим значенням повертається відстань, яка записується у змінну «distance».

```
12 if (sim_call_type==sim_childscriptcall_actuation) then
13     result, distance=simReadProximitySensor(Sensor)
14     if (result>0) and (distance<Detection) and (distance>0) then
15         simSetJointTargetVelocity(LeftHandle, 0.008)
16         simSetJointTargetVelocity(RightHandle, 0.008)
17     else
18         simSetJointTargetVelocity(LeftHandle, 0)
19         simSetJointTargetVelocity(RightHandle, 0)
20     end
21 end
```

Рисунок 1.18 – Фрагмент скрипту, що відповідає за реалізацію алгоритму керування захватом

Тепер необхідно написати логічну частину коду за допомогою розгалуження. Для цього скористаємося умовним оператором «if» і необов'язковою умовою «else», що входить до нього. У 14 рядку на рисунку 1.18 розгалуження виконується трьома умовами – станом сенсора, перевіркою входження в діапазон доступності та вимкнення захвату (нульова відстань до об'єкта).

Якщо всі три умови виконуються, тобто сенсор виявив якийсь об'єкт («result > 0»), відстань до нього менша за значення змінної «Detection», відстань до об'єкта ненульова, то активуємо захват (задаємо швидкість руху моторів). У V-REP допускають керування моторами через задану швидкість і задане положення. У цій роботі будемо використовувати керування за швидкістю. Швидкість задаємо рівною 0.008 м/с.

Якщо ж будь-яка умова з зазначених в «if» не виконується спочатку або потім перестане виконуватися, то необхідно деактивувати захватний механізм, тобто задати нульову швидкість, що вже прописана в умові «else» (інакше).

Наведений у цій роботі алгоритм керування є найпростішим і має низку недоліків, проте з розумінням усього описаного вище можна з легкістю його поліпшити.

Запитання для самоперевірки

1 Які з'єднання доступні в програмі V-REP?

2 Яку умову в коді слід змінити, щоб захват здійснювався незалежно від відстані до об'єкта?

3 Як визначаються шарніри в програмі V-REP?

4 Які датчики можна використовувати в програмі V-REP?

5 Які функції доступні для отримання даних з датчиків?

6 Чому в цій роботі використовують скрипти непотокового типу?

7 Чи можна реалізувати алгоритм захвату за допомогою потокових скриптів?

8 Чим відрізняються 3D-моделі, створені за допомогою V-REP, від моделей, імпортованих з CAD?

9 Чи можна використовувати вкладені умови (гілки)?

10 Чому керуючі скрипти коректно прив'язані до базового елемента механічного захвату?

ПРАКТИЧНА РОБОТА 2

Моделювання роботи маніпулятора

2.1 Мета роботи: моделювання процесу роботи маніпулятора, оснащеного сенсором. Маніпулятор має бути запрограмований на пошук об'єкта заданого кольору, який лежить у робочій зоні маніпулятора. Після виявлення об'єкта маніпулятор має зупинити пошук і підвести кінцеву ланку для захвату об'єкта.

2.2 Завдання

1 Розробити тривимірну модель маніпулятора в будь-якій CAD-системі та імпортувати у V-REP (рисунок 2.1).

2 Задати зв'язки компонентів (задати структуру). Розділити візуальну і фізичну частину моделі на два шари: перша – візуальна, друга – фізична (для розрахунків).

3 Реалізувати алгоритм для виконання пошуку кубика заданого кольору (таблиця 2.1) у скрипті.

4 Написати алгоритм керування для робота, який має підвести захват до об'єкта, зафіксувати його та перенести на задану відстань 20 мм від вихідного положення в будь-якому напрямку.

2.3 Порядок виконання

Промисловий багатоланковий маніпулятор є класичним прикладом динамічної системи в мехатроніці та робототехніці. Розроблення сценарію з моделювання процесу роботи і всю фізику взаємодії компонентів маніпулятора дає змогу найповніше вивчити основні аспекти моделювання великого класу систем.

У цій роботі розглянуто створення симуляції без заглиблення в математичні аспекти моделювання, що дає змогу сконцентруватися на аспектах практичного застосування. Слід пам'ятати, що симуляція не дає точності 100 %, допускаються невеликі відхилення на різних етапах виконання.

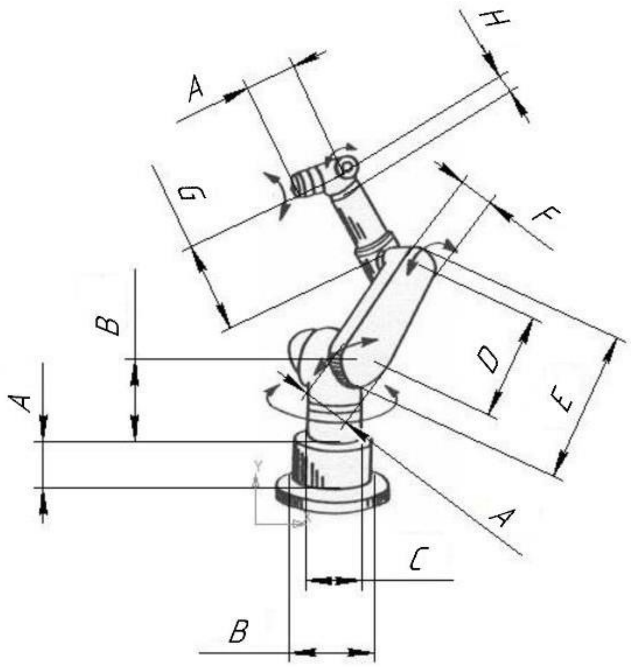

Рисунок 2.1 – Схематичне зображення маніпулятора

Таблиця 2.1 – Варіанти з вихідними даними

Номер	A	B	C	D	E	F	G	H	Колір
1	2	3	4	5	6	7	8	9	10
1	80	140	100	180	200	60	160	30	червоний
2	80	140	100	180	200	60	160	30	синій
3	80	140	100	180	200	60	160	30	зелений
4	160	280	200	360	400	120	380	60	червоний
5	160	280	200	360	400	120	380	60	синій
6	160	280	200	360	400	120	380	60	зелений
7	120	180	140	220	240	100	200	70	червоний
8	120	180	140	220	240	100	200	70	синій
9	120	180	140	220	240	100	200	70	зелений
10	100	160	120	200	220	80	180	50	червоний
11	100	160	120	200	220	80	180	50	синій
12	100	160	120	200	220	80	180	50	зелений
13	130	190	150	230	250	110	210	80	червоний
14	130	190	150	230	250	110	210	80	синій
15	130	190	150	230	250	110	210	80	зелений
16	90	150	110	190	210	70	170	40	червоний

Продовження таблиці 2.1

1	2	3	4	5	6	7	8	9	10
17	90	150	110	190	210	70	170	40	синій
18	90	150	110	190	210	70	170	40	зелений
19	40	70	50	90	100	30	80	15	червоний
20	40	70	50	90	100	30	80	15	синій
21	40	70	50	90	100	30	80	15	зелений
22	60	90	70	110	120	50	100	35	червоний
23	60	90	70	110	120	50	100	35	синій
24	60	90	70	110	120	50	100	35	зелений
25	80	110	90	130	140	70	120	55	червоний
26	80	110	90	130	140	70	120	55	синій
27	80	110	90	130	140	70	120	55	зелений
28	100	130	110	150	160	90	150	75	червоний
29	100	130	110	150	160	90	150	75	синій
30	100	130	110	150	160	90	150	75	зелений

Спочатку необхідно імпортувати вже створену 3D-модель маніпулятора з CAD-системи у V-REP (у роботі зумовлено, що здобувачі здатні створити зазначену модель, можна скористатися безкоштовним онлайн-конструктором «Tinkercad» або аналогічними). Для цього необхідно зберегти збірку маніпулятора з розширенням *.STL. Кожен елемент збірки збережеться окремим файлом. Файли під час збереження у форматі STL зберігають свої координати, і надалі буде зручно з ними працювати вже в програмі V-REP.

Запускаємо V-REP і в головному меню виконуємо «File →Import →Mesh...», вибираємо всі збережені STL файли і натискаємо «Відкрити».

У вікні, що з'явилося, вибираємо пункт «1 unit represents 1 millimeter», що дасть змогу задати коректний масштаб уже в новій системі координат, також може виникнути потреба у зміні орієнтації «Mesh orientation». Після того як усі налаштування коректно задано, залишається натиснути «ОК» (рисунок 2.2).

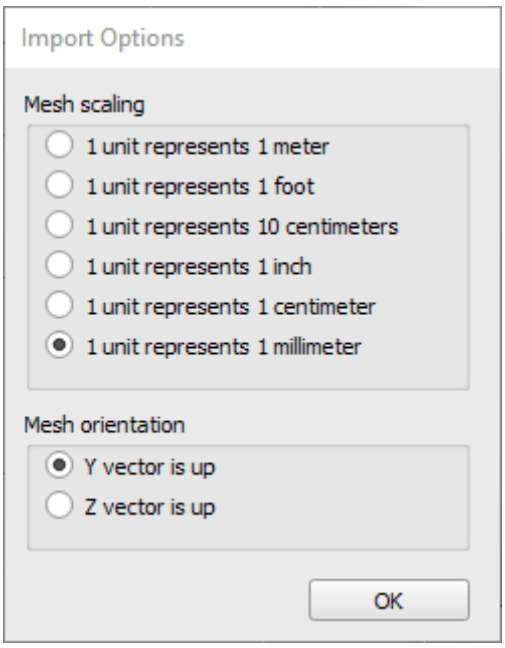

Рисунок 2.2 – Налаштування імпорту файлів

Після імпорту в головному вікні мають відобразитися всі моделі (рисунок 2.3). Також можна побачити, що в ієрархії об'єктів сцени з'явилися нові компоненти, що відповідають кожній частині промислового маніпулятора.

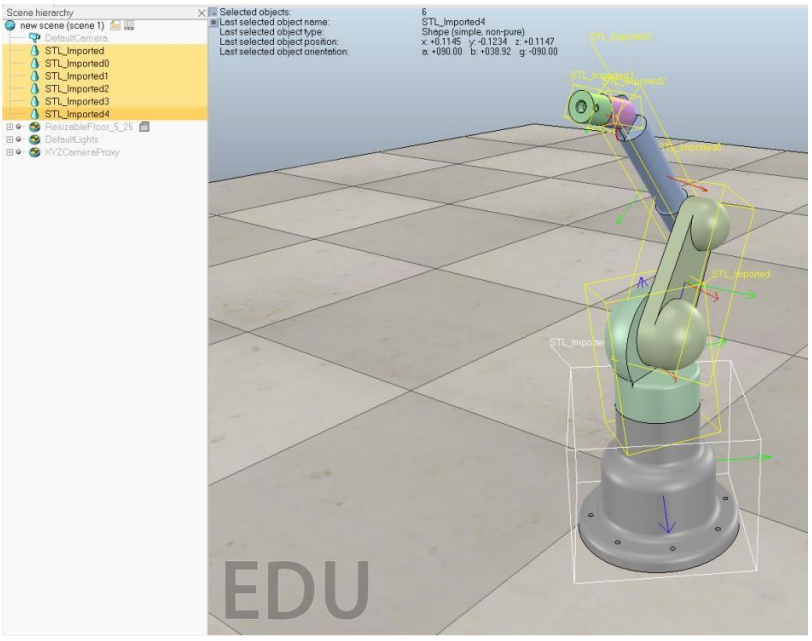

Рисунок 2.3 – Приклад імпортованого з CAD-системи маніпулятора

Звернемо увагу на тип форми кожної компоненти, яку ми імпортували у V-REP. Для ідентифікації типів зручно користуватися піктограмами ліворуч від назви компонента. Наразі всі компоненти мають тип «Складові випадкові форми». Також це можна зрозуміти з того, що тільки цей тип може мати настільки складну геометрію, як зображено на рисунку 2.3.

Надалі, щоб було зручніше працювати з ієрархією сцени, дамо кожній частині маніпулятора назву. Назва має містити інформацію про те, яка це компонента (наприклад близькість до основи, або інша інформація для легкої ідентифікації).

Наявні на цьому етапі елементи можуть бути використані для симуляції, однак цього робити не рекомендується, тому що елементи такого типу не є оптимальними для розрахунку фізичних властивостей. Тому створимо модифіковані копії цих елементів, які будуть використані для моделювання фізики.

Виділяємо кожну компоненту, натискаємо правою кнопкою миші на її назві та у спливаючому меню вибираємо «Add → Convex decomposition of selection...».

Слід звернути увагу, що при зміні параметра «Target nb of triangles of decimated mesh» можна регулювати деталізацію сітки створюваного об'єкта. Рекомендований діапазон значень – від 500 до 1000. Також у контекстному вікні інструменту є низка інших параметрів, які також дають змогу створити оптимізовану сітку.

Виставивши необхідні параметри (рисунок 2.4), натискаємо «ОК». Це необхідно повторити для всіх компонентів маніпулятора. Зауважимо, що з'явилися нові елементи, і піктограми зліва від них відрізняються від піктограм тих елементів, які в нас були.

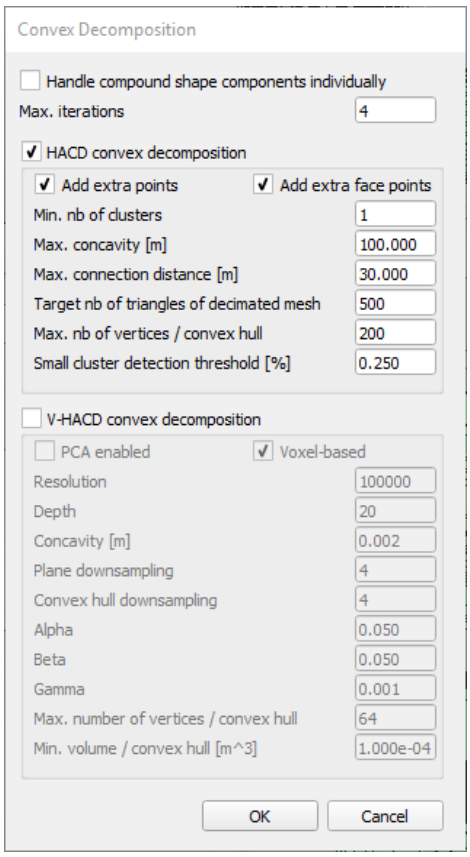

Рисунок 2.4 – Вікно створення копії з оптимізованою сіткою для моделювання динаміки

Далі рекомендовано перейменувати нові компоненти так, щоб у них були назви такі самі, як у вихідних, тільки з додаванням закінчення «_dyn» (у назві допускають тільки латинські літери). Після цього необхідно зв'язати вихідні компоненти з їхніми копіями («_dyn») у такий спосіб, щоб динамічні компоненти були «батьком» для вихідних компонентів. Зробити це можна методом «перетягнути і відпустити», виділивши вихідний компонент в ієрархії сцени (із затиснутою лівою кнопкою миші) і перетягнувши на його копію з динамічними властивостями. Подібну операцію необхідно виконати для всіх компонентів маніпулятора. Результат має вийти схожий на той, що подано на рисунку 2.5.

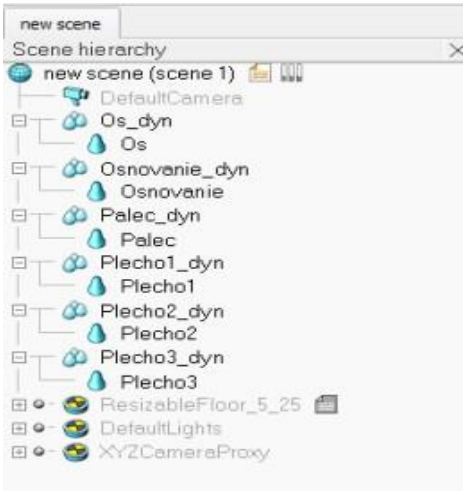

Рисунок 2.5 – Приклад ієрархії сцени маніпулятора

Тепер у сценарії з'явилися об'єкти, які один одного перекривають, і необхідно ці об'єкти рознести на різні шари відображення. Для цього вибираємо елемент з оптимізованою сіткою та активуємо інструмент «Властивості». У вкладці «Common» знаходимо пункт «Camera visibility layers» і встановлюємо значення для першого ряду 0000 0000 і 1000 0000 для другого ряду полів. Аналогічну операцію необхідно виконати з іншими елементами, у яких оптимізована сітка.

У підсумку отримуємо візуально початковий маніпулятор, проте за допомогою інструменту «Шари» можна перемикатися на відображення компонентів іншого типу. На рисунку 2.6 подано відображення тільки елементів з оптимізованою сіткою.

Рисунок 2.6 – Приклад маніпулятора

Тепер залишилося налаштувати елементи так, щоб оптимізовані під динаміку елементи моделювали динаміку, а вихідні компоненти не брали участі в розрахунках, а тільки повторювали поведінку динамічних елементів. Отже, відкриваємо «Властивості» динамічного елемента і натискаємо «Show dynamic properties dialog». Далі необхідно увімкнути функції «Body is Respondable» (розрахунок реакцій, пов'язаних із тілом) і «Body is dynamic» (розрахунок динамічних характеристик). У динамічних параметрах натискаємо «Compute mass and inertia properties for selected...», щоб автоматично розрахувати масу і моменти інерції на основі геометрії (рисунок 2.7), після чого треба ввести щільність матеріалу, і V-REP порахує всі динамічні показники. Вищеописану операцію необхідно повторити для всіх частин маніпулятора, крім основи. Основа маніпулятора є фіксованою і її не слід розглядати як динамічний елемент, але при цьому властивість «Respondable» має бути ввімкнена.

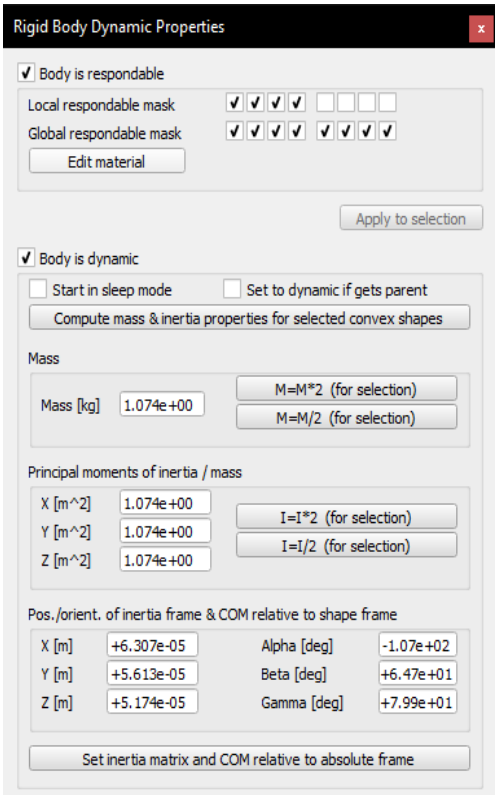

Рисунок 2.7 – Вікно налаштування динамічних властивостей об'єктів

У програмі V-REP є функція встановлення маски взаємодії. За допомогою маски можна повідомити програмі, взаємодію між якими елементами потрібно ігнорувати. Часто буває, що два твердих тіла з'єднані за допомогою шарніра, і в реальній ситуації вони б не діяли одне на одне. Але в програмі V-REP під час моделювання ми можемо отримати грубу сітку, на основі якої і виконують розрахунок. Наслідком цього є те, що сітки двох поруч розташованих елементів стикаються, і для вирішення цієї проблеми треба було б заново створювати дрібнішу сітку, що призвело б до збільшення ресурсів комп'ютера. Використовуючи маски, можна легко вирішити подібні проблеми.

Скористаємося масками об'єктів для виключення з розрахунку взаємодії сусідніх елементів маніпулятора. Для цього відкриваємо властивості основного елемента (основа маніпулятора, зафіксована ланка) і встановлюємо маску, як показано на рисунку 2.8.

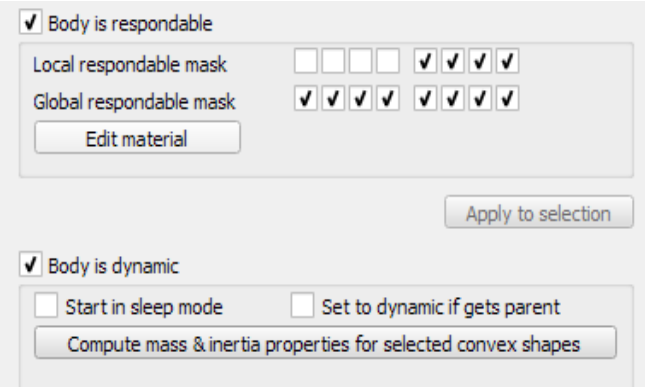

Рисунок 2.8 – Налаштування маски

Далі кожен наступний елемент деревоподібної структури повинен мати локальну маску, відмінну від маски попереднього елемента. Наприклад, «Local respondable mask» наступного за основним елементом буде 11110000, а наступний уже матиме таку саму маску, як і в основи («00001111»), і так далі з чергуванням.

Якщо спробувати запустити отриманий сценарій, то всі деталі маніпулятора мають розсипатися, оскільки відсутні будь-які з'єднання-шарніри.

Для з'єднання частин компонентів маніпулятора потрібно додати в сцену шарніри «Joint» обертального типу «Revolute». Зробити це можна через головне меню «Add → Joint → Revolute», після чого в сцену буде додано новий об'єкт. Шарнір являє собою якусь вісь, відносно якої здійснюватимуться рухи сполучених із шарніром елементів. Виходячи з цього, необхідно точно задати положення й орієнтацію шарнірів у точках відносного обертання ланок маніпулятора. Це завдання є одним зі складних, і в цьому разі можна вдатися до деяких хитрощів. Далі описано один із можливих підходів до цього завдання.

Для позиціонування шарнірів, як і в попередній роботі, скористаємося координатами вже наявних елементів маніпулятора. Затиснувши «Ctrl», виділяємо шарнір і основу маніпулятора, після цього заходимо у вкладку «Position» інструменту «Object/Item Shift» і натискаємо «Apply to selection». Перемикаємося на інструмент «Object/Item Rotate» і у вкладці «Orientation» натискаємо «Apply to selection». Але трапляється так, що шарнір необхідно встановити не в центр деталі за всіма трьома координатами, а обов'язково посередині, у нестандартному місці в пазу або на виступі. З такої ситуації є два виходи: скористатися відносним переміщенням або відносним обертанням. Для цього вибираємо тільки шарнір (після попередньої операції може бути вибрано одразу два елементи) і активуємо інструмент «Object/Item Shift». У вкладці «Translation» вибираємо відносно «Own frame» (координатна система, пов'язана з поточним об'єктом) і вводимо, на яку відстань відносно поточного положення потрібно перемістити виділений об'єкт у відповідних осях, після чого натискаємо «Translate selection». Аналогічно можна обертати шарнір у вкладці «Rotation» інструменту «Object/Item rotate».

Вищеописаний метод вирішує проблему позиціювання шарнірів, але в більшості випадків її можна вирішити ще швидше, якщо скористатися

можливістю поділу елементів із сіткою. Перед тим як зробити це, краще скористатися додатковим «чорновим» сценарієм. Створюємо ще один сценарій і копіюємо туди всі елементи сцени. Далі в чорновій сцені ми маємо виконати позиціювання і після цього скопіювати в робочий сценарій уже коректно позиційований шарнір. Під час копіювання об'єктів з одного сценарію в інший їхні властивості зберігаються.

У чорновому сценарії вибираємо елемент типу «складова опукла форма» і в контекстному меню правої кнопки миші вибираємо «Edit → Grouping/Merging → Ungroup selected shapes». Отже, можна розділити вихідний об'єкт на кілька дуже простих форм і тепер скористатися координатами будь-якої з отриманих форм для позиціювання шарніра. За вже відомим методом задаємо необхідне положення й орієнтацію, після чого копіюємо шарнір із чорнового в робочий сценарій.

У результаті маніпуляцій вийшло задати точне розташування й орієнтацію шарніра. Проробивши аналогічні дії з іншими шарнірами, залишається задати читабельні назви, щоб надалі було зручно з ними працювати.

Тепер, коли всі шарніри на своїх місцях, можна задавати зв'язки між елементами. Для цього створюємо деревоподібну структуру в ієрархії сцени, виставляючи одні об'єкти як дочірні відносно інших. Виставлення ієрархії починається з динамічного об'єкта основи маніпулятора, який є першим батьком, далі йде шарнір, а за шарніром – динамічний об'єкт, найближчий до основи, потім шарнір тощо. На рисунку 2.9 наведено приклад результату, який необхідно отримати після виконання всіх вищеописаних дій (зверніть увагу на ієрархію об'єктів сценарію).

Рисунок 2.9 – Ієрархія об'єктів і розташування шарнірів маніпулятора

Для зручнішого відображення маніпулятора також можна прибрати відображення візуальних властивостей шарнірів. Приступимо до налаштування шарнірів. Вибираємо шарнір в ієрархії сцени і виконуємо команди «Scene object properties → Show dynamic properties dialog». У контекстному вікні «Joint Dynamic Properties» зазначимо три важливі пункти (рисунок 2.10):

– «Motor enabled» – увімкнення/вимкнення мотора;

– «Lock motor when target velocity is zero» – фіксація мотора за швидкості, що дорівнює нулю;

– «Control loop enabled» – ввімкнення/вимкнення циклічного керування.

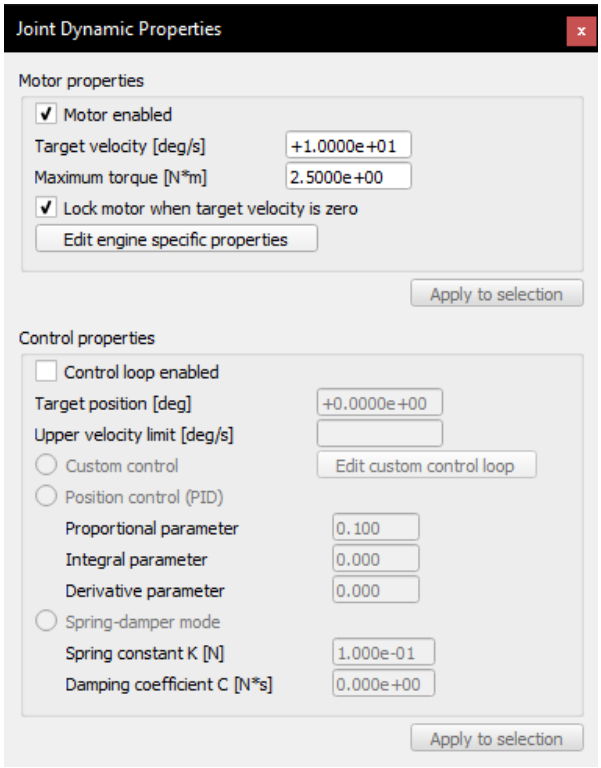

Рисунок 2.10 – Редагування динамічних властивостей шарніра

Ставимо галочки навпроти «Motor enabled» (увімкнути мотор) і «Control loop enabled» (увімкнути циклічне керування) у всіх шарнірах маніпулятора. Наступним кроком необхідно встановити сенсори. Додаємо сенсор через команди «Add → Proximity sensor → Ray type», потім вибираємо цей об'єкт і задаємо потрібне положення та орієнтацію (на ланку маніпулятора). Встановлюємо прив'язку до кінцевої ланки маніпулятора в ієрархії сцени. Цей сенсор визначатиме відстань до об'єкта. Приступимо до його налаштування. Вибираємо сенсор і переходимо «Scene object properties → Show volume parameters». Параметр «Offset» відповідає за відстань, з якої починається відлік, а параметр «Range» відповідає за дальність роботи сенсора.

Додаємо ще один сенсор, який дасть змогу визначати колір об'єкта. Виконуємо команди «Add → Vision sensor → Orthographic type», задаємо

необхідне положення й орієнтацію, потім встановлюємо прив'язку до кінцевої ланки маніпулятора. У цій роботі її налаштовувати не будемо, оскільки налаштування за замовчуванням відповідають усім завданням.

Тепер додамо кубик, який необхідно буде шукати маніпулятору. У головному меню вибираємо «Add → Primitive shape → Cuboid» з необхідними габаритами куба, натискаємо «OK». Виставляємо колір у вікні «Scene object properties → Adjust color → Ambient/diffuse component» і встановлюємо його розташування в межах робочої зони маніпулятора. Для того щоб кубик став видимим для сенсорів, необхідно поміняти його властивості через інструмент «Scene object properties → Common». Необхідно виставити властивості «Collidable», «Measurable», «Detectable» і «Renderable» активними, як показано на рисунку 2.11.

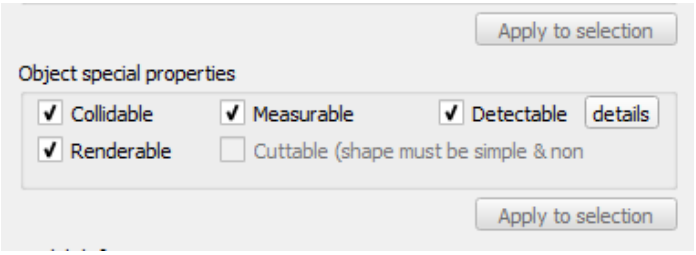

Рисунок 2.11 – Спеціальні властивості об'єктів

Для зручності відображення інформації, одержуваної з сенсора, додамо графік залежності, який відображатиметься під час моделювання. Створюємо графік через головне меню «Add → Graph», заходимо в його властивості та вибираємо «Add new data stream to record». У «Data stream type» вибираємо тип «Various: user-defined», а в «Object/items to record» вибираємо «User data» і натискаємо «OK».

Приступимо до програмної частини. Виділивши основу робота в ієрархії сцени, вибираємо в головному меню «Add → Associated child script → Non threaded». Навпроти назви з'явиться піктограма скрипта

(рисунок 2.12), подвійне натискання на яку відкриває режим редагування скрипту.

Рисунок 2.12 – Піктограма дочірнього скрипту

Далі наведено скрипт керування маніпулятором, який реалізував один зі здобувачів. Скрипт далекий від оптимального як з погляду програмування, так і реалізації алгоритму керування. Наведений скрипт також є універсальним, оскільки багато параметрів залежать від габаритів і особливостей маніпулятора. Алгоритм здійснює пошук і спробу підвести кінцеву ланку до куба червоного кольору. Кубики інших кольорів маніпулятор ігноруватиме (рисунки 2.13, 2.14).

```
if (sim_call_type==sim_childscriptcall_initialization) then
    glaz=simGetObjectHandle("Vision")
    Graph=simGetObjectHandle("Graph")
    rasston=simGetObjectHandle("Proximity")
    check=0
    datas=0
    alt = 0
    bulev = 1
    ctrl = 0
    Osnovanie=simGetObjectHandle('Osi')
    Zveno1=simGetObjectHandle('Ple1')
    Zveno2=simGetObjectHandle('Ple2')
    Zveno3=simGetObjectHandle('Ple3')
    palec=simGetObjectHandle('Pale')
    bu = 0
    nextpositionOsnovanie=0
    nextpositionOsnovanieMax=7
    nextpositionPerTarget=1.1
    nextpositionPer=0.01
    nextPositionPerMax=0.69
    nextpositionZveno3=0
    nextpositionZveno2=0.21
    nextpositionTik=-2.25
end
```

Рисунок 2.13 – Блок ініціалізації кінцевого скрипту

```
if (sim_call_type==sim_childscriptcall_actuation) then
    if (nextpositionPer<nextpositionPerTarget) then
        nextpositionPer=nextpositionPer+0.1
        simSetJointTargetPosition(Zveno1,nextpositionPer+1.15)
        simSetJointTargetPosition(Zveno2,nextpositionPer-0.9)
        simSetJointTargetPosition(Zveno3,nextpositionTik)
    end
    check,datas=simReadProximitySensor(rasston)
    bu,data=simReadVisionSensor(glaz)
    simSetGraphUserData(Graph,'Red_stream',data[12])
    ctrl = data[12]
    if (nextpositionPer > 1.1) then
        if (bulev == 1) then
            if (nextpositionOsn < nextpositionOsnovanieMax) then
                if ((ctrl < 0.63) or (ctrl > 0.77)) then
                 nextpositionOsn = nextpositionOsn + 0.005
                 simSetJointTargetPosition(Osnovanie,nextpositionOsn)
                elseif ((ctrl > 0.63) or (ctrl < 0.77)) then
                 bulev = 0
                 simSetJointTargetPosition(Osnovanie,nextpositionOsn + 0.10)
                end
            end
        end
        if ((bulev == 0) and (check == 1)) then
            if (datas > 0.13) then
                nextpositionZveno2 = nextpositionZveno2 - 0.01
                nextpositionTik = nextpositionTik + 0.01
                simSetJointTargetPosition(Zveno2,nextpositionZveno2)
                simSetJointTargetPosition(Zveno3,nextpositionTik)
            end
        end
    end
end
```

Рисунок 2.14 – Блок керуючого коду

Запускаємо симуляцію (рисунок 2.15).

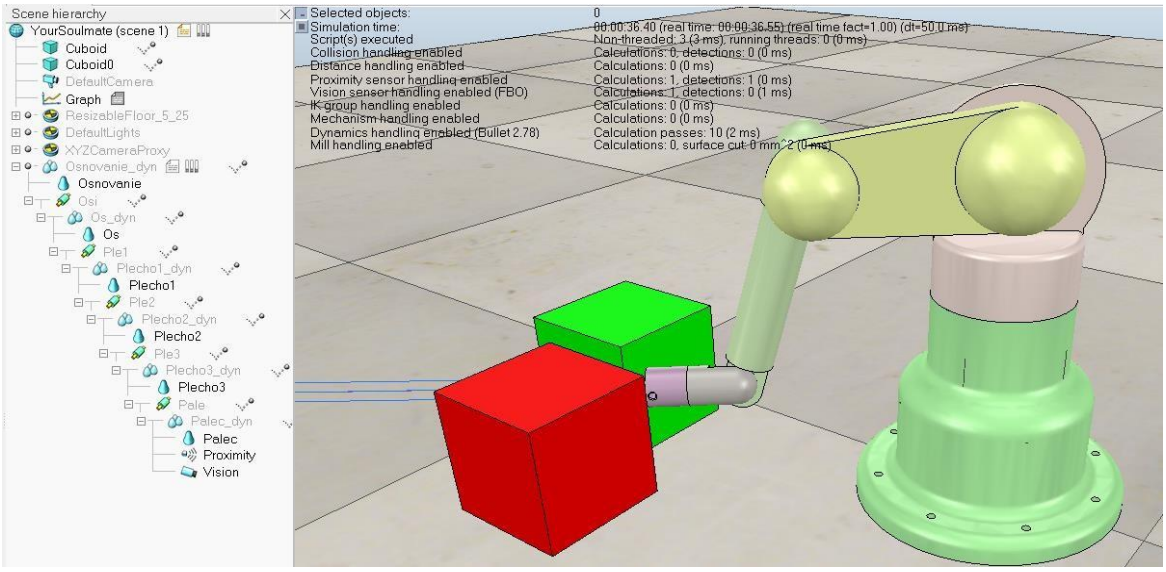

Рисунок 2.15 – Результат моделювання маніпулятора

Щоб зберегти маніпулятор як цілу модель, достатньо у властивостях базового об'єкта (основа маніпулятора) додатково встановити значення «Object is model base» (рисунок 2.16).

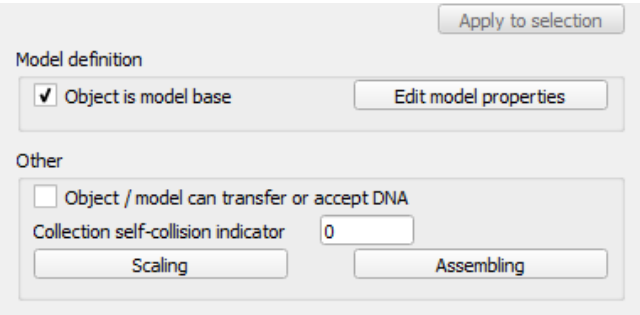

Рисунок 2.16 – Вікно властивостей основи маніпулятора,
розділ «Model definition»

Запитання для самоперевірки

1 Чи можна використовувати потоковий тип скрипту для керування рухом колісного робота?

2 Як змінити колір об'єкта? У яких випадках колір може бути проігнорований фотосенсорами програми V-REP?

3 Скільки значень повертає функція зчитування даних з фотосенсора V-REP?

4 Чи можна виконати попередню обробку даних із сенсора перед виведенням на графік?

ПРАКТИЧНА РОБОТА 3

Моделювання роботи автономного мобільного робота

3.1 Мета роботи: моделювання спрощеного мобільного робота і реалізація алгоритму керування в автономному режимі. Модель робота повинна мати мінімальні характеристики колісного робота. Алгоритм керування має забезпечувати рух замкненим контуром, ґрунтуючись на даних із датчиків.

3.2 Завдання

1 Розробити спрощену модель колісного робота в програмі V-REP з використанням інструменту «Primitive Shape» або розробити аналогічну модель у будь-якій CAD-системі та імпортувати у V-REP.

2 Задати структуру моделі (визначити механічні з'єднання), винести кожен тип елементів на окремий шар. Задати необхідні властивості механічних елементів.

3 Додати в модель кілька сенсорів і написати алгоритм керування, що дає змогу роботу пересуватися замкненим контуром. Розмір робота у вихідному положенні не може перевищувати в довжину 20 см і в ширину 6 см. Опис траси: мінімальна ширина траси – 30 см, максимальна – 50 см, максимальний кут повороту – 180°. Траса огороджена стіною, висота якої становить 20 см.

3.3 Порядок виконання

У роботі запропоновано змоделювати процес роботи найпоширенішого типу мобільних роботів – колісного. Особливості моделювання роботів цього типу зумовлені характером використовуваних приводів і алгоритмів, які відповідають за ефективне виконання поставлених завдань. У рамках цієї роботи необхідно не тільки змоделювати

механічну складову системи, а й реалізувати алгоритм керування, оскільки алгоритм керування є невід'ємною частиною будь-якої сучасної робототехнічної системи.

Під час роботи має бути використана бібліотека моделювання фізики «Bullet Physics», також можна використати бібліотеку ODE. Для реалізації алгоритму керування необхідно скористатися регулярними API функціями, подані в пакеті «sim», опис яких можна знайти на сайті виробника програмного забезпечення в розділі «Resources».

Розглянемо найпростіший варіант реалізації автономного робота на чотирьох колесах. Усі чотири колеса незалежні, і на кожному з них є шарнір.

Спираючись на метод, описаний у практичній роботі 1 (за допомогою інструменту «Primitive Shape»), створюємо прямокутний паралелограм і чотири циліндри заданої товщини (дивитися габаритні розміри в описі до роботи). Після цього необхідно задати положення й орієнтацію для всіх компонентів, як показано на рисунку 3.1. Компоненти типу «Прості форми» уже мають оптимальну сітку для моделювання динаміки, тому залишається тільки переконатися, що динамічні властивості встановлено коректно.

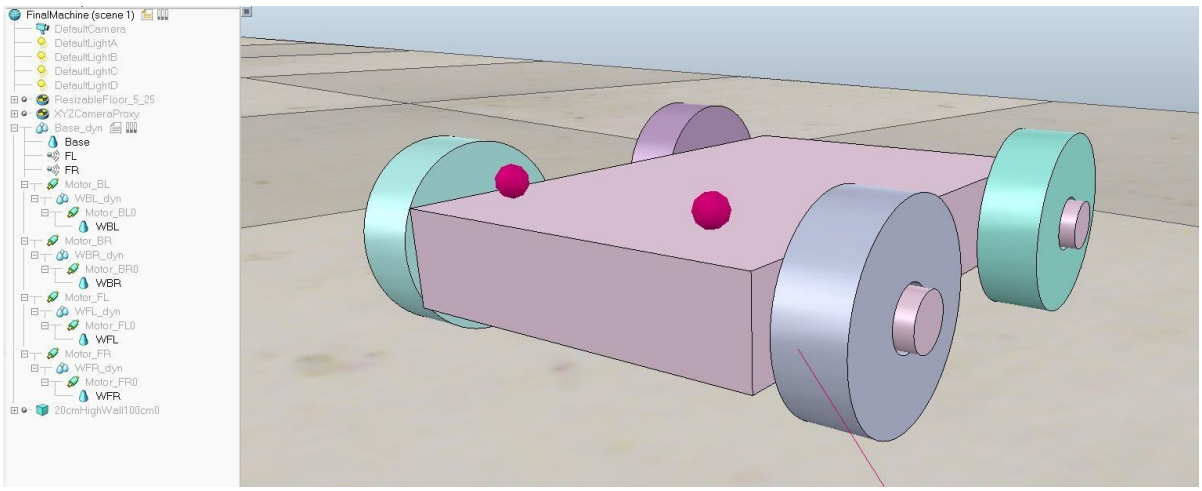

Рисунок 3.1 – Приклад автономного колісного робота

Також можна за методикою, наведеною в практичній роботі 2, вдатися до допомоги будь-якої CAD-системи та імпортувати моделі у V-REP. В останньому випадку необхідно створити копії елементів з оптимізованою сіткою (детально розглянуто в попередній роботі) на основі імпортованих CAD-моделей. Також не забуваємо, що в елементах з оптимізованою сіткою необхідно задати всі динамічні властивості і поставити прив'язки динамічних елементів до вихідних елементів.

Далі додаємо в сценарій шарніри з одним обертальним ступенем свободи і задаємо їм розташування (та орієнтацію) за центрами коліс, потім встановлюємо деревоподібну ієрархію об'єктів, тим самим задаючи зв'язки елементів моделі. *Зверніть увагу*, що в цьому випадку «основою» робота є корпус, до якого всі інші компоненти прив'язуються.

Для своєчасного повороту колісного робота під час руху замкненим контуром достатньо скористатися двома сенсорами. Додаємо два сенсори типу «Proximity sensor» у сценарій симуляції і задаємо їм положення на передній частині корпусу колісного робота, також встановлюємо орієнтацію під кутом 45° у різні боки від робота і вперед. Виставляємо прив'язки сенсорів в ієрархії сцени, у такий спосіб закріплюючи сенсори в поточному положенні відносно корпусу.

Залишилося розробити алгоритм керування і реалізувати скрипт. Дочірній скрипт необхідно прив'язати до основи робота, тобто корпусу колісного робота.

Далі наведено скрипт, розроблений одним зі здобувачів, який реалізує простий циклічний рух по замкненому контуру (рисунки 3.2-3.4).

```
if (sim_call_type==sim_childscriptcall_initialization) then
    jointHandle1=simGetObjectHandle('Motor_BL')
    jointHandle2=simGetObjectHandle('Motor_BR')
    jointHandle3=simGetObjectHandle('Motor_FL')
    jointHandle4=simGetObjectHandle('Motor_FR')
    position1=simGetJointPosition(jointHandle1)
    position2=simGetJointPosition(jointHandle2)
    position3=simGetJointPosition(jointHandle3)
    position4=simGetJointPosition(jointHandle4)
    simAddStatusbarMessage('position1='..position1)
    simAddStatusbarMessage('position1='..position2)
    simAddStatusbarMessage('position1='..position3)
    simAddStatusbarMessage('position1='..position4)
    Lsensor=simGetObjectHandle('FL')
    Rsensor=simGetObjectHandle('FR')
    Lres=0
    Rres=0
    Rdif=0
    Ldif=0
    Speed=10
end
```

Рисунок 3.2 – Скрипт блока ініціалізації

```
if (sim_call_type==sim_childscriptcall_initialization) then
    jointHandle1=simGetObjectHandle('Motor_BL')
    jointHandle2=simGetObjectHandle('Motor_BR')
    jointHandle3=simGetObjectHandle('Motor_FL')
    jointHandle4=simGetObjectHandle('Motor_FR')
    position1=simGetJointPosition(jointHandle1)
    position2=simGetJointPosition(jointHandle2)
    position3=simGetJointPosition(jointHandle3)
    position4=simGetJointPosition(jointHandle4)
    simAddStatusbarMessage('position1='..position1)
    simAddStatusbarMessage('position1='..position2)
    simAddStatusbarMessage('position1='..position3)
    simAddStatusbarMessage('position1='..position4)
    Lsensor=simGetObjectHandle('FL')
    Rsensor=simGetObjectHandle('FR')
    Lres=0
    Rres=0
    Rdif=0
    Ldif=0
    Speed=10
end
```

Рисунок 3.3 – Скрипт блока ініціалізації

```
22 if (sim_call_type==sim_childscriptcall_actuation) then
23     Lres,Ldist=simReadProximitySensor(Lsensor)
24     Rres,Rdist=simReadProximitySensor(Rsensor)
25     Rdif=0
26     Ldif=0
27     if (Ldist<Rdist) then
28         Rdif=-5
29         Ldif=5
30     end
31     if (Ldist>Rdist) then
32         Rdif=5
33         Ldif=-5
34     end
35     simSetJointTargetVelocity(jointHandle1, Speed+Ldif)
36     simSetJointTargetVelocity(jointHandle2, Speed+Rdif)
37     simSetJointTargetVelocity(jointHandle3, Speed+Ldif)
38     simSetJointTargetVelocity(jointHandle4, Speed+Rdif)
39 end
40
```

Рисунок 3.4 – Скрипт основного блока керування

Далі необхідно створити замкнену трасу для забезпечення умов симуляції. Для цього активуємо «Model browser→infrastructure→walls» і вибираємо стіни висотою 20 мм. За допомогою вже наявних моделей стін можна створити трасу, що відповідає умовам практичної роботи (на рисунку 3.5 показано один із можливих варіантів).

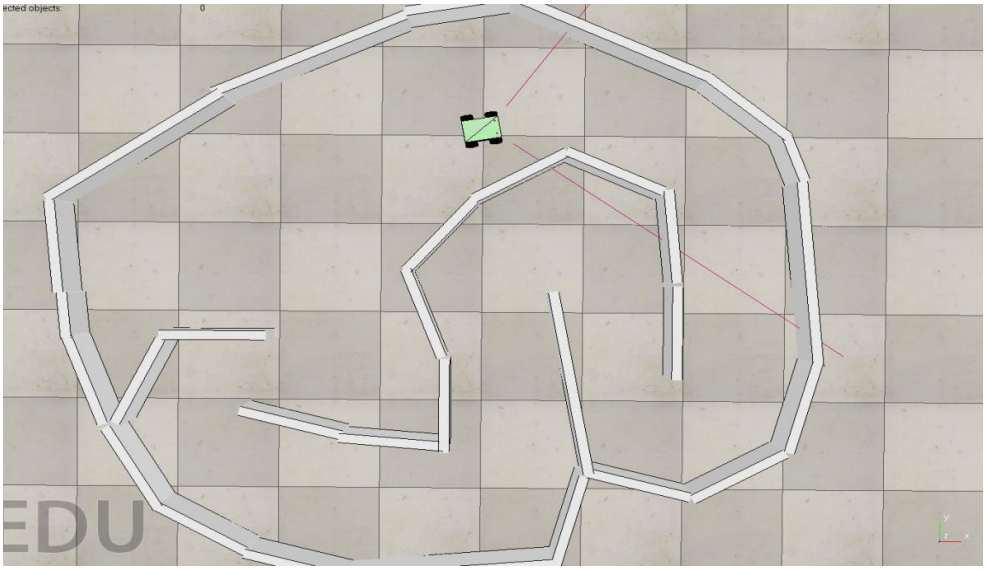

Рисунок 3.5 – Приклад замкненої траси для колісного робота

Запускаємо симуляцію. Колісний робот має проїжджати задану трасу, не зупиняючись і не застрягаючи. Можливо, що через некоректно виставлену масу і встановлену високу швидкість у стартовій позиції робот може не вписуватися в повороти або зовсім перевернутися під час початку руху.

Існує досить поширений метод руху заданою траєкторією, який добре описує деякі реальні системи. Подібний приклад наведено в навчальному посібнику, де повністю розкрито можливості реалізації рухомих систем за заданою траєкторією.

Запитання для самоконтролю

1 Навіщо потрібен «Control Loop» і чому в симуляції його не використовують?

2 Чому вибрано керування через зміну швидкості обертання коліс? Які альтернативні способи керування є?

3 Чи можна використовувати «Vision Sensor» замість «Proximity sensor» у цій симуляції?

4 Чи можна зберегти колісного робота з поточною симуляцією як модель і додати в «Model browser»?

СПИСОК ЛІТЕРАТУРИ

1 Цвіркун Л. І., Грулер Г. Робототехніка та мехатроніка: навч. посіб. / за заг. ред. Л. І. Цвіркуна ; М-во освіти і науки України, Нац. гірн. ун-т. Вид. 3-тє, перероб. і доп. Дніпро: НГУ, 2017. 224 с.

2 Документація до програмного забезпечення ABB Robotic Studio. URL: http://developercenter.robotstudio.com/.

3 Офіційний сайт розробників Gazebo. URL: http://gazebosim.org.

4 Офіційний сайт розробників V-REP. URL: http://coppeliarobotics.com/.

5 Документація відкритої бібліотеки планування руху. URL: http://ompl.kavrakilab.org.

6 Документація для розробників мовою Lua. URL: http://www.lua.ru/doc/.

7 Список регулярних функцій V-REP з докладним описом. URL: http://www.coppeliarobotics.com/helpFiles/en/apiFunctionListCategory.htm.

8 Методичний посібник з додержання вимог нормоконтролю у студентській навчальній звітності. Студентська навчальна звітність. Текстова частина (пояснювальна записка). Загальні вимоги до побудови, викладення та оформлення /за заг. ред. Л. М. Козара. Харків : УкрДАЗТ, 2014. 58 с.

ДОДАТОК А
Основи роботи в програмі V-REP

Структура та ключові особливості V-REP

Платформа V-REP має низку особливостей, які дають розробнику широкі можливості для створення симуляцій. Як основний компонент V-REP можна виділити технологію вбудованих скриптів, які виконують функції контролерів у симуляціях. При цьому наявність можливості прив'язки окремих скриптів до компонентів робота дає змогу реалізувати чітку ієрархію, забезпечуючи портативність і масштабованість.

Будь-яка симуляція в V-REP за замовчуванням має основний скрипт, який вкрай не рекомендують змінювати, оскільки цей скрипт вирішує загальні завдання забезпечення коректності даних під час виконання симуляції. Наприклад, він викликає різні підсистеми для моделювання кінематики і динаміки механічних елементів системи. З основного скрипту також виконується виклик дочірніх скриптів каскадним способом, але цю функцію можна відключити.

Дочірні скрипти, на відміну від основного, прикріплюються до окремих компонентів моделі. Вони є невід'ємною складовою сценарію симуляції і виконуються під час кожної ітерації моделювання, як і основний скрипт. Слід також пам'ятати, що скрипти є виконуваними файлами і не потребують попередньої компіляції. Крім того, дочірні скрипти можуть бути потоковими або непотоковими. Розробники V-REP рекомендують за можливості використовувати непотокові скрипти, проте в деяких завданнях потокові скрипти краще вирішують поставлені завдання. Слід також зауважити, що симуляція – це ітеративний процес, тобто перерахунок параметрів модельованої системи здійснюється через постійний проміжок часу (крок моделювання) ітеративно. У V-REP за замовчуванням використовується 50 мс.

Потокові скрипти – скрипти, виконання яких триватиме під час наступної ітерації. Такі скрипти виконуються один раз від початку до кінця, але також можна відключити цю функцію, щоб не було переривання після першої ітерації.

Непотокові скрипти – скрипти, виконання яких починається на початку при кожній наступній ітерації, при цьому здійснюється повне передавання керування симуляцією в ці скрипти. Якщо з якихось причин скрипт не завершився, і керування не передалося назад на основний скрипт, то симуляція переривається. Такі скрипти викликаються основним скриптом два рази за кожен крок симуляції: під час активації та отримання сенсорної інформації.

Доступність. Поширюється платформа V-REP умовно безкоштовно. Для наукових досліджень і ведення освітньої діяльності ця програма має окремі версії. У разі ж використання в комерційних проєктах необхідно придбати ліцензію. Програма V-REP є повністю кросплатформною (не залежною від використовуваної операційної системи).

REP підтримує модель взаємодії Клієнт-Сервер. Платформа V-REP виступає сервером, на якому запущена симуляція, а керування здійснюється з клієнтського боку. За такої схеми взаємодії клієнтом може виступати як інше програмне забезпечення, так і будь-який пристрій із підключенням до сервера (V-REP), наприклад клієнтом може бути робот.

API-функції. Пропонують широкий набір готових API-функцій мовою Lua, використання яких робить процес створення сценарію симуляції швидким і легким. Широкі можливості також надають API-функції для створення сценаріїв симуляції іншими мовами програмування (Remote API), а саме C/C++, Python, Java, Matlab, Octave і Lua.

У V-REP є чотири різні фізичні рушії моделювання: ODE, Bullet, Vortex і Newton. Кожен із них має вищу точність моделювання в певних

завданнях, але всі вони добре вирішують загальні завдання моделювання роботів.

Вбудовані модулі для спеціалізованих завдань. Є потужний і гнучкий модуль обчислення зворотної і прямої кінематики роботів. Цей пакет прийнятний для вузькоспеціалізованих завдань при роботі з маніпуляторами. Вбудований модуль для швидкої перевірки зіткнень об'єктів дає змогу вирішувати низку завдань, пов'язаних із безпекою, максимально ефективно. Також є схожий модуль, який швидко і точно розраховує мінімальну відстань між об'єктами.

Існують різні типи датчиків, зокрема найбільш часто застосовувані датчики наближення (аналог ультразвукових датчиків). Також є фото-відеодатчики, які аналізують видимі властивості різних компонентів симуляції.

Планування руху. У програмі V-REP планування руху виконують із використанням відкритої бібліотеки планування руху («Open Motion Planning Library», скорочено OMPL) [6].

Запис і візуалізація даних. Можливості для візуалізації даних були одними з ключових для розробників, і починаючи з першої версії у V-REP є весь необхідний інструментарій для запису та візуалізації будь-яких типів даних.

Вбудований редактор сітки. Є редактор сітки, який дає змогу вносити зміни в сітку моделі, на основі якої розраховують фізичні параметри механічних елементів симуляції. Також підтримується кілька спеціальних режимів редагування сітки об'єкта.

Імпорт та експорт даних. Завантаження і вивантаження даних із V-REP виконують через меню: для цих завдань є спеціальний інструмент в основному меню. Також програма може зчитувати дані через з'єднання з іншим пристроєм.

Повнофункціональна ієрархія моделей у сцені. Цей інструмент дає змогу платформі V-REP реалізувати взаємозв'язки компонентів моделі вкрай зручно та з широкими можливостями масштабування.

Це не все, чим вирізняється V-REP з-поміж інших програм моделювання, однак розуміння цих особливостей дасть змогу надалі працювати з платформою та доповнювати загальну інформацію більш детальними та спеціалізованими можливостями програми.

Інтерфейс програми. Інтерфейс програми V-REP поділений на кілька частин залежно від призначення та реалізований у вікні з графічним інтерфейсом: графічне вікно програми використовують для керування всіма вбудованими інструментами (рисунок А.1).

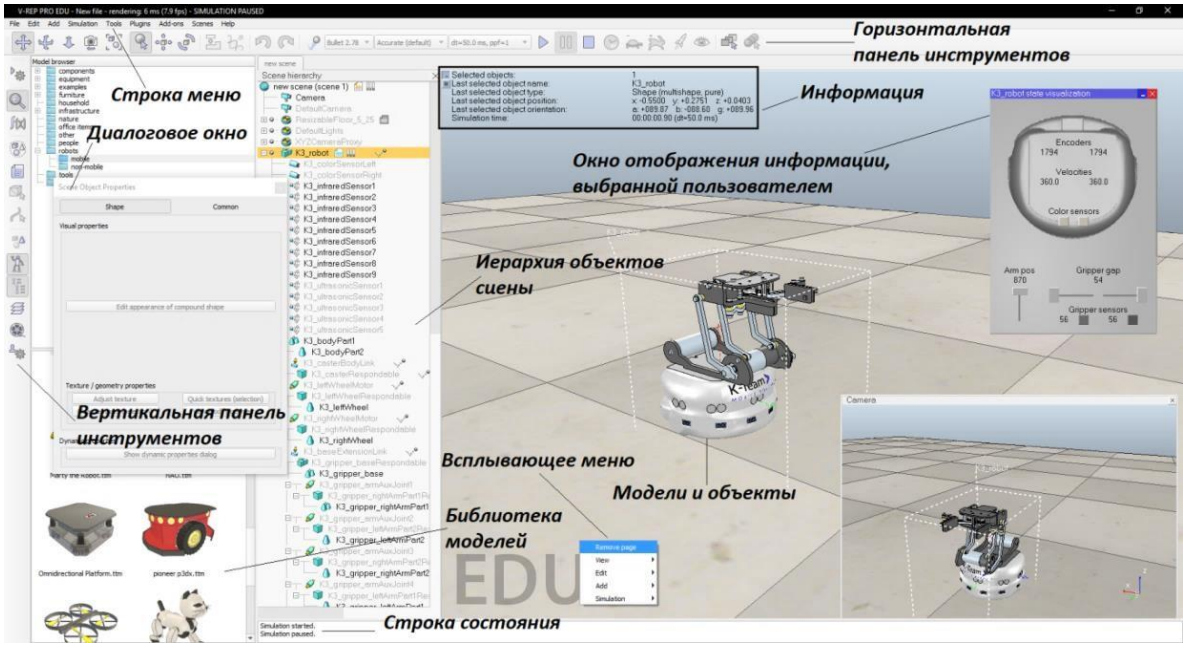

Рисунок А.1 – Вікно застосунку V-REP

Також слід згадати вікно консолі, яке можна спостерігати під час запуску додатка, але за замовчуванням це вікно приховується одразу ж. За необхідності можна налаштувати V-REP на постійне відображення консолі, викликавши «User settings» («Користувацькі налаштування») за допомогою

першої кнопки у вертикальній панелі інструментів. У консольному вікні відображуються плагіни, що завантажуються, і їхні процедури, потрібні тільки під час роботи з плагінами. Тут ми не розглядаємо роботу з плагінами, однак за необхідності можна звернутися до документації з роботи в V-REP на сайті розробників.

На рисунку А.1 наведено інтерфейс програми V-REP із доволі великим набором активних інструментів, однак треба розуміти, що одночасна активація всіх інструментів призведе до їхнього взаємного перекриття.

Далі наведено основні складові графічного інтерфейсу програми V-REP з коротким описом.

Основне меню програми містить різні розділи, у яких є інструменти для редагування наявних і додавання нових об'єктів у сценарій симуляції. Частину інструментів із головного меню також продубльовано у спливаючому контекстному меню V-REP.

Панелі інструментів. Найчастіше використовувані інструменти винесені сюди для швидкого доступу. На рисунках А.2, А.3 наведено кнопки, розташовані на панелях інструментів, з їхніми назвами. Кожен інструмент із цих панелей може перебувати в активному і пасивному режимах. При цьому активація здійснюється простим натисканням на піктограму інструменту.

Бібліотека моделей. За замовчуванням цей інструмент перебуває в активному режимі, але може бути деактивований натисканням на піктограму. В активному режимі цей інструмент відображує структуру папок і моделей, що містяться в поточній папці. Моделі з бібліотеки можна легко додати в сценарій симуляції шляхом перетягування в робоче вікно.

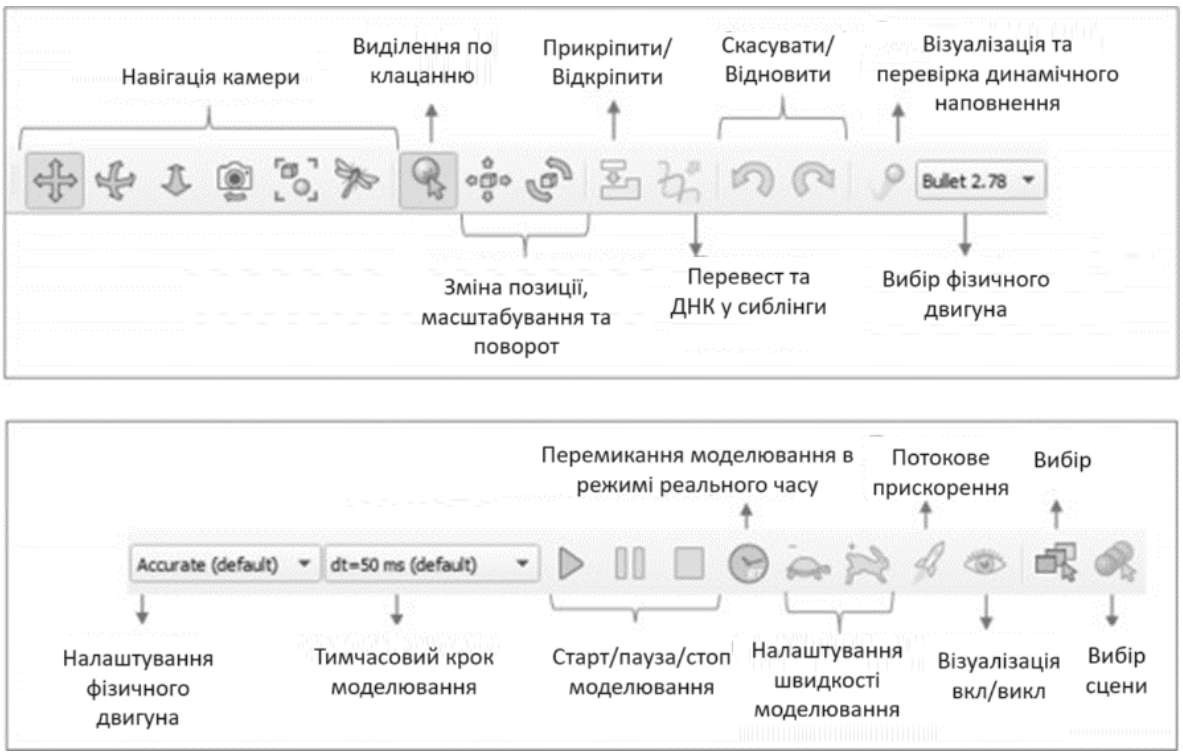

Рисунок А.2 – Горизонтальна панель інструментів V-REP

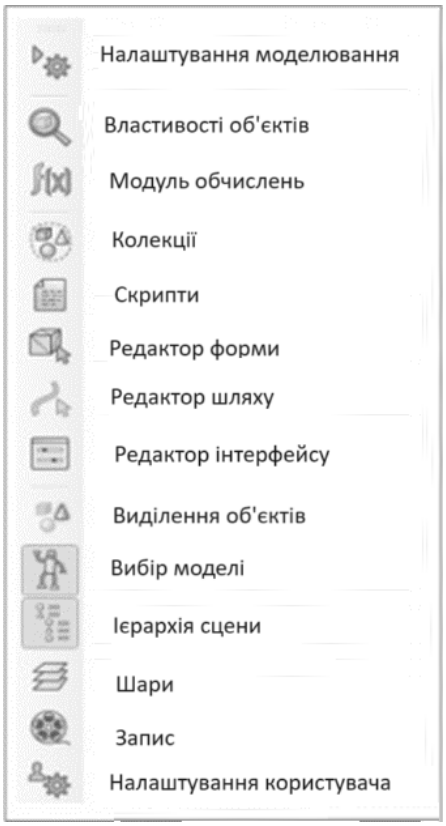

Рисунок А.3 – Вертикальна панель інструментів V-REP

Ієрархія об'єктів сценарію. Деревоподібна ієрархія відображує структуру об'єктів і їхніх компонентів. Їх можна змінювати або додавати нові елементи. Ієрархія об'єктів є ключовим елементом, який задає основні зв'язки між об'єктами сценарію. Також властивості будь-якого об'єкта можна відкрити подвійним натисканням лівої кнопки миші на піктограму ліворуч від назви об'єкта в ієрархії сцени. Також можна вибрати спочатку необхідний елемент за допомогою одного натискання на елемент і потім активувати інструмент «Властивості об'єктів» («Scene Object Properties»). Для зміни назв компонентів достатньо навести курсор на поточну назву і двічі натиснути на ліву кнопку миші, після чого ввести нову назву і завершити натисканням кнопки введення («Enter»). Слід урахувати, що допускають використання тільки літер латинського алфавіту. Однією з ключових функцій, реалізованих за допомогою ієрархії об'єктів, є взаємозв'язок компонентів системи. Для цієї функції доступно два способи. Перетягуючи один об'єкт на інший, можна встановити відносини зв'язку між ними (зробити одного з них «батьком» іншого). Аналогічну дію можна виконати через спливаюче меню, для цього потрібно вибрати спочатку «дочірній» елемент, затиснути клавішу «Shift» і потім вибрати «батьківський», правою кнопкою миші відкрити спливаюче контекстне меню і вибрати «Edit» і «Make last selected object Parent».

Основна сторінка (основне вікно). Кожен сценарій симуляції може містити до восьми сторінок, які у свою чергу можуть включати необмежену кількість областей відображення. Подібний поділ дає змогу максимально ефективно використовувати графічний інтерфейс для виведення різних типів даних одночасно.

Області відображення. Області відображення являють собою області інтерфейсу, призначені для розділеного виведення даних сценарію і візуалізації, отриманих за допомогою віртуальних відеокамер, сенсорів різного типу та іншої заданої користувачем інформації.

Інформація. Блок з інформацією виводять для окремих моделей, компонентів або сценарію залежно від вибору користувача, він показує різні параметри.

Глобальна система координат. Позначення орієнтації завжди присутнє в правому нижньому кутку. Також у кожного об'єкта у V-REP є локальна система координат.

Вікно відображення інформації, обраної користувачем, – вікно, що налаштоване користувачем і відображує необхідну інформацію або діалог з користувачем.

Спливаюче меню. З'являється при натисканні правою кнопкою миші, має широкі можливості і дублює частину інструментів і функцій із панелей інструментів.

Діалогове вікно. Відображує властивості об'єктів із можливістю їх редагування, може бути каскадно розширене в деяких випадках. Вікно відкривається під час активації окремих інструментів. Параметри, що відображуються в цьому вікні, залежать від типу обраного елемента і призначення інструменту.

Рядок стану. Область інтерфейсу використовують для виведення текстової інформації під час виконання симуляції.

Об'єкти порожньої симуляції. При створенні нового сценарію симуляції в програмі V-REP вже є кілька об'єктів, які додають за замовчуванням: джерело світла, підлога зі змінним розміром і камери. Звісно, їх можна видалити, але майже завжди вони необхідні для створення симуляції, також іноді виникає потреба в їхньому налаштуванні. Для зміни розміру поверхні підлоги достатньо виділити її в ієрархії сцени і в користувацькому інтерфейсі, що з'явився, переміщати повзунок доти, доки не вийде потрібний результат (рисунок А.4).

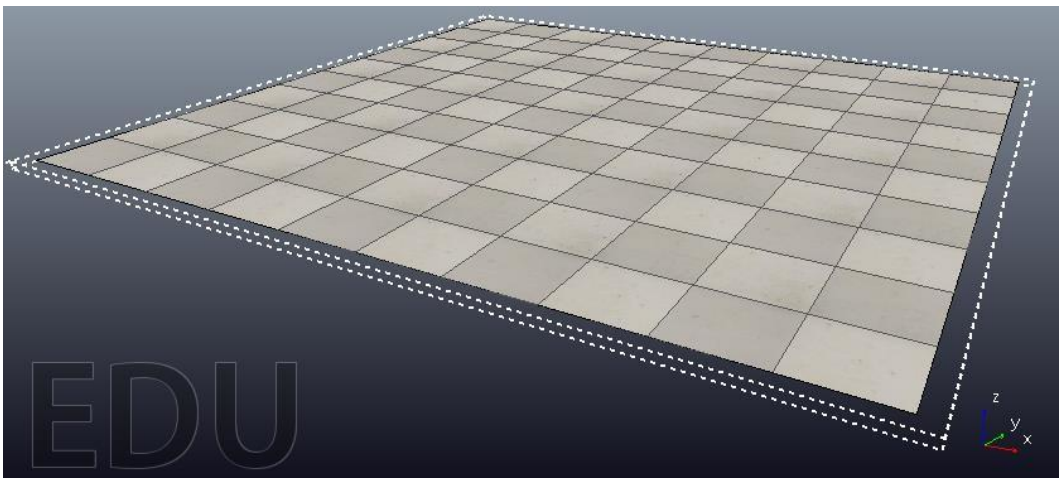

Рисунок А.4 – Поверхня підлоги зі змінним розміром

Огляд інструментів V-REP

Зміна положення, орієнтації та масштабування. Це одні з основних інструментів, необхідних для введення початкових умов симуляції. За допомогою інструменту «Object/Item Shift» (рисунок А.5, ліворуч) можна задавати положення об'єктів у сцені та масштаб (зменшувати або збільшувати об'єкт). Інструмент «Object/Item Rotation» (рисунок А.5, праворуч) дає змогу задавати орієнтацію об'єкта в просторі.

Рисунок А.5 – Інструменти «Положення» та «Орієнтація»

Зміна властивостей об'єкта. Інструмент «Властивості» є не менш часто використовуваним при створенні симуляцій у програмі V-REP і дає змогу задавати різні властивості об'єктів і компонентів симуляції. Після активації інструменту виводиться контекстне меню, яке має дві вкладки. У першій вкладці наведено перелік властивостей, який є спеціалізованим і

залежить від типу обраного об'єкта. У другій вкладці контекстного меню інструменту наведено загальні властивості, які в більшості об'єктів схожі.

Запуск і зупинка симуляції. Для цих завдань у горизонтальній панелі є набір кнопок на панелі інструментів (рисунок А.6), що дають змогу запускати симуляцію (рисунок А.6, зліва), зупиняти (рисунок А.6, посередині) і завершувати симуляцію (рисунок А.6, справа).

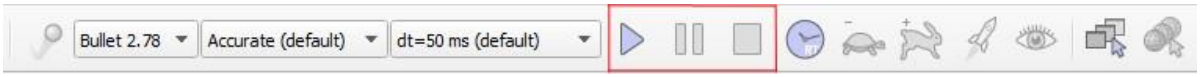

Рисунок А.6 – Інструменти запуску, зупинки та завершення симуляції

Також на рисунку А.6 ліворуч від інструменту запуску симуляції показано налаштування вирішувача (бібліотеки для моделювання законів фізики), де можна змінити крок моделювання, режим моделювання і використовуване ядро фізичного рушія.

Керування візуальними властивостями. У програмі V-REP реалізовано механізм пошарового поділу всіх візуальних об'єктів. Цей функціонал необхідний, тому що в програмі потрібно використовувати поєднання компонентів різних типів, і в таких випадках стає неможливою подальша робота з об'єктами. Наприклад, наявність однакових компонентів, які мають однакове положення, але моделюють різні властивості, призводить до того, що вони стають візуально невиразними. Тому необхідно користуватися інструментом «Шари». Рекомендований принцип рознесення на шари: компоненти одного типу мають бути на окремому шарі (наприклад усі шарніри). Загалом у V-REP доступно 10 шарів, у властивостях кожної компоненти є можливість вибору шару, на який буде винесено елемент. Після того як елементи рознесені по шарах, можна активувати інструмент «layers» («Шари») з лівої панелі інструментів і перемикатися між шарами.

Інструмент керування скриптами. Додавання і видалення скриптів, а також зміна прив'язки до окремих компонентів легко виконати через ієрархію сценарію симуляції. Однак V-REP має і дублюючий інструмент «Scripts», який теж дає змогу виконувати аналогічні операції і часто робить це швидше. Але для початківців роботу у V-REP цей інструмент може бути складним, тому що принцип його роботи не є настільки інтуїтивно зрозумілим, як аналогічна операція через ієрархії компонентів.

Редактор форм. Цей інструмент призначений для редагування сітки механічних елементів модельованої системи. Встановлення дрібнішої сітки дає змогу отримати вищу точність симуляції, але якщо подрібнити сітку в усіх компонентах, то це призводить до збільшення споживаної обчислювальної потужності. Тому необхідно задавати розміри сітки змінної концентрації за допомогою редагування сітки в ручному режимі. Найчастіше цей інструмент необхідний під час створення симуляції на основі тривимірних моделей, імпортованих із CAD-системи. У редакторі форм є три режими роботи: редагування трикутників, редагування вершин і редагування кромки (рисунок А.7).

Режим редагування трикутників. У цьому режимі окремі трикутники, що складають форму, можуть бути виділені, після чого їх можна видаляти, закривати порожнечі, що залишилися, і розбивати на більш дрібні трикутники. За допомогою кнопки «Subdivide largest triangles» можна розбити всі трикутники на більш дрібні, тобто кожне натискання зменшує розмір трикутників у два рази.

Режим редагування вершин. У цьому режимі окремі вершини, що складають фігуру, можуть бути виділені, а потім видалені. Також є можливість створення нових трикутників, для цього необхідно вибрати три або більше вершин і натиснути «Insert triangles».

Режим редагування кромки. У цьому режимі доступні функції, аналогічні описаним вище. Відмінність полягає лише в тому, що для виділення і редагування доступні окремі кромки форми.

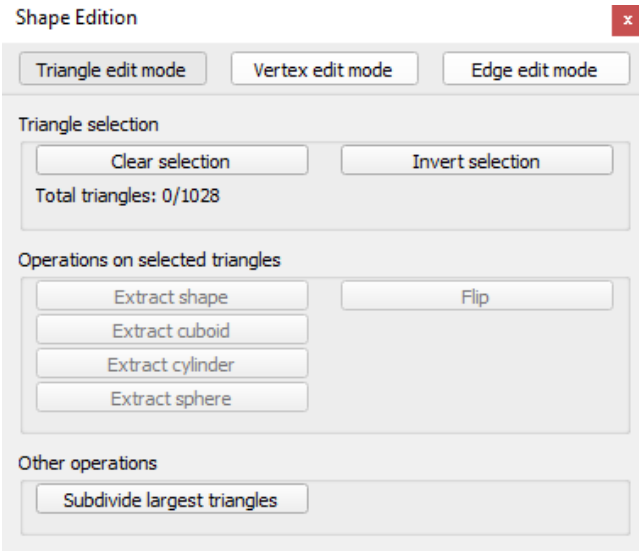

Рисунок А.7 – Інструмент редагування форм

Слід зазначити, що складові форми не можна редагувати безпосередньо за допомогою цього інструменту, їх необхідно заздалегідь розгрупувати і вносити зміни окремо.

Написання скриптів у програмі V-REP

Як було зазначено вище, платформа V-REP підтримує роботу з різними мовами програмування, зокрема і найпоширенішими, такими як C++ і Python. Більш детальне ознайомлення з можливостями написання скриптів будемо виконувати з використанням мови Lua, оскільки ця мова є вбудованою мовою V-REP, і почати написання скриптів можна відразу після запуску програми.

Мова Lua

Відмінною особливістю Lua є простота: багато в чому синтаксис схожий із популярною мовою C, що значно спрощує роботу для тих, хто вже

знає цю мову. Скрипти у V-REP відкривають великі можливості для реалізації керування як окремими об'єктами сцени, так і платформою.

Для початку необхідно згадати, що скрипти у V-REP бувають двох типів: **потокові та непотокові**. Розглянемо структуру непотокового скрипту як рекомендованого розробниками і найбільш часто використовуваного. Такий скрипт завжди складається з декількох блоків (рисунок А.8).

Блок ініціалізації. Вміст цього блока виконується тільки один раз під час запуску симуляції. У цьому блоці виконують такі операції, як оголошення необхідних змінних і присвоєння їм вихідних значень. Також у цій частині скрипту задають обробники для керування об'єктами сцени.

Блок активації. Вміст цього блока виконується ітеративно через рівні проміжки часу (крок моделювання). Частина скрипту, написана в цьому блоці, повторюватиметься аж до зупинки симуляції або виникнення критичної помилки (за якої теж симуляцію буде зупинено). Тут описується основний алгоритм керування.

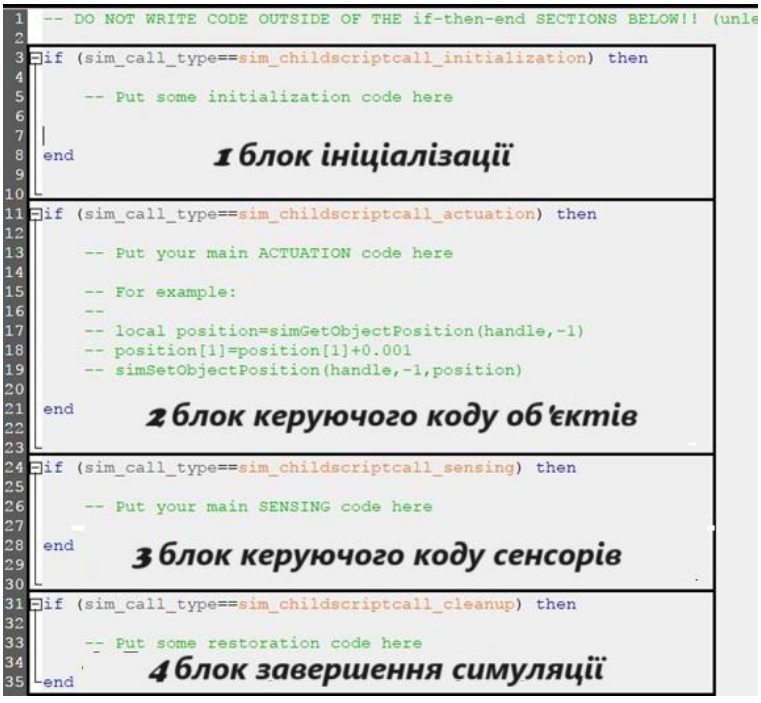

Рисунок А.8 – Вікно редагування потокового скрипту в програмі V-REP

Блок керуючого коду сенсорів. Вміст цього блока виконується стільки ж разів, скільки і блок активації. Однак основний скрипт V-REP звертається до цього блока тільки після завершення виконання скрипту з блока активації. Цей блок розроблено для отримання даних із сенсорів.

Блок завершення симуляції. Цей блок дає змогу вибірково стерти дані, отримані під час симуляції. Скрипт із цього блока починає виконуватися один раз перед завершенням симуляції або видаленням скрипту. Цей блок у більшості випадків залишається порожнім.

Скрипти непотокового типу мають простішу структуру. Увесь скрипт може вказуватися у файлі без поділу на блоки, проте з огляду на ітеративність модельованих процесів майже завжди необхідна наявність основного циклу. На рисунку А.9 наведено класичну структуру непотокового скрипту в програмі V-REP, однак основний цикл там може бути відсутній під час розв'язання задач певного класу.

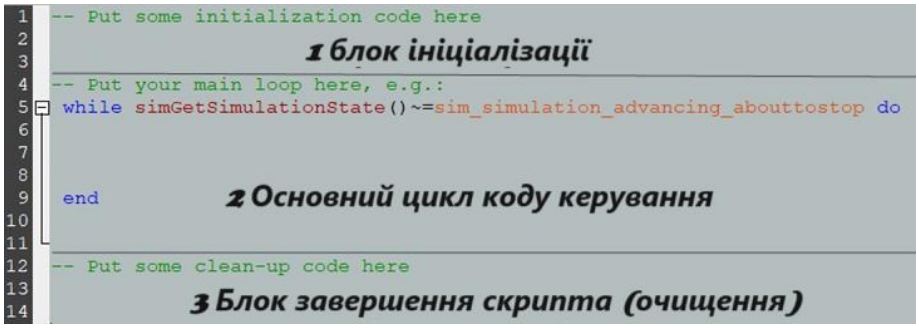

Рисунок А.9 – Вікно редагування непотокового скрипту в програмі V-REP

Основні конструкції під час написання скриптів

Коментарі мовою Lua мають починатися з подвійного дефіса (рисунок А.10, рядок 4). Рекомендовано не видаляти фрагменти скрипту під час розроблення алгоритмів керування, а позначати їх як коментар, щоб потім не довелося заново писати, якщо вони будуть потрібні.

Як правило, у кожному скрипті використовують змінні. У Lua допускають оголошення нових змінних у будь-який момент часу, також під

час оголошення можна задати їхнє початкове значення (рисунок А.10, рядок 9). Також нема необхідності вказувати тип змінної. Lua визначить його залежно від значення, яке присвоюють змінній уперше. Змінні з числовим значенням сприймаються як число з плаваючою крапкою. Також допускають використання змінних із символьними значеннями і змінних логічного типу (можливі значення «істина» або «хибність»). Є і функція знищення (звільнення) змінних. Усі змінні в Lua за замовчуванням є глобальними, тобто доступні з будь-якої області.

```
3  if (sim_call_type==sim_childscriptcall_initialization) then
4      -- Put some initialization code here
5      base=simGetObjectHandle('verh_dyn')
6      jointhandle=simGetObjectHandle('joint_1')
7      jointcurrentpos=simGetJointPosition(jointhandle)
8      simAddStatusbarMessage('position='..jointcurrentpos)
9      nextposition=1
```

Рисунок А.10 – Приклад скрипту ініціалізації

Розгалуження і цикли реалізовані в Lua так само, як і в більшості С-подібних мовах програмування, з деякими застереженнями.

Умовний оператор «If» («Якщо») вимагає обов'язкового використання конструкції «then/end». При цьому після ключового слова «If» необхідно вказати логічний вираз або змінну. Після логічного виразу слідує ключове слово «then», за яким починається «тіло» умови – частина скрипту, що виконуватиметься в разі істинності зазначеної в скрипті умови. В умовному операторі є необов'язкова складова – додаткова умова «Інакше» (elseif), «тіло» цієї умови – частина скрипту, що виконується, тільки якщо вказана в скрипті умова виявиться хибною. На рисунку А.11 наведено приклад використання умовного оператора.

```
11  k=10
12  if k ~= 10 and k<0 then
13      print('negative')
14      simAddStatusbarMessage('negative')
15  else
16      print('positive')
17      simAddStatusbarMessage('positive and k='..k)
18  end
```

Рисунок А.11 – Приклад використання умовного оператора і функцій виведення інформації в консольне вікно та вікно стану

Умовний оператор також допускає комбінування умов за допомогою операторів «and» (логічне «І») і «or» (логічне «Або»). Рядки 13 і 16 на рисунку А.11 виводять ключові слова базовими функціями Lua в консольне вікно, яке за замовчуванням приховано. Логічніше і зручніше виводити дані не в консольне вікно, а в рядок стану V-REP, для цього необхідно скористатися регулярною функцією V-REP (приклад наведено на рисунку А.11, рядки 14 і 17). Також на рисунку А.11 у рядку 17 виводиться не тільки ключове слово, а й значення змінної.

Як і у випадку з умовними операторами, система керування рідко обходиться без використання хоча б одного циклу. Цикли в Lua задаються за допомогою ключових слів, що позначають тип циклу спільно з ключовими словами «do» і «end». При цьому порядок такий: спочатку вказано тип, потім умову виконання циклу, далі слово «do», після цього «тіло циклу» – фрагмент скрипту, який буде виконуватися циклічно, і закінчується цикл словом «end». Найпоширенішими є цикли типу «while» і «for». На рисунку А.12 наведено приклад найпростішого циклу «while», який збільшує на одиницю значення змінної «k», поки умова «k<50» не стане хибною.

```
10 k=1
11 while k < 50 do
12 k = k + 1
13 end
```

Рисунок А.12 – Приклад використання циклу «while»

Цикли в Lua, як і умовне розгалуження, мають завершуватися ключовим словом «end». Особливу увагу слід приділити умові, використовуваній у циклі. Особливо це важливо, коли використовують значення змінної замість логічної умови. У невизначених змінних значення за замовчуванням дорівнює «nil». При цьому в умові циклу тільки змінні зі значенням «nil» і «false» (логічний тип змінної «хибність») повертають «false», у той час як значення змінної «0» і ' ' повертають «true».

Другий тип циклів, найчастіше застосовуваний, – «for». Цей тип добре підходить для задач, коли необхідно виконати цикл із лічильником, але слід згадати, що в більшості випадків «for» можна замінити на цикл «while», задіявши кілька додаткових змінних. Приклад використання циклу «for» наведено на рисунку А.13, цей фрагмент скрипту виконує 100 ітерацій починаючи від 1 (включно з 1 і 100). При цьому можна поміняти умови місцями, замість 1 поставити 100 і 100 замість 1, тоді цикл виконуватиметься зі зміною значення змінної «i» від 100 до 1. Також у циклі «for» є необов'язковий параметр кроку, за замовчуванням крок дорівнює одиниці, і нема необхідності його вказувати. Однак для випадків, коли потрібне використання кроку зі значенням, відмінним від 1, потрібно вказати його через кому після кінцевого значення (якщо необхідний крок 2, то, як приклад на рисунку А.13, це буде умова «i = 1, 100, 2»).

```
10      Sum = 0
11      for i = 1, 100 do -- 100 iterations from 1.
12       Sum = Sum + i
13      end
```

Рисунок А.13 – Приклад використання циклу «for»

Таблиці в Lua є єдиним структурним елементом, вони поєднують у собі властивості масиву, хеш-таблиці («ключ» – «значення»), структури та об'єкта. Найчастіше таблиці використовують як словники, а ключ при цьому за замовчуванням має рядковий (символьний) тип. Приклад наведено на рисунку А.14. Рядок 10 оголошує змінну типу «таблиця» і задає два ключі та відповідні їм значення. Доступ до значень можна отримати зазначенням назви таблиці та ключа через крапку (приклад на рисунку А.14).

```
11      t = {key1 = 'value1', key2 = false}
12      print(t.key1) -- 'value1'
13      t.newKey = {} --add new key
14      t.key2 = nil -- delete key2
```

Рисунок А.14 – Приклад використання таблиць мовою Lua

Як видно з прикладу, допускається використовувати як ключ не тільки рядки, але також і всі інші типи змінних, доступні в Lua. Іншою, і вкрай важливою, складовою мови програмування є функції.

Функції можна створювати свої власні, а також використовувати вже готові, які є в довіднику з Lua. Приклад заданої розробником функції наведено на рисунку А.15. Як видно з прикладу, функція може набувати кількох значень. Також функції можуть повертати кілька значень.

```
6  function bar(a, b, c)
7      sum=a+b+c
8      r=a-b-c
9    return sum,r
10 end
11 k,p=bar(2,2,3)
12 simAddStatusbarMessage('k='..k)
13 simAddStatusbarMessage('p='..p)
```

Рисунок А.15 – Приклад використання користувацької функції мовою Lua

Особливу увагу слід приділити регулярним функціям V-REP. Ці функції вже інтегровані в Lua під час написання скриптів V-REP і починаються на «sim». Саме ці функції і використовують для взаємодії з симуляцією: керування, зчитування даних, налагодження та вирішення багатьох інших завдань. Повний список функцій із докладним описом змінних, яких вони набувають, і даних, що повертаються, доступний в офіційній документації V-REP.

Типи об'єктів програмі V-REP

У програмі V-REP доступна велика кількість об'єктів різного призначення. Деякі з них застосовують вкрай рідко, а деякі потрібні під час моделювання кожної мехатронної та робототехнічної системи. Тут викладено основні типи об'єктів, їхні властивості та призначення.

Типи фізичних об'єктів (форми). Форми у V-REP являють собою жорсткі сітчасті об'єкти, що складаються з трикутних граней. Вони можуть бути імпортовані, експортовані та відредаговані; входять до трьох різних підтипів: проста випадкова форма, складена випадкова форма; проста опукла форма, складена опукла форма; чиста проста форма, чиста складена форма. На рисунку А.16 подано піктограми всіх типів у тому самому порядку зліва направо, як зазначено вище.

Рисунок А.16 – Піктограми різних типів форм у V-REP

Піктограми типу форми виведені в головній ієрархії сцени ліворуч від назви кожного компонента механічної складової робототехнічної системи. Як було зазначено, усі вони поділяються на три типи, і кожен тип – на простий (єдиний) і складовий (що складається з декількох простих).

Відмінні риси випадкових форм полягають у тому, що вони можуть бути будь-якої форми, але для моделювання динаміки їх не рекомендовано використовувати, тому що це потребуватиме значних обчислювальних ресурсів. Тому їх застосовують часто для отримання хороших візуалізацій, часто моделюють тільки візуальні властивості, і вони є такими, що повторюють рух опуклих і чистих форм.

Опуклі та чисті форми містять оптимізовану сітку і прийнятні для моделювання динаміки, проте візуальні властивості опуклих форм часто мають незвичний вигляд. Тому рекомендовано використовувати опуклі форми разом із випадковими формами, де опукла форма бере участь у розрахунках фізичних властивостей (водночас візуальні властивості приховано), а випадкова – у розрахунках візуальних властивостей (фізичні властивості відключено).

V-REP також дає змогу створювати оптимізовані для моделювання опуклі форми на основі випадкових форм.

Додавання форм у середовищі V-REP може бути виконано двома способами: через вбудований інструмент «Primitive Shape» і меню [File→import Mash from...]. В останньому випадку необхідно заздалегідь створити тривимірну модель у будь-якій CAD-системі та зберегти модель у форматі STL.

З'єднання елементів у V-REP. «Шарнір» («Joint») – це об'єкт, який має хоча б один внутрішній ступінь свободи. З'єднання використовують для створення механізмів і завдання переміщень інших об'єктів. У V-REP доступно три типи шарнірних з'єднань: обертальний шарнір, призматичний і сферичний.

Шарніри додають у сценарій симуляції через головне меню: [Add → Joint]. Нові шарніри мають нульові координати в глобальній системі координат сцени. Тому необхідно після додавання шарніра коректно задати необхідну орієнтацію, скориставшись інструментами «Object/Item Shift» і «Object/Item Rotation». Після того як у шарніра буде встановлено необхідне положення й орієнтацію, можна з'єднувати шарнір з іншими об'єктами шляхом створення деревоподібної ієрархії. Цей процес більш детально розкрито в лабораторних роботах.

З'єднання з нульовим ступенем свободи в програмі V-REP виконують без використання елементів типу шарнір. Два динамічні елементи можуть бути з'єднані через елемент типу «Force sensor». Існує й альтернативний варіант - скористатися контекстним меню («Edit» → «Grouping/Merging» → «Group selected shapes»), попередньо виділивши необхідні для з'єднання компоненти. Через інструмент групування допускається з'єднання необмеженої кількості елементів.

Об'єкти типу «Шарнір» мають велику кількість налаштувань. Вони, зокрема, дають змогу моделювати роботу вбудованих двигунів. Також візуальні властивості шарнірів можуть бути відредаговані через властивості об'єкта. Візуальні властивості жодним чином не впливають на симуляцію і використовувані винятково для зручності роботи під час створення сценарію симуляції.

Графіки. Виведення даних на графіку є одним із найпоширеніших і найзручніших способів подання. У програмі V-REP є окремий клас об'єктів «Graph» (графік), за допомогою якого можна легко візуалізувати як дані з

окремих сенсорів, так і користувацькі дані (незалежно від способу отримання). Щоб додати графіки в сценарій симуляції, достатньо в головному меню виконати «Add→Graph». Далі необхідно задати властивості. Для цього потрібно виділити об'єкт в ієрархії сценарію й активувати інструмент «Object/Item Properties». У контекстному меню, що з'явилося, у розділі «Data stream recording list» необхідно натиснути «Add new data stream to record». У контекстному вікні, що з'явилося (рисунок А.17), необхідно вказати тип даних і джерело даних. Якщо необхідно вивести дані з сенсора на графік, то необхідно вказати тип даних у спливаючому полі «Data stream type» і назву сенсора в полі «Object/Item to record». Найчастіше виникає необхідність у попередній обробці даних із сенсорів перед виведенням на графік: зробити це можна через використання скрипту як проміжного пункту, де і буде виконана обробка даних. У такому разі необхідно вказати в налаштуваннях графіка, що будуть виведені користувацькі дані, тобто встановити «Various: user-defined» у полі «Data stream type», а в полі «Object/items to record» вибрати «User data».

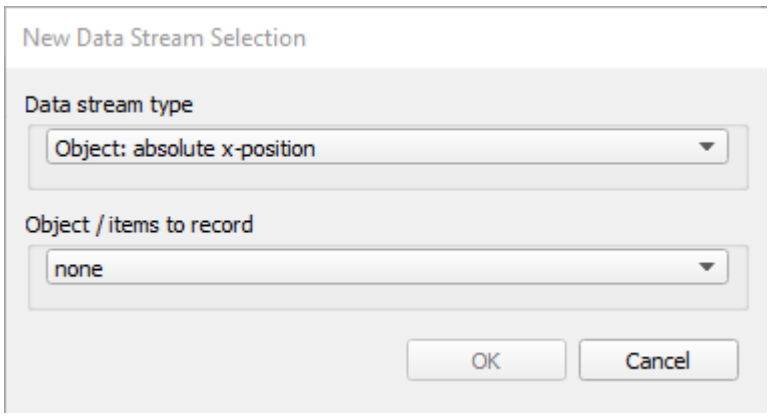

Рисунок А.17 – Контекстне вікно створення потоку даних у V-REP

Після натискання «ОК» у розділі «Data stream recording list» з'явиться новий потік даних із назвою за замовчуванням. Подвійним натисканням на назву потоку даних можна перейти в режим редагування назви. Цю назву буде використано для звернення до графіка зі скрипту під час виведення даних.

На рисунку А.18 наведено приклад фрагмента скрипту, який виводить користувацькі дані зі скрипту на графік. У прикладі в рядку 4 «Graph» – назва об'єкта, у рядку 7 «Red» – назва потоку даних, «data» – назва змінної, значення якої виведено на графік. Як отримати дані з сенсора, дивіться в розділі «Сенсори».

```
3  if (sim_call_type==sim_childscriptcall_initialization) then
4      graph=simGetObjectHandle("Graph")
5  end
6  if (sim_call_type==sim_childscriptcall_actuation) then
7      simSetGraphUserData(graph,'Red',data)
8  end
9
```

Рисунок А.18 – Фрагменти скрипту для виведення користувацьких даних на графік у програмі V-REP

Сенсори. У програмі V-REP доступні різні типи сенсорів: сенсор сили та моментів («force sensor»), відеосенсори («vision sensor»), сенсори відстані («proximity sensor»). Читання даних із сенсорів виконується в скрипті з використанням спеціальних функцій (API-функцій). Приклади читання даних із «vision sensor» і «proximity sensor» наведено на рисунку А.19.

```
2  if (sim_call_type==sim_childscriptcall_initialization) then
3      sensor1=simGetObjectHandle('Sensor')
4      sensor2=simGetObjectHandle("Proximity")
5  end
6  if (sim_call_type==sim_childscriptcall_actuation) then
7      result1,data=simReadVisionSensor(sensor)
8      if (result1>=0) then
9          simAddStatusbarMessage('Data[1]='..data[1])
10     end
11     result2,distance=simReadProximitySensor(sensor2)
12 end
```

Рисунок А.19 – Фрагменти скрипту для читання даних із сенсорів у програмі V-REP

МОДЕЛЮВАННЯ РОБОТОТЕХНІЧНИХ СИСТЕМ У V-REP

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт і самостійної роботи

з дисципліни

«ТЕОРЕТИЧНІ ОСНОВИ РОБОТОТЕХНІКИ»

Відповідальний за випуск Щебликіна О. В.

Редактор Ібрагімова Н. В.

Підписано до друку 03.04.2024 р.
Умовн. друк. арк. 4,25. Тираж . Замовлення № .
Видавець та виготовлювач Український державний університет залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха,7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.