

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“ ” _____ 2024 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності 122 ***“Комп’ютерні науки”***

на тему: **Мауа-плагін для редагування тривимірних об’єктів**

Виконав:
студент IV курсу, групи ТМ-02

_____ ФЕДУСЕНКО Федір Сергійович
(прізвище, ім’я, по батькові)

_____ (підпис)

Керівник:

_____ асистент, Олександр Кардашов
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Рецензент: _____

_____ (посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Н.контроль: _____ асистент Володимир РУДИК
(посада, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

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

Студент _____
(підпис)

Київ – 2024

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)

спеціальність 122 “Комп’ютерні науки”

Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

Наталія АУШЕВА

«__» _____ 2024 р.

**ЗАВДАННЯ
на дипломну роботу студенту**

ФЕДУСЕНКУ Федору Сергійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи Мауа-плагін для редагування тривимірних об’єктів

керівник роботи *Кардашов Олександр Вадимович, асистент*

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 202__ р. № _____

2. Термін подання роботи студентом 07.06.2024 року

3. Вихідні дані до роботи мова програмування Python, Python модуль cmds, Maya Embedded Language, Maya API, середовище розробки Qt Designer,

4. Перелік питань, які потрібно розробити _____

1) вивчення функціональності та обмежень наявних рішень

2) аналіз технологій та інструментів для розробки плагіна

3) створення архітектури програмного забезпечення, включаючи структуру плагіну та компоненти

4) демонстрація процесу використання плагіну у середовищі Maya

5. Орієнтований перелік ілюстративного матеріалу рисунки, що демонструють роботу плагіну, взаємодію користувачів із плагіном; приклади роботи з програмним продуктом.

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи	17.05.2024	
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.23	
8.	Передзахист	03-06.06.23	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студент

(підпис)

Федір ФЕДУСЕНКО

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Олександр КАРДАШОВ

(ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Записка містить 64 сторінок, 36 рисунків, 1 додаток, 14 посилань.

Метою кваліфікаційної роботи є розробка інструменту, який дозволяє моделювати кабелі з високою точністю та ефективністю, забезпечуючи значне зниження часу моделювання. Розробка базується на використанні мов програмування Python та MEL (Maya Embedded Language), а також використанні Maya Python API та інструменту Qt Designer для створення користувацького інтерфейсу.

У роботі описано основні етапи створення плагіну, включаючи аналіз існуючих рішень, проектування архітектури плагіну, розробку інтерфейсу та інтеграцію плагіну в середовище Maya. Також представлені результати тестування плагіну та рекомендації щодо його використання та подальшого розвитку.

Рекомендації з використання плагіну включають його адаптацію для освітніх цілей у сфері 3D графіки та анімації, а також комерційне використання в індустрії комп'ютерних ігор та кіно. Це демонструє високу ефективність плагіну та його потенціал у професійному середовищі моделювання.

Ключові слова: AUTODESK MAYA, MAYA-ПЛАГІН, 3D МОДЕЛЮВАННЯ, PYTHON, MEL, MAYA PYTHON API, QT DESIGNER, КАБЕЛЬ, ДРІТ.

ABSTRACT

The report contains 64 pages, 36 figures, 1 appendix, and 14 references.

The qualification work aims to develop a tool that allows for the precise and efficient modeling of cables, significantly reducing modeling time. The development is based on using Python and MEL (Maya Embedded Language) programming languages and the Maya Python API and Qt Designer tool for creating the user interface.

The work describes the main stages of plugin creation, including the analysis of existing solutions, plugin architecture design, interface development, and plugin integration into the Maya environment. It also presents the results of plugin testing and recommendations for its use and further development.

Recommendations for using the plugin include its adaptation for educational purposes in the field of 3D graphics and animation, as well as commercial use in the computer game and film industries. That demonstrates the high efficiency of the plugin and its potential in the professional modeling environment.

Keywords: AUTODESK MAYA, MAYA-PLUGIN, 3D MODELING, PYTHON, MEL, MAYA PYTHON API, QT DESIGNER, CABLE, WIRE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ТЕРМІНІВ.....	8
ВСТУП.....	9
1. РОЗРОБКА ПЛАГІНУ ДЛЯ AUTODESK MAYA ДЛЯ СТВОРЕННЯ КАБЕЛІВ..	10
1.1 Постановка прикладної задачі, яка розв’язується.....	10
1.2 Аналіз технологій для розробки плагінів в Autodesk Maya.....	12
1.3 Важливість правильно вибору інструментів для розробки плагіна.....	13
1.3.1 Вибір інструментів для розробки: Python, MEL, та API Autodesk Maya..	14
1.3.2 Дослідження можливостей Qt Designer для створення користувацького інтерфейсу.....	16
1.4 Огляд існуючих рішень для створення дротів у 3D-середовищі Maya.....	17
2. ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОЗРОБКИ.....	22
2.1 Використання Python та MEL у розробці плагіна для Maya.....	22
2.2 Використання Qt Designer для створення графічного інтерфейсу користувача	23
2.3 Застосування Maya Python API.....	24
3. МАТЕМАТИЧНЕ ПОДАННЯ СПОСОБУ КОНСТРУЮВАННЯ КРИВИХ ТА МОДЕЛЮВАННЯ ОСВІТЛЕННЯ.....	26
3.1 Теоретичні основи сплайн-кривих Безьє.....	26
3.1.1 Математичні основи кривих Безьє.....	27
3.1.2 Властивості кривих Безьє.....	29
3.1.3 Застосування кривих Безьє та кубічних сплайнів у плагіні.....	30
3.2 Важливість правильного шейдингу.....	30
3.2.1 Освітлення та відблиск від поверхні.....	31

3.2.2 Blinn-Phong шейдинг.....	33
4. ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОГО ПРОДУКТУ.....	36
4.1 Структура плагіна та його компоненти.....	37
4.2 Реалізація функцій створення кривих та маніпуляцій з дротами.....	39
4.3 Інтеграція інтерфейсу користувача із функціональністю плагіна.....	40
5. ВЗАЄМОДІЯ КОРИСТУВАЧА З ПЛАГІНОМ.....	42
5.1. Інструкції з інсталяції, вимоги до системи для встановлення та використання плагіна.....	43
5.2. Опис інтерфейсу користувача та інструкція з використання розробленої системи.....	44
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	58
ДОДАТКИ.....	60
Додаток А. Реалізація основних функцій плагіна.....	60

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ТЕРМІНІВ

API – це прикладний програмний інтерфейс (англ. Application Programming Interface).

MEL – Maya Embedded Language.

Хоткей – комбінація клавіш для виконання певної команди.

Едж – ребро.

Софт – гладкий.

Меш – геометрична сітка яка є набір вершин, ребер і граней, які описують форму тривимірного об'єкта.

Скейл – зміна розміру певного об'єкта.

Шейдинг – використання затемнення або просвітлення окремих ділянок при створенні об'єкта.

Stroke – об'єкт, який формується зі штрихів, створений за допомогою інструмента Paint Effects.

ВСТУП

Розвиток цифрових технологій і зростання потреби у більш ефективних і точних інструментах для 3D-моделювання постійно підвищує вимоги до програмного забезпечення та інструментів, зокрема до таких потужних платформ як Autodesk Maya. Актуальність теми обумовлена потребою у створенні спеціалізованих інструментів, що забезпечують моделювання складних об'єктів, таких як кабелі, дроти та подібні конструкції з високою точністю та ефективністю. Створення плагіну для Maya, який оптимізує і полегшує ці процеси, є не тільки важливим кроком для професіоналів у цій області, але й сприяє підвищенню загальної продуктивності робочих процесів.

Метою даної кваліфікаційної роботи є розробка плагіну для Autodesk Maya, який дозволяє користувачам з високою точністю та ефективністю моделювати кабелі. Цей плагін забезпечує значну економію часу, необхідного для моделювання складних кабельних систем. Для досягнення обраної мети були поставлені наступні завдання:

- 1) аналіз існуючих методів та інструментів для створення кабелів у 3D середовищах;
- 2) аналіз програмних засобів для реалізації плагіна Maya;
- 3) розробка програмного забезпечення для моделювання кабелів;
- 4) тестування розробленого програмного забезпечення.

Структура роботи включає вступ, три основні розділи, що описують розробку плагіну, технології та інструменти розробки, математичне моделювання та програмну реалізацію, висновки та список використаних джерел. Загальний обсяг роботи становить 64 сторінок, що включають 36 ілюстрацій, 1 додаток та 14 посилань.

1 РОЗРОБКА ПЛАГІНУ ДЛЯ AUTODESK MAYA ДЛЯ СТВОРЕННЯ КАБЕЛІВ

Розвиток цифрових технологій відкриває нові можливості для вдосконалення професійних інструментів у різних галузях, зокрема у 3D моделюванні. Спеціалізоване програмне забезпечення, таке як Autodesk Maya, є важливим елементом у створенні візуальних ефектів, анімації та інших аспектів цифрового контенту. З огляду на це, плагіни, що розширюють можливості цих програм, мають значний потенціал для підвищення ефективності робочих процесів.

1.1 Постановка прикладної задачі, яка розв'язується

Метою розробки плагіну для Autodesk Maya є надання моделерам, візуалізаторам та інженерам інструменту для створення, редагування та управління дротами в програмі Autodesk Maya. Цей плагін дозволяє значно спростити процеси, пов'язані з моделюванням складних кабельних конструкцій, які використовуються в усіх сферах 3D.

Задачі, які розв'язує плагін:

- автоматизація процесу створення дротів і кабелів, знижуючи час, необхідний для моделювання;
- надання користувачам можливості швидко налаштовувати параметри кабелів, такі як щільність сітки, згладжування, товщина, кількість секцій та матеріал;
- легке редагування кабеля за точками кривої;
- встановлювати матеріали для дротів та кабелів з випадково обраними кольорами для поліпшення візуального відображення;
- додаткова функція плагіну – вибір однакових об'єктів у сцені.

Опис вхідної / вихідної інформації:

- створення кривих: вхід — параметри кривої (тип, довжина); вихід — тривимірна крива у середовищі Maya;
- налаштування параметрів: вхід — слайдери і значення параметрів (скейл, щільність, секції, твіст та інші параметри); вихід — оновлена геометрія дроту згідно вказаних параметрів;
- генерація матеріалів: вхід — команда на застосування випадкових кольорів; вихід — дріт з новим матеріалом.

Засоби розробки:

- мови програмування: Python і MEL (Maya Embedded Language) для створення логіки плагіну [1];
- Maya Python API: для інтеграції плагіна з Maya і маніпуляції об'єктами сцени;
- Qt Designer: для створення графічного інтерфейсу користувача.

Плагін для створення дротів у Autodesk Maya розроблено спеціально для 3D-моделерів, які працюють у різних галузях медіа-продукції, включаючи архітектурну візуалізацію, дизайн продукту, виробництво контенту для відеоігор та спецефекти для кіно і телебачення. Доволі часто необхідно змодельовати якісь комплексні об'єкти, такі як електричні дроти, мережеві кабелі, гнучкі труби або шланги, які вимагають точності в деталях та високої якості візуалізації.

Відеоігри та розваги: 3D-моделери в галузі відеоігор можуть використовувати плагін для створення деталізованих сцен, де необхідно точно відтворити елементи інфраструктури, такі як міські електричні лінії, кабелі у внутрішніх інсталяціях, а також складні механізми та вузли, що містять дроти та кабелі.

Дизайн продукту та промислова візуалізація: 3D-моделери в області промислового дизайну можуть використовувати плагін для створення деталізованих 3D-моделей складних пристроїв, включаючи електронні прилади та великі машини, де точне відображення кабелів є критично важливим для оцінки дизайну та функціональності.

Отже, цей плагін дає можливість моделерам ефективно вирішувати задачі, пов'язані з моделюванням дротових конструкцій зменшуючи час, необхідний для ручного моделювання складних структур, та покращуючи загальну якість і реалістичність візуального контенту.

1.2 Аналіз технологій для розробки плагінів в Autodesk Maya

Розробка плагінів для Autodesk Maya вимагає глибокого розуміння відповідних технологій та методів програмування. Maya API надає широкий спектр можливостей для розширення та налаштування її функціоналу, що дозволяє створювати високоефективні інструменти, адаптовані під специфічні потреби користувачів. Наступний аналіз технологій детально описує ключові інструменти та підходи, які використовувались при розробці плагіна, та оцінка їх можливостей реалізовувати різні функції.

Maya API дозволяє розробникам взаємодіяти з основними функціями програми, такими як створення та маніпуляція об'єктами, управління анімацією та рендерингом. API доступний у двох основних формах: Maya Commands (cmds) і Maya Python API [1]. Maya Commands є високорівневим інтерфейсом, що дозволяє легко виконувати багато стандартних операцій Maya, тоді як Maya Python API надає більш низькорівневий доступ до структури програми, що дозволяє більш деталізоване управління даними.

Розробка плагінів для Maya переважно виконується за допомогою двох мов програмування: Python та MEL (Maya Embedded Language). Python є сучасною, високорівневою мовою з великою кількістю бібліотек та фреймворків, що робить його ідеальним для швидкої розробки складних функціональних можливостей. MEL, з іншого боку, традиційно використовувалась для автоматизації задач в Maya і забезпечує тісну інтеграцію з внутрішніми функціями програми.

Інтеграція плагінів з Maya здійснюється через ряд інтерфейсів, що включають вищезгаданий Maya API. Розробники можуть використовувати ці

інтерфейси для взаємодії з основним програмним забезпеченням, забезпечуючи гладке та ефективне виконання новостворених функцій. Розробка зазвичай включає створення персоналізованих інтерфейсів користувача, обробку даних та маніпуляцію об'єктами сцени, що вимагає детального планування та тестування.

Тестування плагінів є критично важливим для забезпечення їх надійності та ефективності. Розробники повинні забезпечити, щоб нові інструменти не конфліктували з існуючими функціями Maya і не призводили до нестабільності програми. Тестування включає як ручні перевірки, так і автоматизоване тестування скриптів, що використовуються для гарантування якості та безпеки коду.

Аналіз технологій підкреслює важливість вибору відповідних інструментів та підходів при розробці плагінів для Autodesk Maya. Вибір правильних технологій і забезпечення глибокої інтеграції з Maya є ключовими для успішного впровадження нових інструментів, які можуть значно підвищити продуктивність і креативні можливості користувачів.

1.3 Важливість правильного вибору інструментів для розробки плагіна

Отже, так як успішна розробка плагіну для Autodesk Maya, ключовою функцією якого буде створення та управління дротами або кабелями, вимагає використання спеціалізованих технологій та інструментів. Ці технології та інструменти не тільки сприяють ефективній розробці та інтеграції плагіна з середовищем Maya, але й забезпечують гнучкість, потрібну для налаштування плагіна під конкретні потреби користувачів. Основні технології, які використовуються в процесі розробки, охоплюють мови програмування, бібліотеки, фреймворки та інтерфейси програмування додатків (API).

Правильний вибір інструментів є критичним для забезпечення продуктивності розробки, стабільності продукту та здатності до масштабування. Від інструментів, що використовуються для розробки інтерфейсу користувача, до

мов програмування, які дозволяють маніпулювати складними об'єктами у Maya, кожен компонент відіграє ключову роль у формуванні кінцевого продукту.

Одним з основних викликів при розробці плагінів для Maya є забезпечення їх інтеграції та сумісності з різними версіями програмного забезпечення та операційних систем. Інструменти, обрані для розробки, повинні підтримувати цю гнучкість, дозволяючи плагіну функціонувати без збоїв у різноманітних середовищах користувачів.

Інструменти, які мають активну підтримку спільноти та регулярні оновлення, надають додаткові переваги, такі як нові функції та вдосконалення безпеки. Така підтримка може бути вирішальною для довгострокової стабільності та надійності плагінів.

У наступних підпунктах детально розглянемо основні технології та інструменти, що використовуються у розробці плагіна: від програмування з використанням Python і MEL до створення користувацьких інтерфейсів з Qt Designer та використання Maya Python API для інтеграції скриптів і розширення функціоналу Maya.

1.3.1 Вибір інструментів для розробки: Python, MEL, та API Autodesk Maya

Для розробки плагіна, що дозволяє створювати та маніпулювати дротами та кабелями в Autodesk Maya, було обрано використання Python, MEL (Maya Embedded Language), і API Autodesk Maya. Цей вибір базується на аналізі вимог до функціональності плагіна, потреб користувачів, а також здатності інструментів задовольняти технічні виклики проекту.

Python є основною мовою програмування для розробки плагінів в Maya завдяки своїй гнучкості, великій підтримці спільноти та інтеграції з численними зовнішніми бібліотеками [2]. Його переваги включають:

- читабельність та простота: Python легкий для читання та написання, що спрощує розробку та підтримку коду;

- широкий спектр бібліотек: існує велика кількість бібліотек Python, що можуть бути використані для розширення функціональності плагіна, включаючи бібліотеки для математичних розрахунків, обробки даних та інтерфейсу користувача;

- підтримка Maya Python API: Maya Python API дозволяє глибоко інтегрувати скрипти Python у процеси Maya, що забезпечує велику гнучкість у маніпулюванні об'єктами сцени і автоматизації завдань [3].

MEL є важливим інструментом для розробки скриптів в Maya через його тісну інтеграцію з середовищем Maya та високий рівень контролю над вбудованими функціями програми. Використання MEL дозволяє:

- виконувати завдання автоматизації: MEL добре підходить для написання скриптів, що автоматизують звичайне моделювання в Maya;
- швидка інтеграція з Maya: MEL має високу продуктивність при взаємодії з Maya, оскільки він вбудований безпосередньо в її ядро.

API Autodesk Maya є ключовим елементом для розширення функціональності Maya через плагіни. Використання API дозволяє:

- маніпулювати об'єктами сцени та їх властивостями: API надає програмні інтерфейси для створення, зміни та видалення об'єктів у сцені Maya;
- розширювати існуючу функціональність Maya: розробники мають можливість використовувати API для додавання нових функцій, які не доступні в стандартних налаштуваннях Maya.

Ці інструменти разом формують міцну основу для розробки складного та високофункціонального плагіна для створення дротів, кабелів та інших подібних конструкцій у Maya, забезпечуючи високу продуктивність, широку адаптивність до різних задач та легкість у використанні для кінцевого користувача.

1.3.2 Дослідження можливостей Qt Designer для створення користувацького інтерфейсу

Qt Designer є потужним інструментом для створення графічних користувацьких інтерфейсів (GUI), який інтегрований з Qt Framework [4]. Цей

інструмент використовується для дизайну та розробки інтерфейсів, які можуть бути легко імплементовані в додатки, написані на мовах програмування, які підтримують Qt, включаючи C++ та Python. Використання Qt Designer для створення користувацького інтерфейсу плагіна в Autodesk Maya пропонує ряд переваг, що робить його ідеальним вибором для цієї задачі.

Qt Designer має візуальний редактор з драг-енд-дроп функціональністю, що дозволяє розробникам швидко створювати інтерфейси без необхідності писати код вручну. Це значно знижує поріг входу для розробників, які можуть не мати глибоких знань в програмуванні, але мають досвід роботи з графічними інтерфейсами.

Qt Designer надає розширену бібліотеку віджетів, таких як кнопки, текстові поля, слайдери, та інші елементи управління, які можуть бути легко інтегровані у користувацький інтерфейс [5]. Ці віджети можуть бути налаштовані за допомогою властивостей, що дозволяє досягнути необхідної функціональності та дизайну.

Qt Designer інтегрується з Maya через PySide (бібліотека Python для Qt) [6]. Це дозволяє розробникам плагінів використовувати створені GUI напряду у Maya, забезпечуючи гладку взаємодію з користувацькими скриптами та функціональністю Maya. PySide надає обгортки для Qt, що дозволяє використовувати Python скрипти для маніпуляції інтерфейсом і обміну даними між Maya і Qt.

Qt Designer підтримує створення динамічних інтерфейсів, які можуть адаптуватися до змін у логіці додатка або даних. Це особливо корисно в середовищах, де інтерфейс користувача має відображати змінні дані або конфігурації, такі як візуальне програмування або кастомізоване налаштування параметрів об'єктів.

Ці можливості Qt Designer роблять його ідеальним інструментом для розробки користувацьких інтерфейсів у плагінах для Autodesk Maya, дозволяючи розробникам зосередитися на створенні високої якості, функціональних і гнучких рішень для кінцевих користувачів.

1.4 Огляд існуючих рішень для створення дротів у 3D-середовищі Maya

Для розробки ефективного плагіна для створення дротів у Maya, важливо ознайомитися з існуючими рішеннями, що вже доступні на ринку. Це дозволяє ідентифікувати можливі недоліки в наявних інструментах та визначити, які функції можуть бути покращені або доповнені. Використання Maya для створення дротів та кабелів уже має кілька відомих підходів, кожен з яких має свої переваги та обмеження.

Maya надає базові інструменти для створення кривих та поверхонь, які можуть бути використані для моделювання дротів та кабелів. Наприклад, інструменти як “EP Curve Tool”, “Bezier Curve Tool” та “Pencil Curve Tool” дозволяють користувачам створювати криві, що можуть слугувати основою для моделювання дротів. Розглянемо цей метод детальніше, для початку створюємо криву за допомогою “EP Curve Tool” (рисунок 1.1).

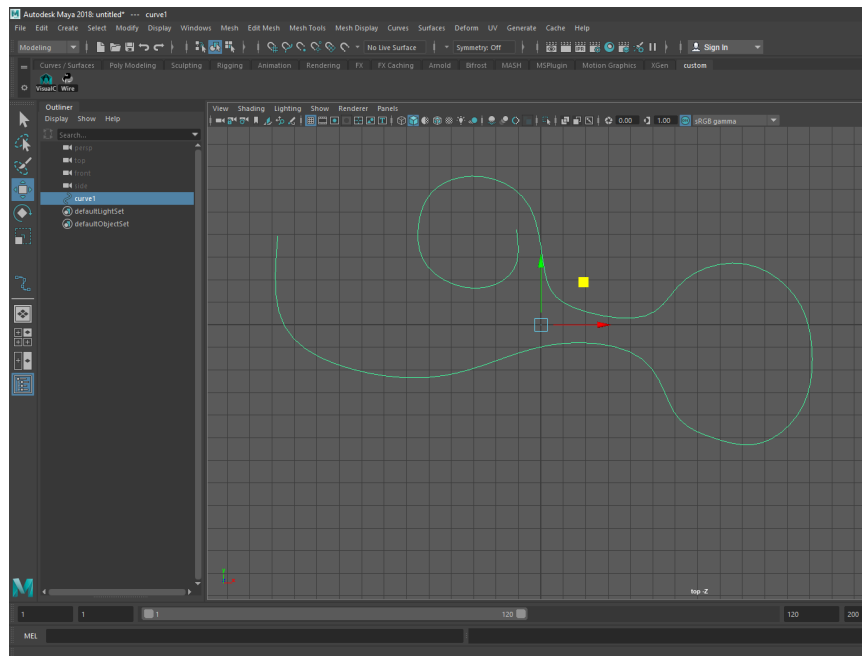


Рисунок 1.1 – Створено криву

Далі маємо створити циліндр з певною кількістю ребер, в нашому випадку нехай буде 12 (рисунок 1.2), це буде кількістю секції подальшого кабелю.

Надалі, щоб змінити кількість секцій кабелю доведеться починати процес спочатку.

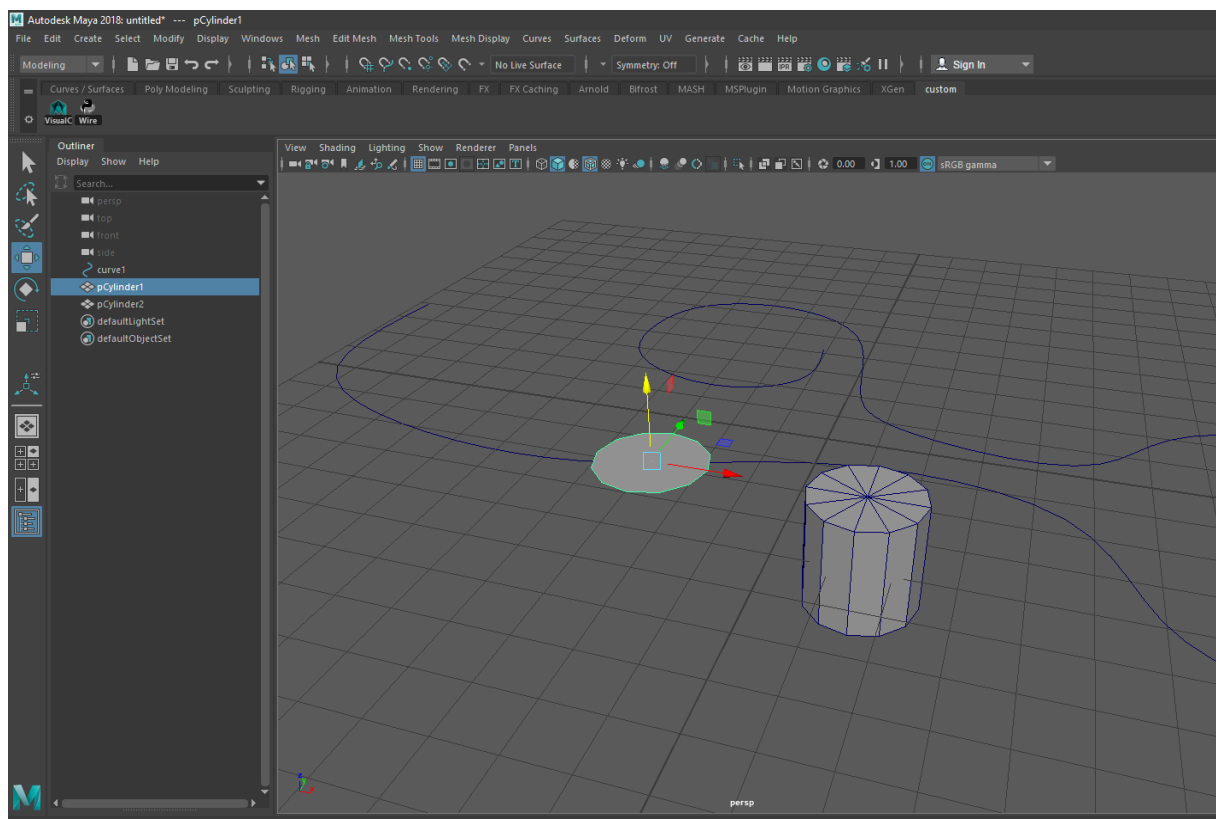


Рисунок 1.2 – Створення циліндра та видалення непотрібних граней

Наступним кроком необхідно розташувати грань циліндра в один з кінців кривою таким чином, щоб крива входила в грань перпендикулярно в центр грані (рисунок 1.3).

Далі необхідно очистити історію, щоб в подальшому не виникало ніяких помилок та скинути трансформи (обнулити значення положення об'єкта в сцені). Потім обираємо грань циліндра та криву, натискаємо хоткей `ctrl + e` для екструду та отримуємо наступне (рисунок 1.4).

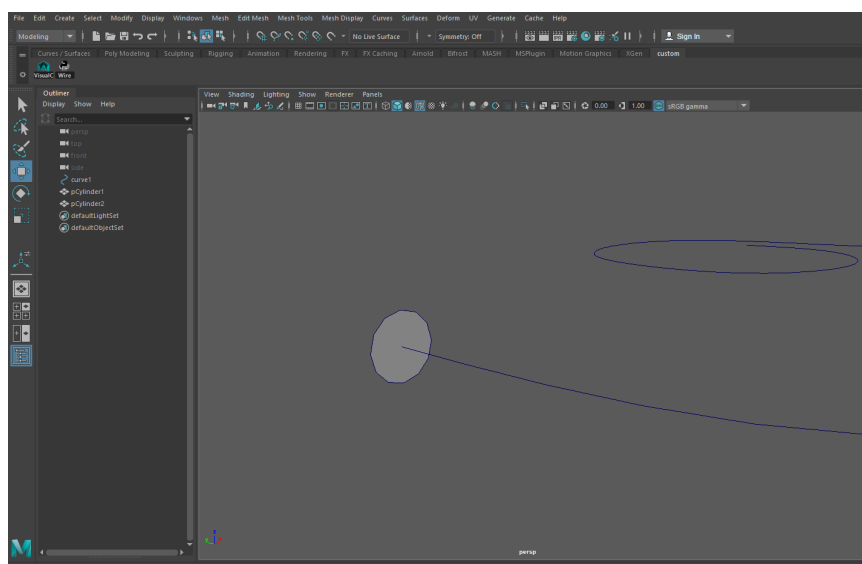


Рисунок 1.3 – Розміщення грані циліндру перпендикулярно до кривої

Додаємо ділянки (Divisions) до нашого мешу, виставимо, наприклад, 100 ділянок (рисунок 1.5).

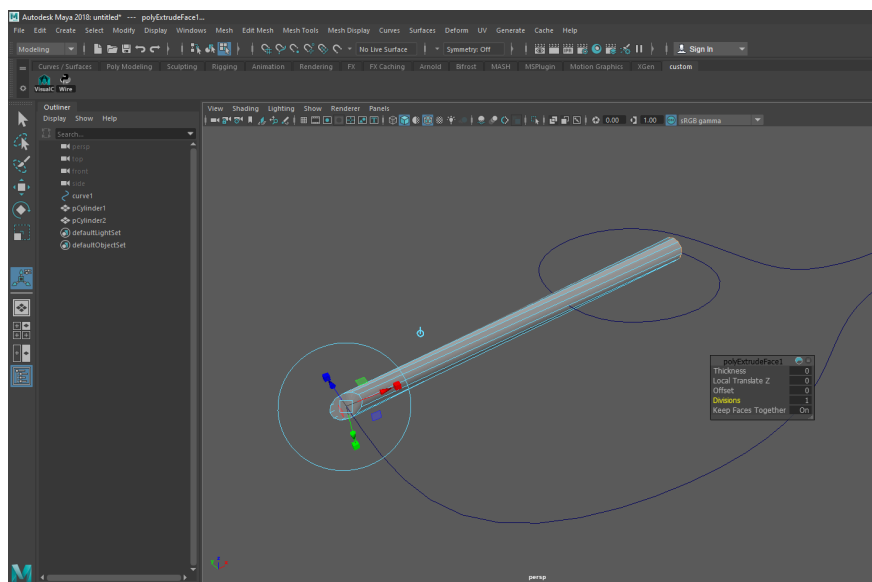


Рисунок 1.4 – Екструдимо грань циліндра по всій кривій

Тепер залишається тільки вивернути нормалі та назначити софт еджі. Допоки ми не очистили історію меша, в нас є можливість редагувати криву за допомогою контрольних точок (рисунок 1.6).

Великим недоліком цього метода, який є одним з найоптимальніших методів, не враховуючи сторонні плагіни, це неможливість редагувати розмір кабелю та кількість секцій початкової грані циліндра.

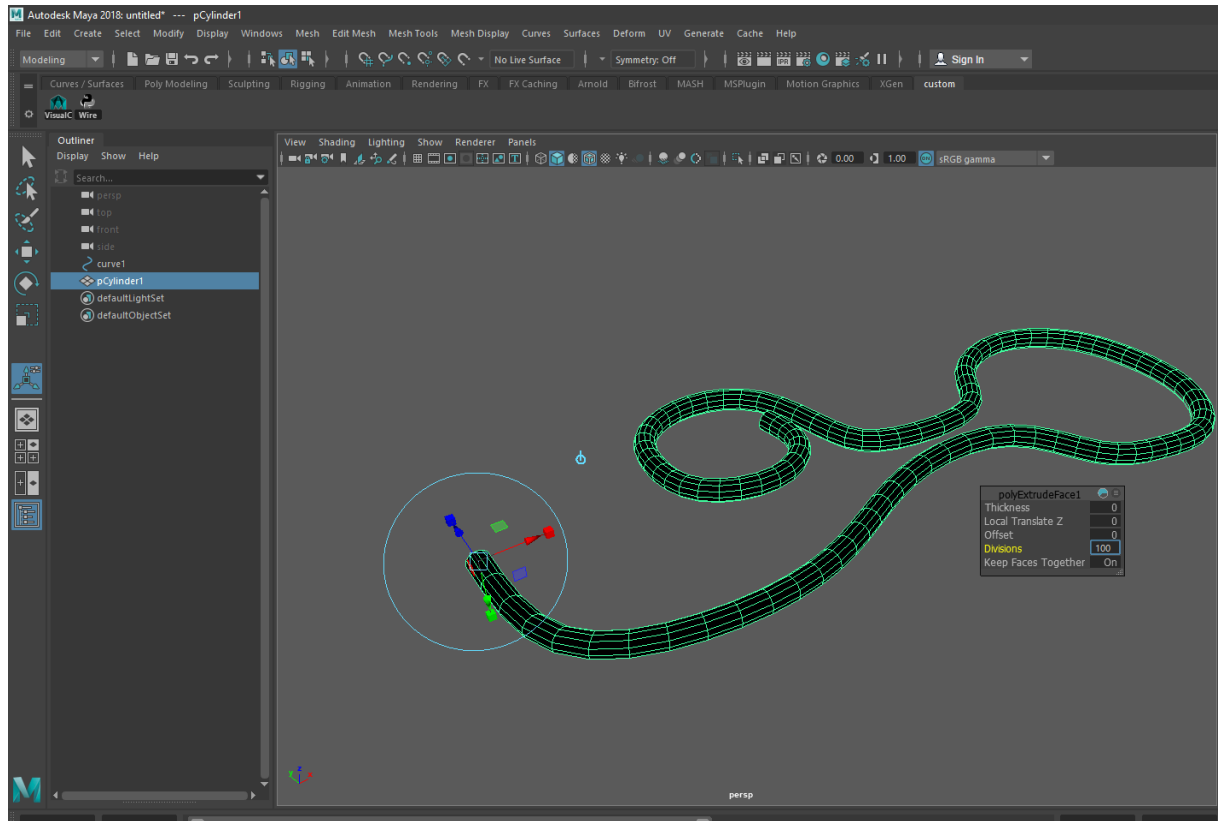


Рисунок 1.5 – Збільшення щільності сітки

Для зміни вищезазначених параметрів необхідно буде заново створювати циліндр, заново визначати кількість ребер. Стосовно розміру кабелю також потрібно буде повертатися назад та скейлити розмір грані циліндру та вже потім знову приміняти екструд по обраній кривій.

Також є інструмент Paint Effects: ця технологія дозволяє генерувати комплексні структури, включаючи дроти, використовуючи процедурні методи. Paint Effects є потужним інструментом, але є складним в налаштуванні та перетворенні “stroke” на “меш”.

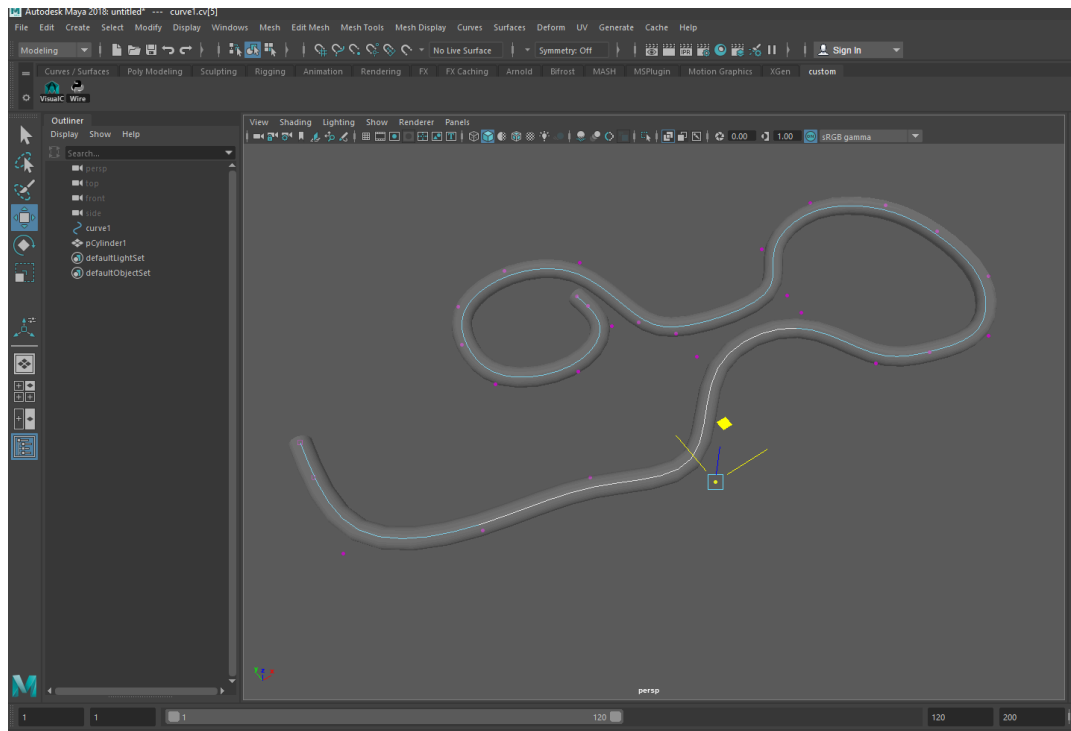


Рисунок 1.6 – Можливість редагування кривої

Отже, ці інструменти не завжди є оптимальними для складних або динамічно змінних структур, вимагаючи від користувачів багато ручної роботи та налаштувань.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОЗРОБКИ

У сучасному світі програмування, вибір правильних технологій та інструментів є критично важливим для успішної розробки будь-якого програмного забезпечення, особливо коли йдеться про спеціалізовані плагіни для комплексних програм, таких як Autodesk Maya. У цьому контексті, розділ 2 розглядає технології та інструменти, що були використані для розробки плагіну Maya для створення кабелів, а також їх вплив на функціональність і продуктивність фінального продукту.

2.1 Використання Python та MEL у розробці плагіна для Maya

Python є основною мовою програмування для всіх аспектів плагіна, включаючи логіку обробки даних, інтерфейс користувача та інтеграцію з Maya [7]. Його переваги включають високу читабельність, доступність численних бібліотек та інтеграцію з Maya Python API, що дозволяє глибоко інтегрувати скрипти Python з процесами Maya.

У плагіні Python використовується для ініціалізації інтерфейсу користувача. Код, написаний на Python, керує завантаженням та відображенням інтерфейсу, створеного за допомогою Qt Designer.

Python обробляє всі дії користувача в інтерфейсі, такі як натискання кнопок та корегування параметрів через слайдери, що дозволяє плагіну оперативно реагувати на взаємодії.

Також, Python використовується для створення кривих та управління їх параметрами, що демонструється у коді через функції, які створюють та налаштовують властивості кабелів.

MEL (Maya Embedded Language) застосовується у випадках, коли потрібна тісна інтеграція з внутрішніми механізмами Maya, яка може не бути повністю доступна через Python API. У плагіні MEL використовується для виконання

специфічних команд, таких як `polyToCurve`, які є частиною стандартних можливостей Maya і оптимально виконуються через MEL. Такий підхід гарантує, що важливі операції, які потребують високої продуктивності та надійності, виконуються ефективно та швидко.

Плагін демонструє ефективну взаємодію між Python і MEL, використовуючи Python для загальної логіки та управління інтерфейсом, а MEL для виконання специфічних завдань у Maya, таких як виклики до вбудованих команд Maya, що оптимізують робочі процеси та забезпечують більшу продуктивність.

2.2 Використання Qt Designer для створення графічного інтерфейсу користувача

У плагіні, Qt Designer був використаний для створення комплексного інтерфейса, що дозволяє користувачам керувати параметрами дротів та кабелів. Візуальне редагування інтерфейсу через Qt Designer відбувалося ефективно, оскільки цей інструмент дозволяв створювати та редагувати комплексні інтерфейси без потреби написання коду вручну.

Завдяки графічному візуальному редактору (рисунок 2.1), можна легко перетягувати та налаштовувати віджети в режимі реального часу, що значно спрощувало процес розробки.

Користувацький інтерфейс, створений з використанням Qt Designer, безпосередньо інтегрований з логікою плагіна, написаною на Python за допомогою PySide. Це забезпечило тісну взаємодію між графічними компонентами та функціональними аспектами плагіна, дозволяючи користувачам отримувати відповідь від інтерфейсу в залежності від їх дій у реальному часі.

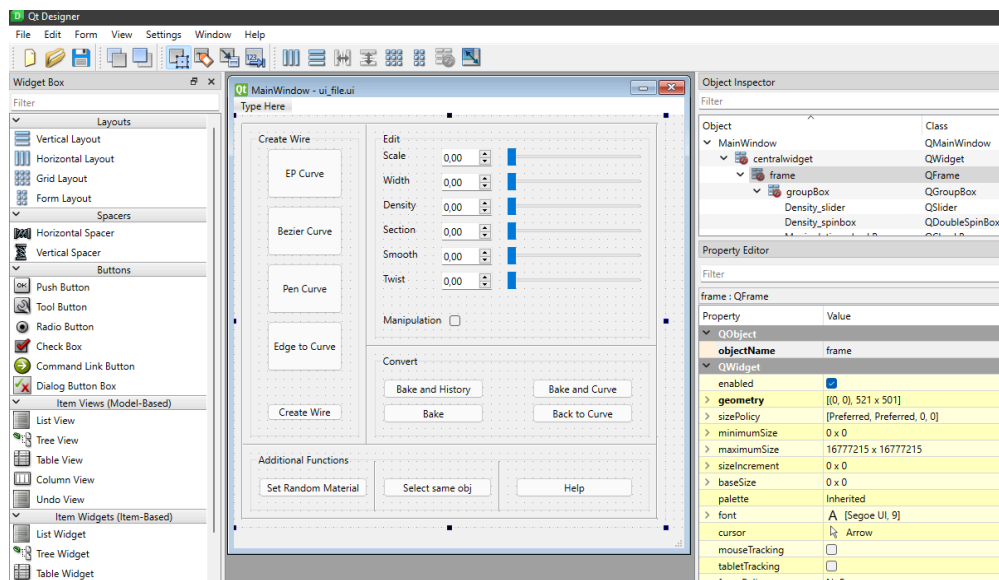


Рисунок 2.1 – Інтерфейс Qt Designer

Інтерфейс плагіна був спроектований таким чином, щоб адаптуватися до різних режимів роботи та відображати зміни параметрів об'єктів. Віджети, такі як слайдери та кнопки, були налаштовані для зміни станів дротів, кабелів або подібних конструкцій, забезпечуючи користувачам зворотний зв'язок та контроль над процесом моделювання.

Ці особливості Qt Designer дозволили створити ефективний інструмент, що відповідає потребам користувачів Maya, забезпечуючи високу продуктивність, адаптивність та інтуїтивну взаємодію з плагіном.

2.3 Застосування Maya Python API

Maya Python API відіграє ключову роль у розширенні та кастомізації функціональності Autodesk Maya через плагіни. Використання API в розробці плагіна для створення та управління дротами і кабелями дозволяє досягти високого рівня інтеграції з базовими процесами Maya, оптимізувати взаємодію користувача і забезпечити стабільну роботу програми.

Maya Python API надає можливість взаємодії з об'єктами сцени, включаючи створення, редагування та видалення об'єктів. Це особливо корисно для плагіна, який потребує динамічного управління параметрами дротів, оскільки API дозволяє програмно контролювати властивості цих об'єктів, такі як геометрія, позиціонування. Також Maya Python API дозволяє розробникам читати та змінювати атрибути об'єктів сцени. Плагін використовує цю можливість для налаштування деталізованих параметрів дротів, таких як товщина, кількість секцій, згладжування, забезпечуючи користувачам гнучке управління цими параметрами через графічний інтерфейс. API є інструментом для автоматизації звичайних задач у Maya, що може значно підвищити продуктивність користувачів. Плагін використовує API для автоматизації процесу створення кабелів, дозволяючи користувачам швидко генерувати та адаптувати ці об'єкти без потреби вручну налаштовувати кожен параметр. Плагін використовує можливості Maya Python API для інтеграції зі стандартними інструментами Maya, забезпечуючи миттєву взаємодію з іншими аспектами програми.

3 МАТЕМАТИЧНЕ ПОДАННЯ СПОСОБУ КОНСТРУЮВАННЯ КРИВИХ ТА МОДЕЛЮВАННЯ ОСВІТЛЕННЯ

Метод сплайн-апроксимації був розроблений французьким інженером П'єром Безьє для використання в дизайні кузовів автомобілів Renault. Сплайни Безьє мають низку властивостей, які роблять їх дуже корисними та зручними для проектування кривих і поверхонь. Їх також легко реалізувати. З цих причин сплайни Безьє активно використовуються доступні в різних програмах для 3D моделювання та інших графічних пакетах, а також у різноманітних програмах малювання.

3.1 Теоретичні основи сплайн-кривих Безьє

Загалом, секцію кривої Безьє можна вписати до будь-якої кількості контрольних точок, хоча деякі графічні пакети обмежують кількість контрольних точок до чотирьох. Ступінь полінома Безьє визначається кількістю контрольних точок, що потребують апроксимації, та їхнім відносним розташуванням.

Як і з інтерполяційними сплайнами, ми можемо вказати шлях кривої Безьє в околицях контрольних точок, використовуючи функції змішування, характеризуючи матрицю або граничні умови. Для загальних кривих Безьє, без обмежень щодо кількості контрольних точок, специфікація функції змішування є найзручнішим представленням.

3.1.1 Математичні основи кривих Безьє

Спочатку розглянемо загальний випадок $n + 1$ контрольних точок, позначених як $p_k = (x_k, y_k, z_k)$, де k змінюється від 0 до n . Ці координатні точки змішуються для створення наступного вектора положення $P(u)$, який описує шлях наближеної функції полінома Безьє між p_0 та p_n :

$$P(u) = \sum_{k=0}^n p_k BEZ_{k,n}(u), \quad 0 \leq u \leq 1 \quad (3.1)$$

Функції змішування Безьє $BEZ_{k,n}(u)$ є поліномами Бернштейна:

$$BEZ_{k,n}(u) = C(n, k) u^k (1 - u)^{n-k}$$

де параметри $C(n, k)$ є біноміальними коефіцієнтами:

$$C(n, k) = \frac{n!}{k!(n-k)!}$$

Рівняння 3.1 представляє набір із трьох параметричних рівнянь (3.2) для окремих координат кривої:

$$\begin{aligned} x(u) &= \sum_{k=0}^n x_k BEZ_{k,n}(u) \\ y(u) &= \sum_{k=0}^n y_k BEZ_{k,n}(u) \\ z(u) &= \sum_{k=0}^n z_k BEZ_{k,n}(u) \end{aligned} \quad (3.2)$$

У більшості випадків крива Безьє є поліномом степеня, який на одиницю менший за визначену кількість контрольних точок: три точки утворюють параболу, чотири точки — кубічну криву тощо [8]. Рисунок 3.1 демонструє зовнішній вигляд деяких кривих Безьє для різних наборів контрольних точок на площині xu ($z = 0$). Проте, з певним розміщенням контрольних точок можна отримати дегенеративні поліноми Безьє.

Наприклад, крива Безьє, створена з трьох колінеарних контрольних точок, є відрізком прямої, а набір контрольних точок, розташованих у одній і тій же координатній позиції, утворює "криву" Безьє, яка є однією точкою. Рекурсивні обчислення можна використовувати для отримання послідовних значень біноміального коефіцієнта як:

$$C(n, k) = \frac{n-k+1}{k} C(n, k-1)$$

для $n \geq k$. Також функції змішування Безьє задовольняють рекурсивні відносини:

$$BEZ_{k,n}(u) = (1 - u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u), \quad n > k \geq 1$$

з формулою:

$$BEZ_{k,k} = u^k \text{ та } BEZ_{0,k} = (1 - u)^k$$

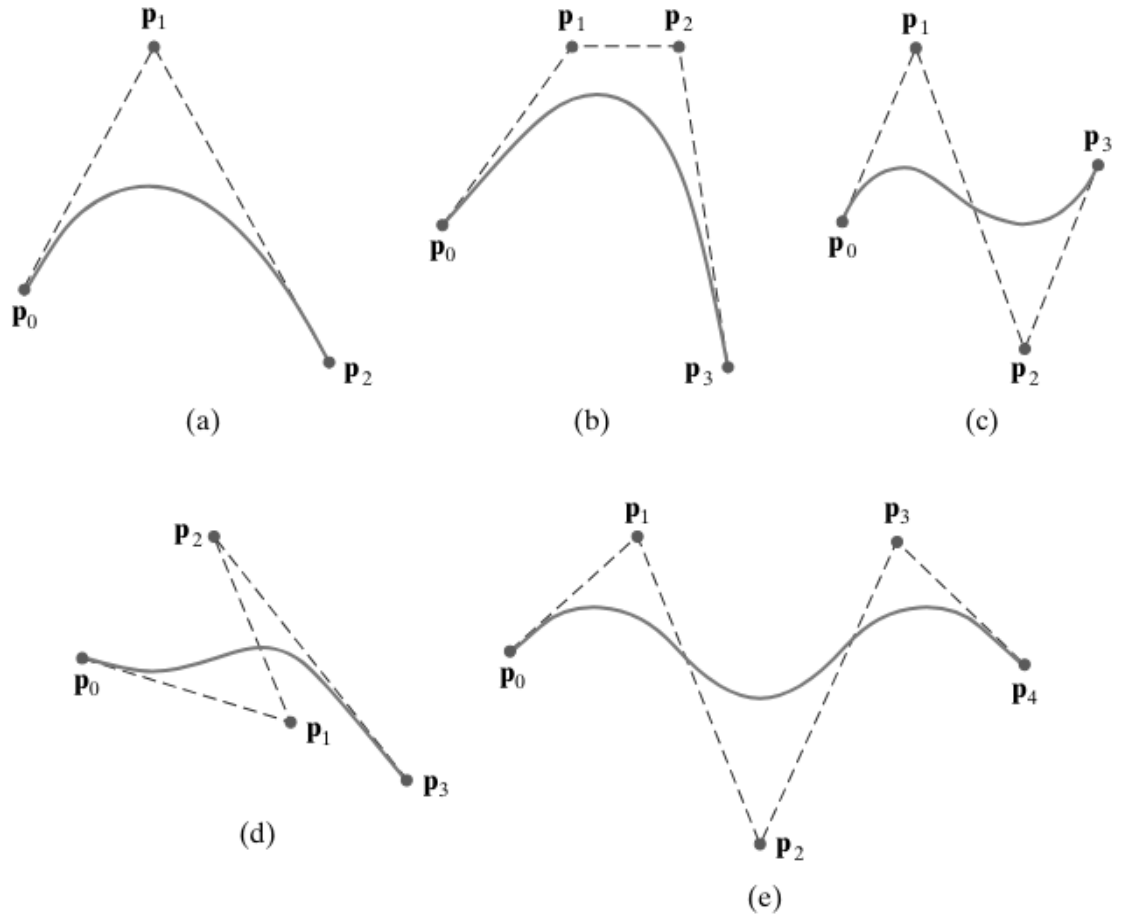


Рисунок 3.1 – Репрезентація сплайнів

Приклади двовимірних кривих Безьє, створених за допомогою трьох, чотирьох і п'яти контрольних точок. Пунктирні лінії з'єднують положення контрольних точок.

3.1.2 Властивості кривих Безьє

Дуже корисною властивістю кривої Безьє є те, що крива з'єднує першу та останню контрольні точки [9]. Таким чином, основною характеристикою будь-якої кривої Безьє є те, що:

$$P(0) = p_0$$

$$P(1) = p_n$$

Значення для параметричних перших похідних кривої Безьє в кінцевих точках можна обчислити з координат контрольної точки як:

$$P'(0) = -np_0 + np_1$$

$$P'(1) = -np_{n-1} + np_n$$

З цих виразів ми бачимо, що нахил на початку кривої знаходиться вздовж лінії, що з'єднує перші дві контрольні точки, а нахил у кінці кривої – уздовж лінії, що з'єднує дві останні кінцеві точки. Подібним чином параметричні другі похідні кривої Безьє в кінцевих точках обчислюються як:

$$P''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$$

$$P''(1) = n(n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

Інша важлива властивість будь-якої кривої Безьє полягає в тому, що вона лежить у опуклій оболонці (границі опуклого багатокутника) контрольних точок. Це впливає з того факту, що всі функції змішування Безьє додатні, а їх сума завжди дорівнює 1:

$$\sum_{k=0}^n BEZ_{k,n}(u) = 1$$

таким чином будь-яке положення кривої є просто зваженою сумою положень контрольних точок.

Властивість опуклої оболонки для кривої Безьє гарантує, що поліном плавно слідує контрольним точкам без непостійних коливань.

3.1.3 Застосування кривих Безьє та кубічних сплайнів у плагіні

У плагіні для Autodesk Maya криві Безьє та кубічні сплайни використовуються для ефективного моделювання складних кривих, які слугують основою для формування дротів і кабелів у 3D-просторі. Функціональність кривих у плагіні включає можливість створення різних типів кривих, від простих Безьє до більш складних кубічних сплайнів. Ці криві дозволяють користувачам легко ініціювати процес моделювання, вибираючи криві як інструменти для первинного створення форми майбутнього кабелю.

Форма кривих в плагіні може бути налаштована за допомогою зміни розташування контрольних точок кривих, завдяки динамічного редагування кривих, користувачі можуть детально керувати формою кривих, що важливо для точного відтворення форм дротів, кабелів та подібних конструкцій. Це дозволяє користувачам детально налаштовувати властивості кривих, забезпечуючи точність та гладкість ліній, що є критично важливим для візуалізації складних 3D моделей.

Після створення та налаштування кривих плагін використовує вбудовані засоби Maya для рендеринга кінцевого об'єкту, що забезпечує високу якість та точність моделювання. Використання кривих Безьє та кубічних сплайнів значно підвищує якість роботи, дозволяючи створювати більш складні та деталізовані об'єкти, покращуючи робочий процес та ефективність візуалізації ідей.

3.2 Важливість правильного шейдингу

Шейдинг або затінення у тривимірному моделюванні та анімації — це процес визначення візуальних характеристик поверхні об'єкта, таких як колір, яскравість та текстура, які використовуються для відтворення реалістичного візуального ефекту під час рендеринга. В Autodesk Maya та інших 3D програмах шейдинг відіграє ключову роль у створенні візуальних ефектів та реалістичних зображень.

Інформація про поверхню моделі, наприклад положення точок на поверхні та вектори нормалей у цих точках, зберігається лише для кожної вершини трикутної сітки [10]. Нормалі у 3D графіці — це вектори, які вказують напрямки "назовні" з поверхні полігону або мешу і відіграють критичну роль у багатьох аспектах обробки та візуалізації. Вони визначають, як світло взаємодіє з поверхнею, що є основою для різних технік шейдингу, таких як Гуро шейдинг чи Фонг шейдинг. Нормалі також використовуються для обчислення кутів відбиття та заломлення світла, що дозволяє реалістично відтворювати матеріали, які мають властивості прозорості, наприклад, скло або вода.

Коли візуалізується один трикутник, інформація, відома в кожній вершині, інтерполюється по всій грані трикутника. Вершини трикутника несуть таку інформацію, як кольори освітлення та координати відображення текстури, які називаються атрибутами вершин.

Кожен атрибут вершини повинен бути інтерпольований по всій грані трикутника під час його растеризації. Більш сучасне графічне обладнання дозволяє обчислювати повну формулу освітлення для кожного окремого пікселя, намальованого на дисплеї. Спосіб визначення освітлення для поверхні трикутника в поєднанні з будь-якою кількістю текстурних карт називається затіненням ("шейдингом").

3.2.1 Освітлення та відблиск від поверхні

Колір, який ми обчислюємо для будь-якої точки на поверхні, є сумою внесків від усіх джерел світла, які освітлюють поверхню. Стандартні типи джерел світла, які підтримуються системами тривимірної графіки, бувають чотирьох різновидів (рисунок 3.2): навколишнє, спрямоване, точкове та освітлення площиною.

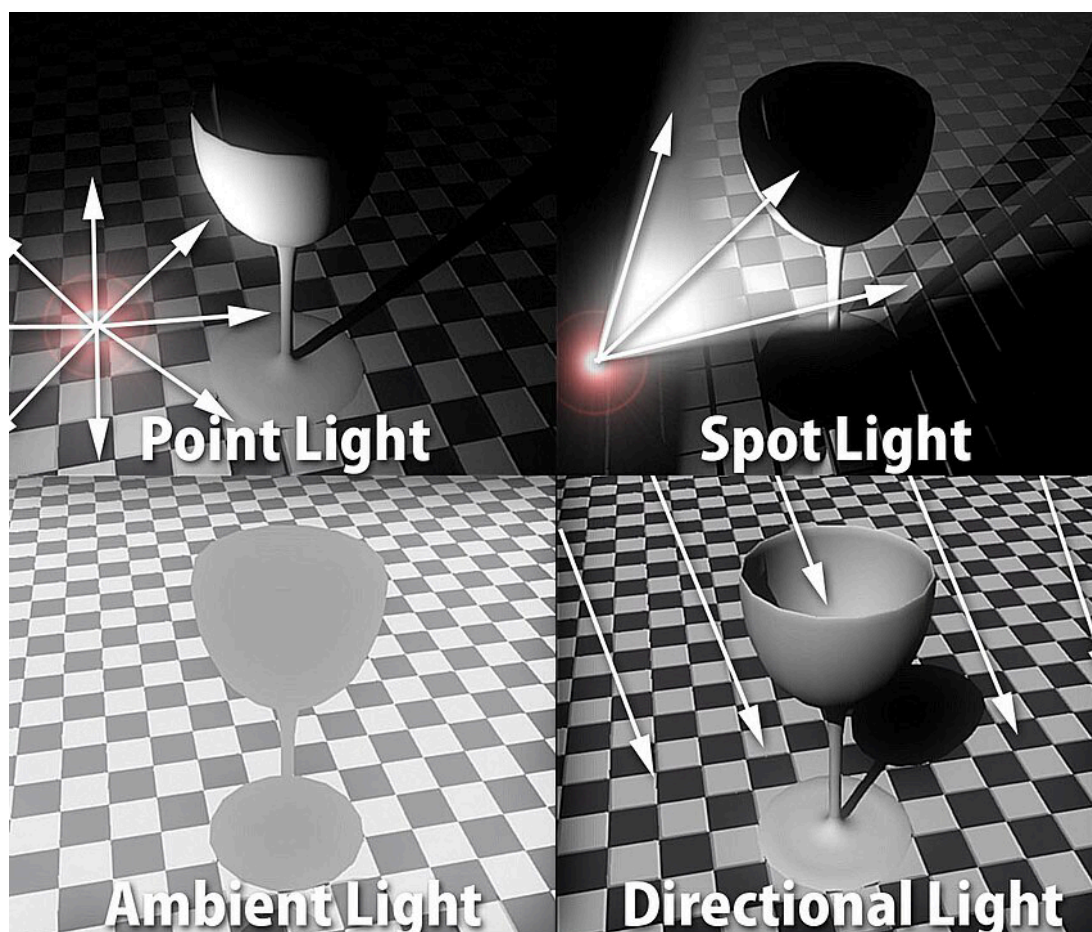


Рисунок 3.2 – Типи освітлення

Будь-які поверхні мають властивість відбивати світло вздовж траєкторії, заданої відбиттям напрямку падіння через нормаль до поверхні [11]. Це призводить до появи блискучого відблиску на поверхні, який називається Specular Reflection. На відміну від дифузного відбиття, дзеркальне відбиття, видиме на поверхні, залежить від кута, з якого глядач дивиться на поверхню.

Формулювання Specular Reflection, яке потребує менше обчислень у деяких випадках, використовує напрямок, який називається половинним вектором. Показаний на малюнку 3.3 половинний вектор H — це вектор, що лежить точно посередині між вектором V , спрямованим на глядача, і вектором L , спрямованим на освітлення. Specular Reflection є найбільш інтенсивним, коли H вказує в напрямку вектора нормалі N .

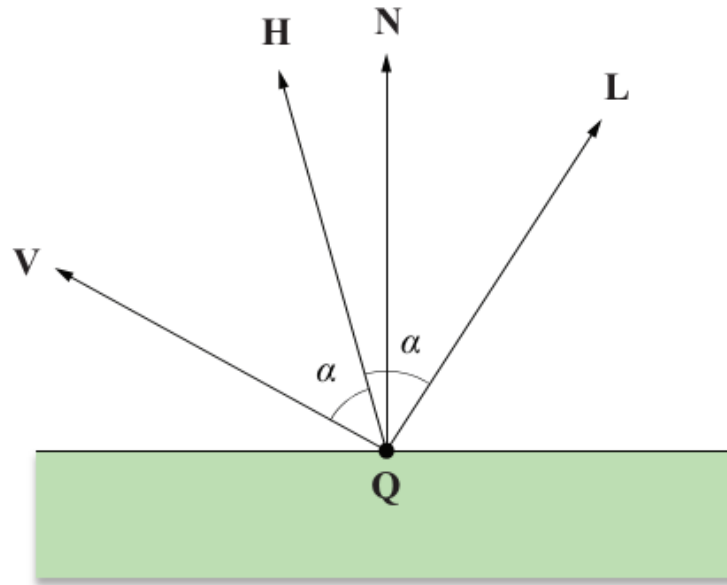


Рисунок 3.3 – Кут між нормальним вектором N і половинним вектором H також можна використовувати для визначення дзеркальної інтенсивності

Додавши внески від n джерел світла, ми можемо виразити дзеркальний компонент нашої формули освітлення як:

$$K_{specular} = S \sum_{i=1}^n C_i \max\{N \cdot H_i, 0\}^m (N \cdot L_i > 0),$$

де H_i — половинний вектор для i -ого джерела світла, заданого як:

$$H_i = \frac{L_i + V}{\|L_i + V\|} \quad (3.3)$$

3.2.2 Blinn-Phong шейдинг

Замість інтерполяції значень освітлення, обчислених у кожній вершині (вертексі), модель шейдинга (затінення) Blinn-Phong інтерполуює нормаль вершини N , напрямок на джерело світла L і напрямок на глядача V через трикутник і оцінює формулу освітлення для кожного пікселя [12]. Половина вектора H обчислюється за допомогою рівняння (3.3) для кожного пікселя.

Графічне обладнання, яке може виконувати складні обчислення для кожного

пікселя (процес, який називається затіненням пікселів або затіненням фрагментів), можна налаштувати для оцінки всього виразу в кожному пікселі, що складає грань трикутника:

$$K = K_{emission} + K_{diffuse} + K_{specular} =$$

$$= \varepsilon M + DTA + \sum_{i=1}^n C_i [DT(N \cdot L_i) + SG(N \cdot H_i)^m (N \cdot L_i > 0)]$$

Перевагою Blinn-Phong шейдинга над шейдингом Gouraud, коли інтерполяція значень освітлення обчислюється в кожній вершині на грані трикутника є те, що воно набагато краще моделює дзеркальність завдяки тому, що скалярний добуток $N \cdot H$ обчислюється для кожного пікселя. Коли гострий дзеркальний відблиск потрапляє у внутрішню частину трикутника, шейдинг Gouraud дає гірші результати, оскільки дзеркальний компонент, розрахований у вершинах трикутника, не відповідає справжнім значенням, що існують у інших місцях на поверхні трикутника.

Саме тому було прийнято рішення взяти за основу матеріалу для перевірки шейдингу моделі саме Blinn матеріал. У плагіні для Maya функціонал назначення матеріалів для перевірки шейдингу, а саме: назначення матеріалів “Low mat” та “High mat” використовується для аналізу цілісності моделей на наявність артефактів. Цей підхід забезпечує можливість оцінити, як світло взаємодіє з поверхнею моделі, не відволікаючись на текстури. Це дозволяє зосередитися на формі, освітленні та тіньових ефектах, які можуть виявити потенційні проблеми у геометрії, такі як неправильне з'єднання полігонів, небажані гострі краї та інші графічні дефекти, що часто приховані за текстурами. Використовуючи матеріал, основою якого є Blinn або Phong, можна легко ідентифікувати геометричні аномалії та розриви у полігонах, адже відсутність градієнтів у кольорах робить будь-які недоліки більш виразними. На рисунку 3.4 продемонстровано поганий шейдинг за допомогою матеріалу Blinn.

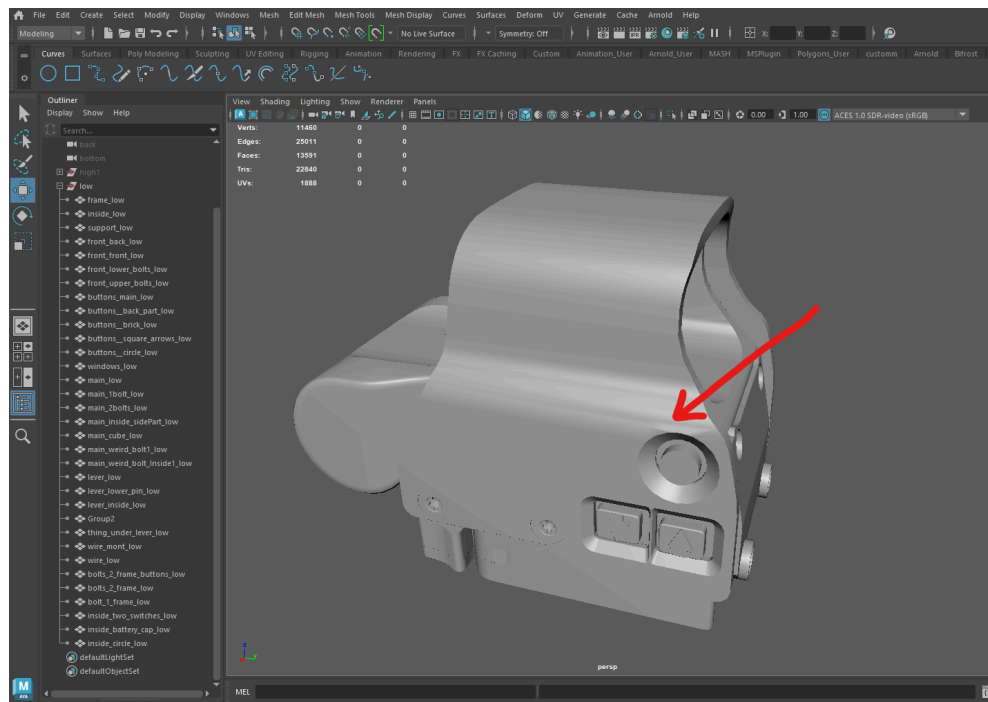


Рисунок 3.4 – Приклад поганого шейдингу

Для ідентифікації некоректного шейдингу було створено матеріал на основі Blinn – Low mat та High mat. Їх можна назначити на будь-який меш за допомогою натискання на відповідну кнопку в плагіні.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОГО ПРОДУКТУ

Детальний опис програмної реалізації плагіна для Autodesk Maya, що дозволяє створювати та маніпулювати дротами, кабелями або подібними конструкціями, включає аналіз архітектури плагіна, опис його компонентів, взаємодію між цими компонентами та спосіб інтеграції плагіна з Maya. Архітектура плагіна включає кілька ключових модулів:

Графічний інтерфейс користувача (GUI): створений за допомогою Qt Designer та інтегрований через PySide, GUI плагіна надає інтуїтивно зрозумілі візуальні інструменти для керування параметрами дротів і кабелів. Інтерфейс містить елементи управління, такі як слайдери, кнопки та інші віджети, які дозволяють користувачам налаштувати властивості об'єктів, такі як товщина, згладжування, щільність сітки, кількість секцій та твіст.

Логіка обробки даних: основний функціонал плагіна, написаний на Python, керує логікою створення та модифікації дротів. Ця частина коду використовує Maya Python API для маніпулювання сценою Maya та об'єктами [13].

MEL (Maya Embedded Language) використовується для автоматизації репетитивних завдань, управління сценами та маніпулювання об'єктами. У плагіні MEL використовується для виконання кількох критичних завдань, що забезпечують його функціональність та інтеграцію з інтерфейсом користувача, створеним через Qt Designer.

MEL застосовується для конвертації кривих в дроти, плагін включає функціональність для перетворення кривих, створених за допомогою інструментів Maya, в об'єкти, що імітують дроти. MEL використовується для запуску команд `polyToCurve`, яка конвертує полігональні меші в криві, що є необхідним для створення більш складних структур дротів з базових форм.

Також для реалізації специфічних вимог до форми та властивостей дротів, MEL використовується для додаткової обробки кривих. Команди, як `AttachBrushToCurves` та `convertCurvesToStrokes`, використовуються для

прикріплення стилів пензлів до кривих і перетворення їх у структури, які можна використовувати для деталізації візуалізації дротів.

Інтеграція MEL скриптів з Python кодом відбувається через `maya.cmds` інтерфейс, де Python скрипти можуть викликати MEL команди. Це дозволяє використовувати Python для керування логікою плагіна та використовувати MEL для виконання спеціалізованих завдань у Maya, оптимізуючи використання ресурсів та ефективність виконання завдань.

Такий підхід забезпечує максимальну гнучкість та продуктивність плагіна, дозволяючи точно контролювати кожен аспект обробки даних та взаємодії з користувачем, а також використовувати переваги обох мов програмування для досягнення кращих результатів.

4.1 Структура плагіна та його компоненти

Структура плагіна для Autodesk Maya включає кілька специфічних компонентів, які взаємодіють для забезпечення функціональності створення та маніпуляції дротами, кабелями та подібними конструкціями. Плагін реалізовано на Python із використанням MEL для специфічних завдань, що дозволяє оптимізувати робочі процеси та використовувати можливості Maya ефективно.

Одним з основних компонентів є клас `MayaUITemplate (GUI)`, він є центральним класом для користувацького інтерфейсу, що наслідує від `QtWidgets.QWidget`. Цей клас використовує Qt Designer для розміщення різних віджетів, таких як кнопки, слайдери, і чекбокси. Він відповідає за створення візуального інтерфейсу, який дозволяє користувачам взаємодіяти з плагіном, налаштовуючи параметри дротів, такі як розмір, товщина, кількість секцій, твіст та згладжування.

До компонентів також належать функції створення кривих і маніпуляції ними. `create_ep_curve()`, `create_bezier_curve()`, `create_pen_curve()`, та `create_edge_curve()` є методами в класі `MayaUITemplate`, які ініціюють інструменти

створення кривих в Maya за допомогою Python команд [14]. Ці методи забезпечують можливості для користувачів створювати криві, які потім можуть бути перетворені в моделі дротів або кабелів з використанням додаткових параметрів, налаштованих через GUI.

MEL команди для оптимізації дій, такі як: `AttachBrushToCurves`, `convertCurvesToStrokes`, що викликаються з Python за допомогою `maya.mel.eval()`, є прикладами використання MEL для виконання завдань, оптимізованих під внутрішній механізм Maya. Ці команди використовуються для автоматизації процесів, таких як перетворення кривих у структури, що можуть використовуватися для створення дротів.

Користувацький інтерфейс плагіна слугує основним зв'язком між користувачем та його функціональною логікою, відіграючи роль мосту, який перетворює взаємодію користувача з графічними елементами управління на конкретні дії в середовищі Maya.

Зміни, ініційовані через GUI, такі як налаштування параметрів дротів, передаються через Python API, що дозволяє динамічно оновлювати властивості і зображення дротів у сцені Maya.

Водночас, Python слугує основою для викликання MEL скриптів, забезпечуючи глибоку інтеграцію з Maya і використання специфічних для цієї платформи оптимізацій та функціональних можливостей, які покращують загальну продуктивність і користувацький досвід.

4.2 Реалізація функцій створення кривих та маніпуляцій з дротами

Реалізація функцій створення кривих та маніпуляції з дротами у плагіні для Autodesk Maya базується на використанні Python та MEL для досягнення високої гнучкості та точності у роботі з 3D моделями. Ці функції є фундаментальними для основного призначення плагіна — дозволяти користувачам легко створювати,

налаштовувати та маніпулювати дротами з високим рівнем контролю над їх властивостями.

Функціональність створення кривих включає кілька методів, кожен з яких активує певні інструменти Maya для генерації кривих, які потім можуть бути використані як основа для моделювання дротів:

1. EP Curve Tool: функція `create_ep_curve()` ініціює інструмент створення EP кривих в Maya, який дозволяє користувачам векторно створювати гладкі криві. Цей інструмент корисний для деталізованого контролю над формою дроту;

2. Bezier Curve Tool: метод `create_bezier_curve()` активує інструмент для створення кривих Безьє, забезпечуючи користувачам більш інтуїтивний спосіб моделювання кривих з контрольними точками, що відмінно підходить для створення складних кабельних конфігурацій;

3. Pen Curve Tool: використання `create_pen_curve()` дозволяє користувачам малювати криві вільною рукою в середовищі Maya, надаючи можливість максимально швидкого та динамічного проектування дротів;

4. Poly to Curve Tool: `create_edge_curve()` використовується для перетворення полігональних країв у криві, що може бути особливо корисним при роботі з існуючими полігональними мешами для швидкого створення кривих, що відображають контури цих мешів.

Після створення кривих, плагін має ряд методів для маніпуляції та налаштування дротів. Функція `set_random_material()` дозволяє користувачам швидко призначати випадкові матеріали та кольори для створених дротів, що поліпшує візуалізацію та допомагає в розрізненні об'єктів сцени. Налаштування параметрів дротів відбувається за допомогою використання слайдерів та інших елементів управління в GUI дозволяє користувачам налаштовувати такі параметри, як товщина, гнучкість, довжина та інші фізичні атрибути дротів в реальному часі модифікуючи властивості об'єктів.

4.3 Інтеграція інтерфейсу користувача із функціональністю плагіна

Інтеграція інтерфейсу користувача з функціональною логікою плагіна для Autodesk Maya полягає в тісному зв'язку між графічними компонентами, що користувачі використовують для введення даних, та внутрішніми механізмами, які виконують обробку цих даних і впливають на об'єкти в сцені Maya. Ось як ця інтеграція реалізована у плагіні:

Кожен слайдер у графічному інтерфейсі прямо зв'язаний з певним атрибутом дроту в Maya, наприклад, слайдери для масштабу, скейлу, твісту та щільності. Коли користувач змінює положення слайдера, відповідний атрибут об'єкта в Maya оновлюється миттєво через PySide, що забезпечує двосторонню синхронізацію. Наприклад, коли користувач зміщує слайдер 'Scale', Python-функція `scale_val()` викликається, яка в свою чергу використовує Maya Python API для оновлення атрибуту `globalScale` відповідного дроту.

Плагін містить спеціалізовані функції, як `update_spin_box_scale()` та `update_slider_scale()`, що синхронізують значення між слайдерами та спінбоксами. Це забезпечує узгодженість введених даних від користувача та їх відображення, уникнення помилок або розбіжностей у даних.

Застосування MEL для оптимізації дій з маніпуляціями, які потребують високої продуктивності. Деякі дії, такі як перетворення кривих у дроти, виконуються з використанням MEL для оптимізації продуктивності. Команди MEL, як `AttachBrushToCurves` та `convertCurvesToStrokes`, використовуються для виконання комплексних маніпуляцій з об'єктами, що забезпечують плавну інтеграцію та високу швидкість обробки. Реакція на події інтерфейсу, такі як натискання кнопок або зміна параметрів через інтерфейс, часто викликають MEL скрипти, що дозволяє виконувати складні задачі безпосередньо в середовищі Maya з мінімальними затримками.

Ця інтеграція між інтерфейсом користувача і функціональністю плагіна забезпечує ефективне та інтуїтивно зрозуміле середовище для моделювання,

дозволяючи користувачам використовувати складні інструменти з легкістю і точністю.

4.4 Тестування і налагодження плагіну

Функціональним тестуванням було перевірено плагін у реальних сценаріях використання в Maya, зосереджуючись на користувацькому досвіді та відповідності функціональних вимог. Це включає тестування реакцій програми на команди користувача, точність візуалізації дротів та їх взаємодії з іншими об'єктами сцени.

Продуктивність при великій кількості дротів: тестування включало створення великої кількості дротів у сцені, щоб зрозуміти, як плагін впорається з високим навантаженням і чи збереже він свою реактивність та стабільність.

Для більш глибокого аналізу і налагодження використовували інструменти відладки, які дозволяють крок за кроком переглядати виконання коду і зміну станів у реальному часі, ідентифікувати точні моменти збоїв або неправильної поведінки коду.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПЛАГІНОМ

Взаємодія користувача з плагіном для створення та маніпуляції дротами і кабелями в Autodesk Maya забезпечується через інтерфейс користувача (рисунок 5.1), який включає низку функціональних інструментів та елементів управління.

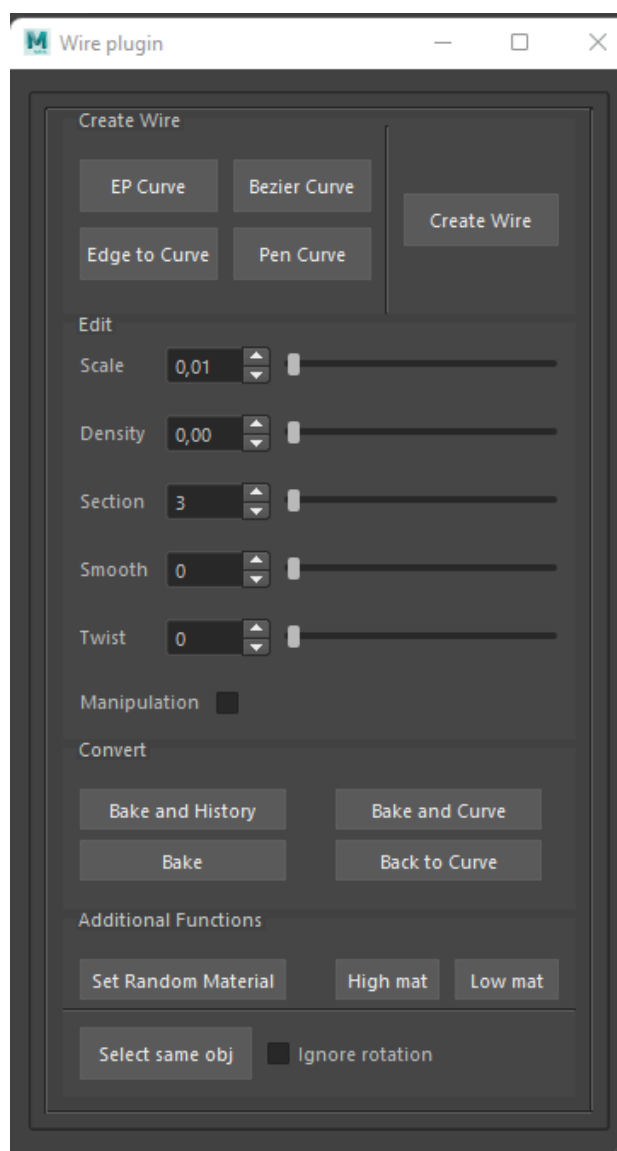


Рисунок 5.1 – Інтерфейс плагіна

Цей підхід дозволяє користувачам ефективно використовувати плагін, незалежно від їх рівня досвіду в роботі з Maya, і сприяє підвищенню продуктивності роботи з 3D моделями.

5.1 Інструкції з інсталяції, вимоги до системи для встановлення та використання плагіна

Плагін написаний для версії програми Autodesk Maya 2017 і вище. Для коректного встановлення плагіна необхідно перенести архів “wire_plugin” до папки Autodesk, яка знаходиться по наступному шляху: C:\Program Files\Autodesk, після цього розархівовуємо архів. Наступним кроком необхідно запустити Autodesk Maya, заходимо в Script Editor, створюємо вікно Python (рисунок 5.2).

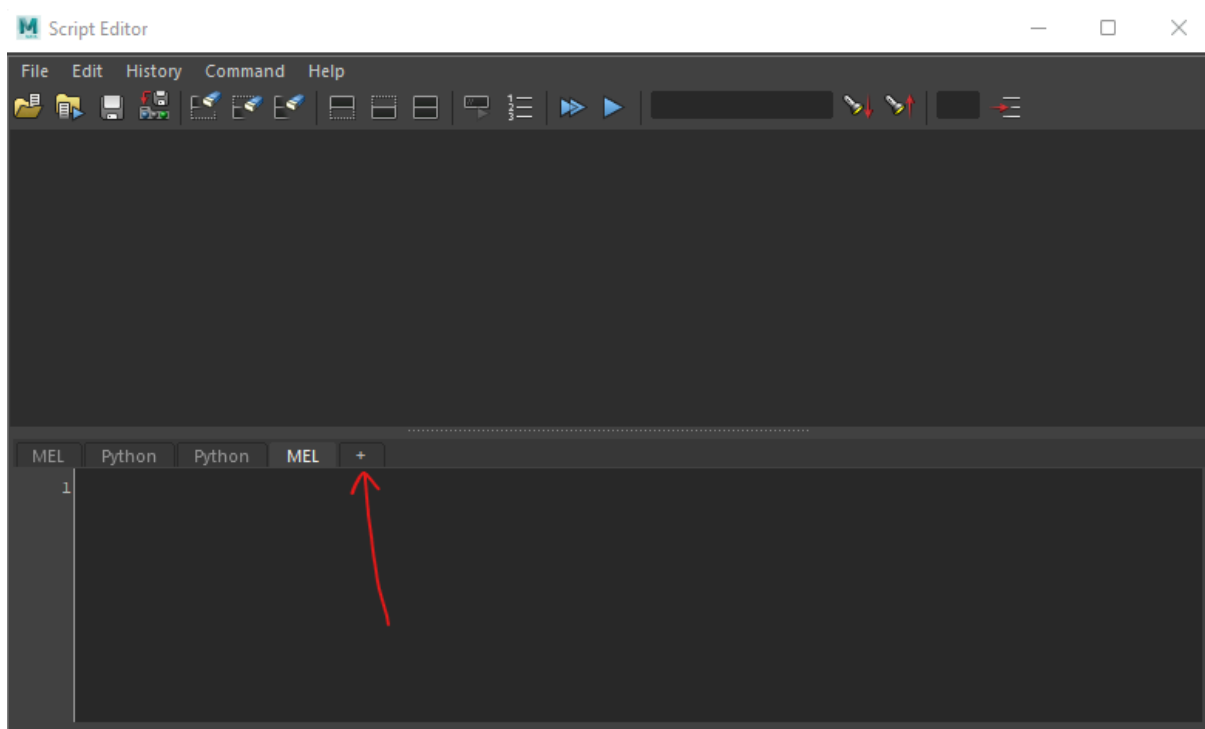


Рисунок 5.2 – Створення нового вікна Python

Далі обираємо в архіві файл wire.py та перетягуємо у вікно Python. Потім у вікні Script Editor натискаємо “File” у верхньому лівому кутку обираємо “Save Script to Shelf...” (рисунок 5.3) та даємо назву.

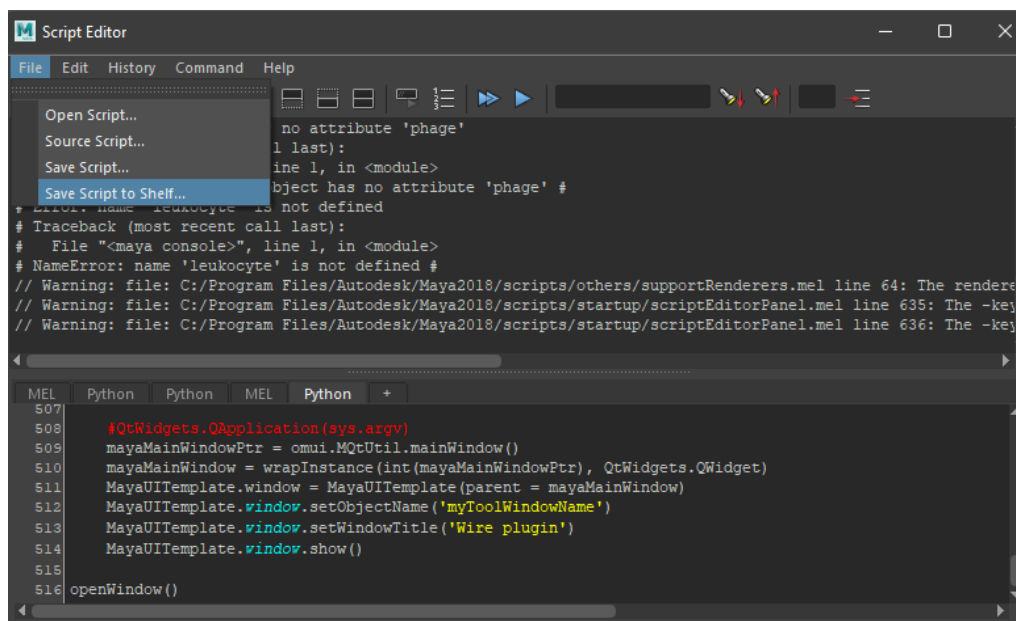


Рисунок 5.3 – Збереження плагіна на полиці швидкого доступу

Таким чином, на полиці швидкого доступу з'являється плагін (рисунок 5.4).

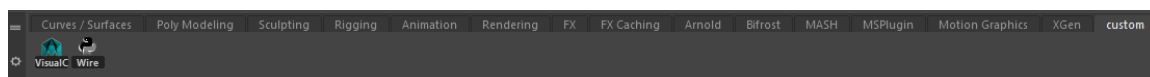


Рисунок 5.4 – Відображення плагіна на полиці

Надалі, щоб викликати плагін потрібно просто натиснути на його іконку на полиці, також за бажанням можна змінити іконку на будь-яку картинку для того, щоб чітко розрізняти Wire плагін від інших.

5.2 Опис інтерфейсу користувача та інструкція з використання розробленої системи

Інтерфейс плагіна для Autodesk Maya структурований таким чином, щоб максимально спростити процес створення і маніпуляції дротами, кабелями та

подібними конструкціями. Це досягається завдяки логічній організації інструментів і функцій, що забезпечують користувачам інтуїтивно зрозумілу взаємодію.

Створення кривої реалізується наступними кнопками: EP Curve, Bezier Curve, Pen Curve, Edge to Curve: ці кнопки активують різні інструменти для створення кривих в Maya, які служать основою для майбутніх дротів. Розділ налаштувань Edit має наступні параметри: Scale, Density, Section, Smooth, Twist: слайдери для регулювання параметрів дротів, що дозволяють користувачам точно налаштувати властивості створених дротів, включаючи їх розмір, щільність, кількість секцій, гладкість та скручування. Нехай оберемо інструмент Bezier Curve, створимо криву (рисунок 5.5).

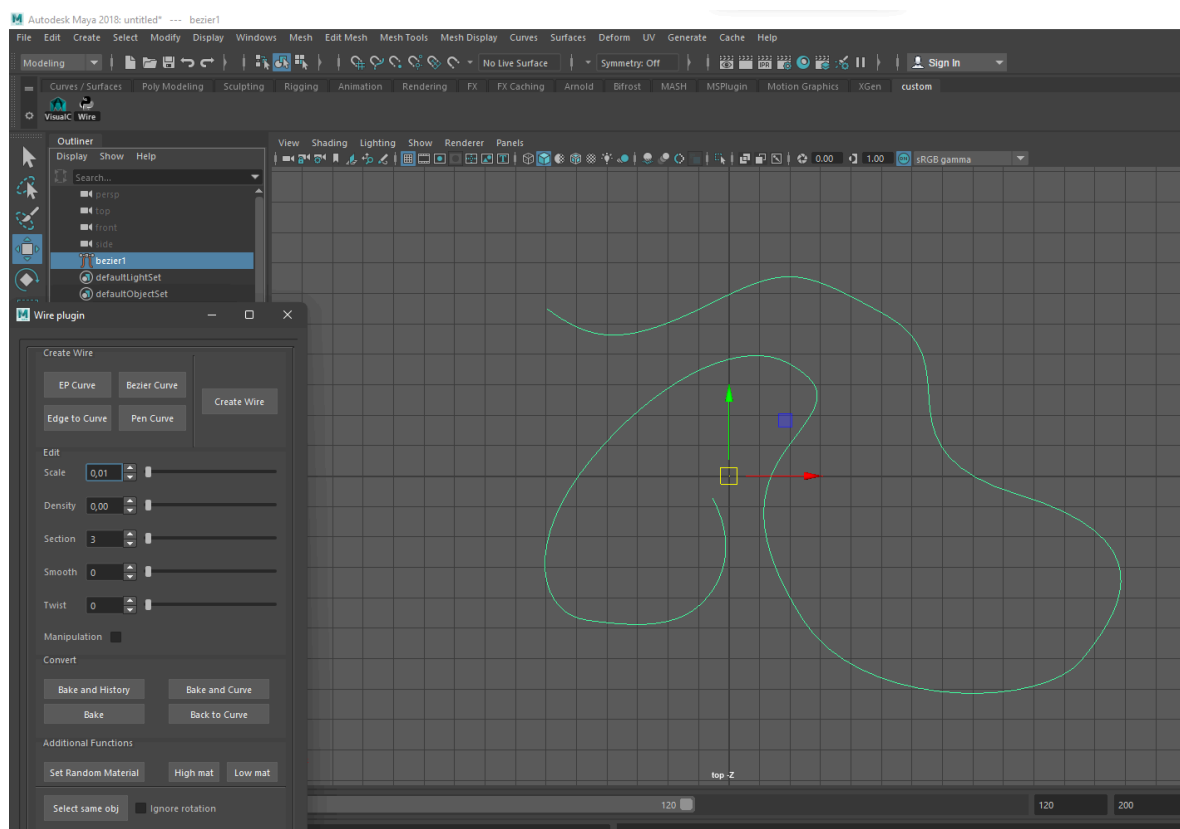


Рисунок 5.5 – Створення кривої

Щоб прикріпити об'єкт stroke до кривої маємо спочатку обрати криву, потім натиснути на кнопку Create Wire. В аутлайнері бачимо, що з'явився об'єкт stroke

(рисунок 5.6). У вьюпорті бачимо, що скейл об'єкта замалий, збільшимо розмір нашого кабелю за допомогою меню інтерактивного редагування Edit. Для зміни розміру можна рухати слайдер, змінити значення стрілочками спінбоксу або ж безпосередньо вписавши потрібне значення у спінбокс. Збільшимо розмір до значення 5 (рисунок 5.7). Розділ налаштувань Edit має наступні параметри: Scale, Density, Section, Smooth, Twist: слайдери для регулювання параметрів дротів, що дозволяють користувачам точно налаштувати властивості створених дротів, включаючи їх розмір, щільність, кількість секцій, гладкість та скручування.

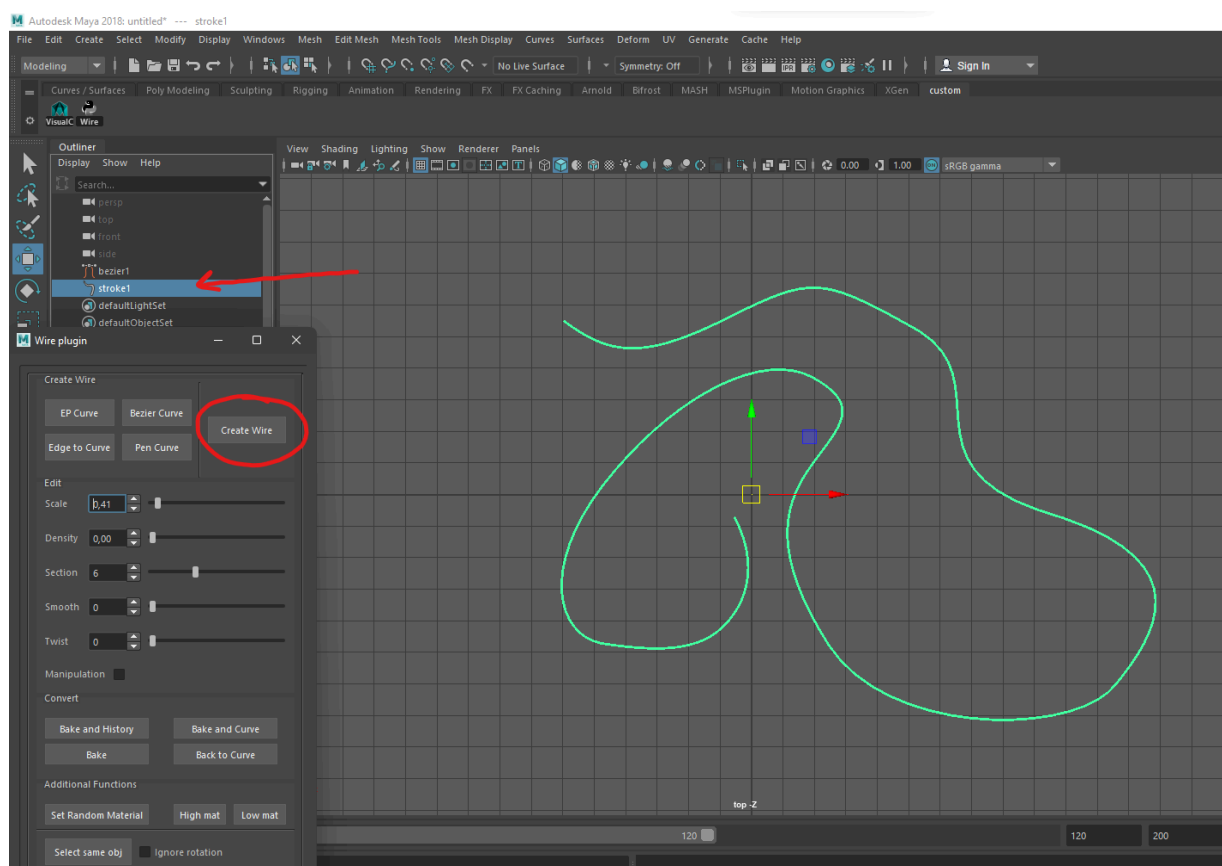


Рисунок 5.6 – Прикріплення об'єкту stroke до кривої за допомогою кнопки Create Wire

Скейл вже виставили, тепер потрібно змінити density, якщо щільність сітки не влаштовує. Аналогічно зі скейл можемо різними способами змінити цей

параметр, а саме: через слайдер, стрілочки спінбокса або вписати значення в спінбокс.

Важливим параметром є Section, який відповідає за кількість секцій, по замовчуванню значення стоїть на 3, максимально можна виставити 12. Було створено обмеження 12, так як зазвичай кабелі та дроти це об'єкти невеликого розміру, відповідно, наприклад, для лоу-полі моделі буде достатньо 6 секцій якщо кадр на рендері буде загальним, для приближених рендерів (клоузапів) можна виставити 12, цього буде достатньо для більшості випадків. В нашому випадку оберемо значення 6 (рисунок 5.7).

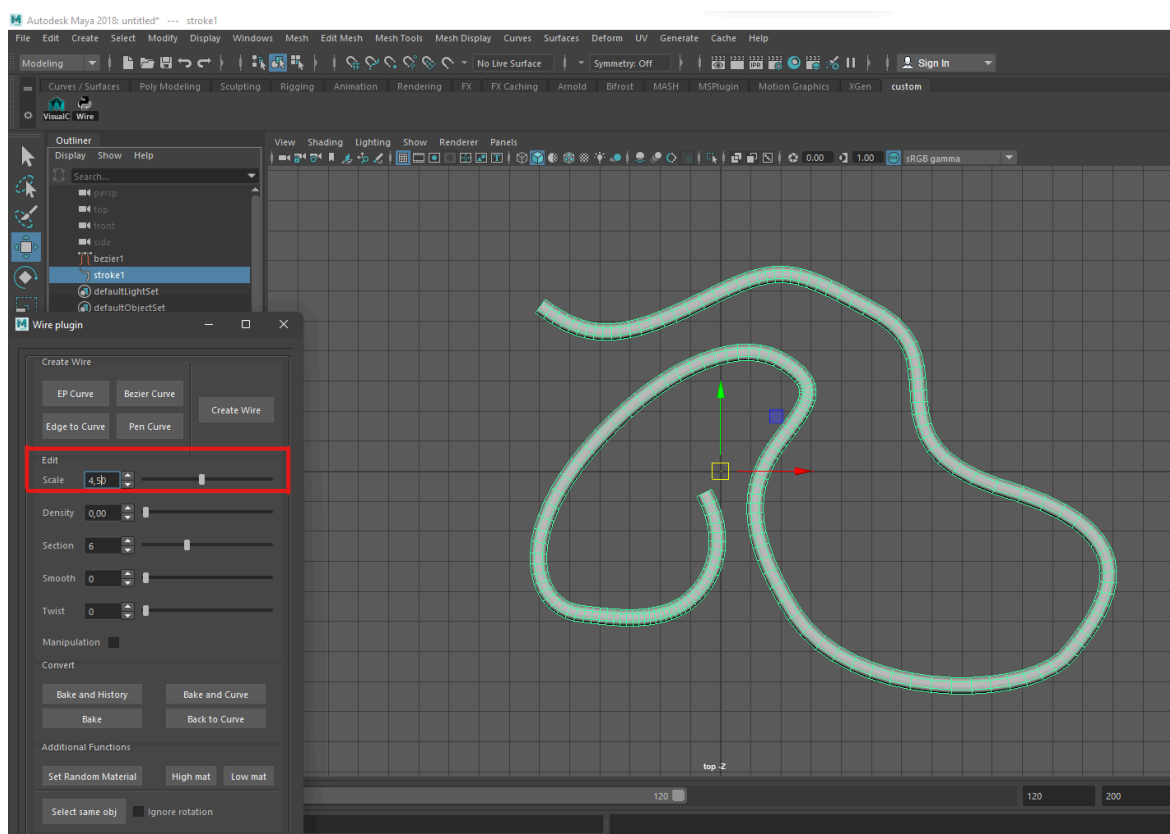


Рисунок 5.7 – Зміна розміру дроту, встановлення значення кількості секцій 6

Якщо розглядати пайплайн розробки моделей для геймдеву, враховуючи, що дріт чи кабель може бути не один, а безліч. На вищеописаний можливий варіант моделі, де буде чимало дрових конструкцій під занадто багато полігонів, в

такому випадку буде розумно виставити параметр Section на значення 6 для оптимізації моделі.

Також є параметр Smooth, який дозволяє згладити певні повороти кривої для забезпечення гарного шейдингу й відповідної гладкості. Змінимо значення Smooth на 36 за допомогою слайдери (рисунок 5.8), бачимо, що дріт став більш гладким та трохи відійшов від початково заданої кривої. Таким чином, якщо в нас занадто різкі повороти кривої це можна виправити параметром Smooth, інакше будуть проблеми з геометричною сіткою в місцях крутих згинів кривої.

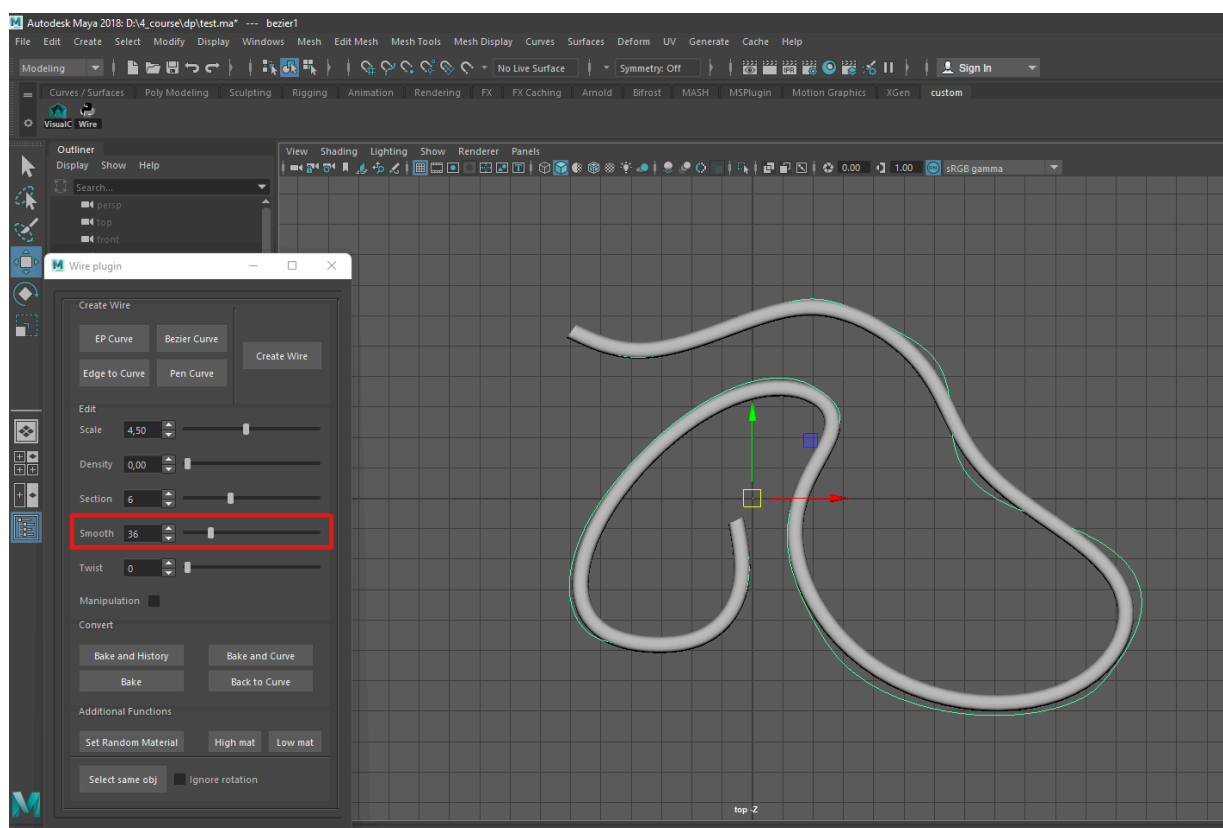


Рисунок 5.8 – Зміна значення Smooth

Параметр Twist перекручує сітку, може бути застосований для отримання ефекту перекручування, виставимо значення 54, на рисунку 5.9 бачимо, що геометрична сітка змінилася.

Різниця в кінцевому результаті між створенням кривої за допомогою EP Curve, Bezier Curve чи Pen Curve немає, єдиний нюанс в тому, що Pen Curve

дозволяє намалювати криву в рази швидше, проте гладкість кривої буде гірша в порівнянні з інструментами EP Curve та Bezier Curve, проте це можна компенсувати параметром Smooth.

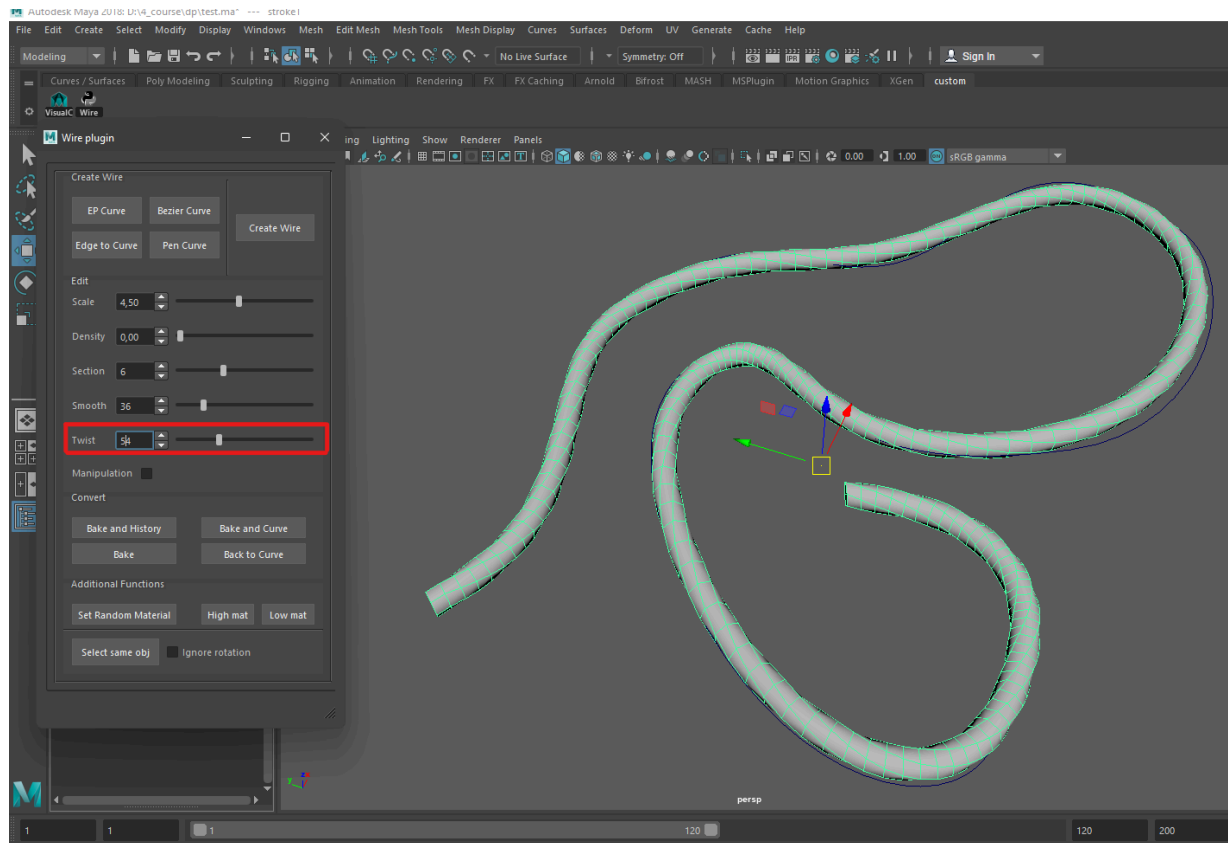


Рисунок 5.9 – Зміна параметру Twist

Використовуючи Edge to Curve результат буде інший, з назви інструменту зрозуміло, що він перетворює будь-яке ребро або ребра на криву чи криві відповідно. Результат створення кривої використовуючи Edge to Curve показано на рисунку 5.10. Наступні кроки аналогічні до вищеописаних, створюємо об'єкт stroke за допомогою кнопки Create Wire та налаштовуємо параметри.

Також для зручної маніпуляції кривої є Manipulation Checkbox, він дозволяє активувати або деактивувати маніпуляцію кривої.

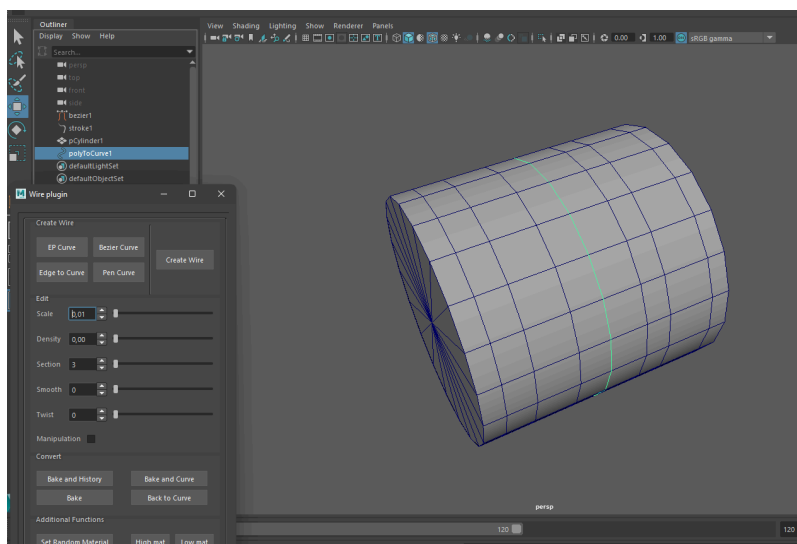


Рисунок 5.10 – Обираємо ребро й створюємо з нього криву

Так як переміщення об'єкта stroke буде роз'єднувати криву та об'єкт stroke (рисунок 5.11), це створить певні незручності при можливому редагуванні кривої за допомогою контрольних точок.

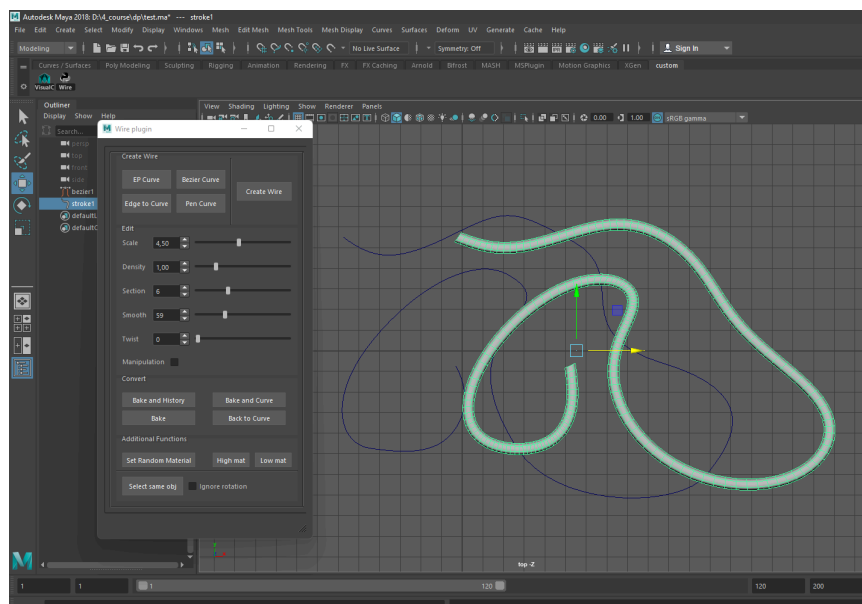


Рисунок 5.11 – Переміщення кабелю без відміченого чекбокса Manipulation

Для збереження єдності між кривою та дротом є вищезгаданий чекбокс Manipulation.

Відмітимо його та спробуємо перемістити наш дріт. При спробі обрати наш дріт обирається крива, а не об'єкт stroke, що і забезпечує одночасне переміщення. Як бачимо на рисунку 5.12, при відміченому чекбоксі переміщується і крива, і об'єкт stroke одночасно.

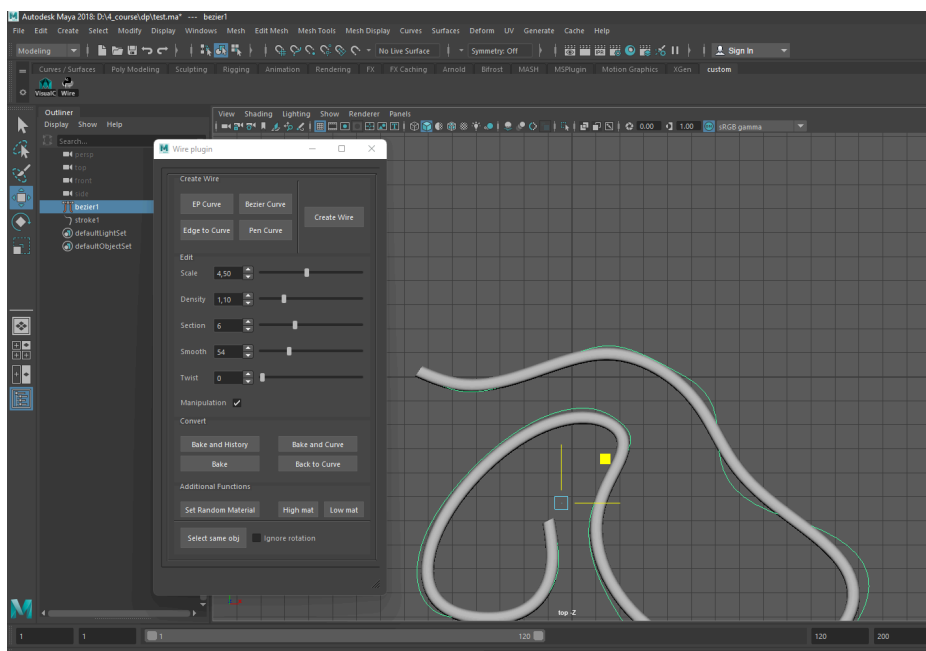


Рисунок 5.12 – Одночасне переміщення кривої та об'єкта stroke

Конвертація відбувається за допомогою наступних кнопок: Bake and History, Bake, Back to Curve: кнопки для конвертації дротів у статичні меші. Bake and History конвертує stroke у меш та залишає криву та можливих змін параметрів. Оберемо нашу дріт та натиснемо Bake and History для конвертації, результат бачимо на рисунку 5.13. Конвертація успішна, адже можемо працювати з кривою, як з мешем, наприклад, виділити грані, як показано на рисунку 5.13. Також бачимо, що в аутлайнері зберігається початкова крива та stroke, є можливість обрати криву та змінити її, меш буде змінюватися одночасно. Перетягнемо одну з контрольних точок кривої та зможемо переконатися, що меш дроту також рухається за кривою (рисунок 5.14).

Bake конвертує stroke в меш та видаляє всю історію, тобто залишається тільки меш.

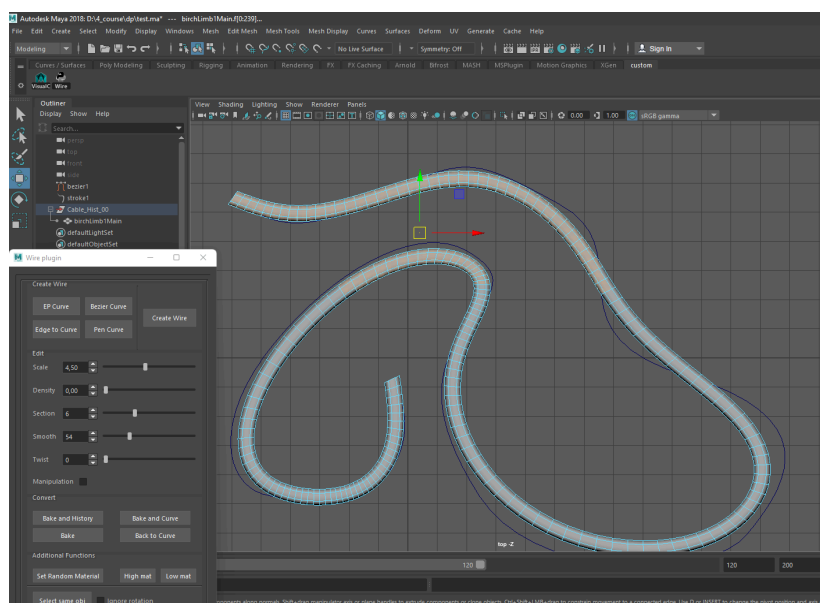


Рисунок 5.13 – Конвертація stroke в меш за допомогою Bake and History

Bake and Curve конвертує stroke в меш, видаляє історію, але також залишає криву, на відміну від Bake and History, крива не буде прив'язана до мешу.

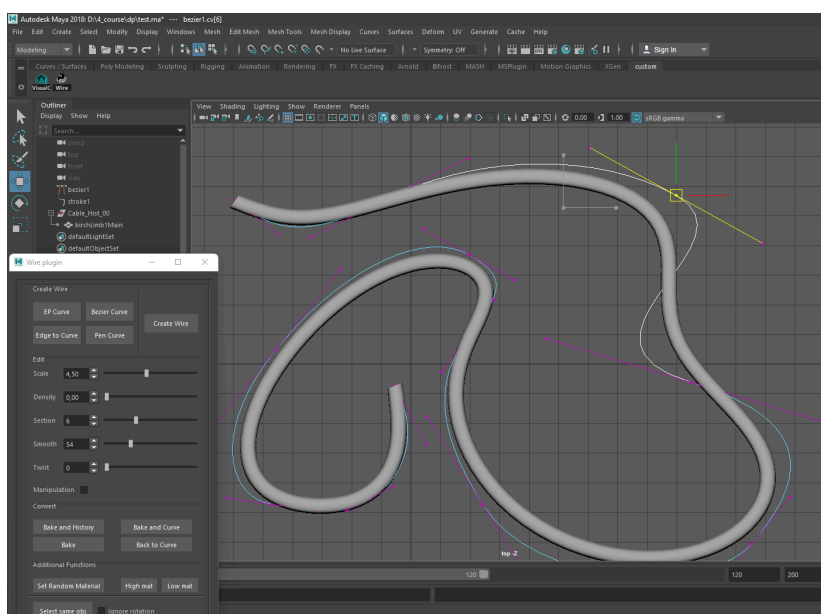


Рисунок 5.14 – Зміна кривої та одночасна зміна мешу кабеля

Back to Curve створює з мешу криву, яку в подальшому можна використовувати для створення подібних конструкцій.

Також в плагіні реалізовані додаткові функції, а саме: Set Random Material задає випадковий матеріал вибраним дротам, або будь-яким іншим об'єктам що спрощує процес візуалізації та відокремлення різних об'єктів. Також може застосовуватися для швидкого створення ID мап. На рисунку 5.15 було обрано малий циліндр та назначений рандомний матеріал за допомогою функції Set Random Material, це в подальшому спростить роботу з текстурами, адже буде можливість швидкого вибору об'єкта через ID карту.

Low mat назначає спеціальний матеріал на модель для перевірки шейдингу. За допомогою кнопки High mat можна назначити матеріал на “хай полі” модель.

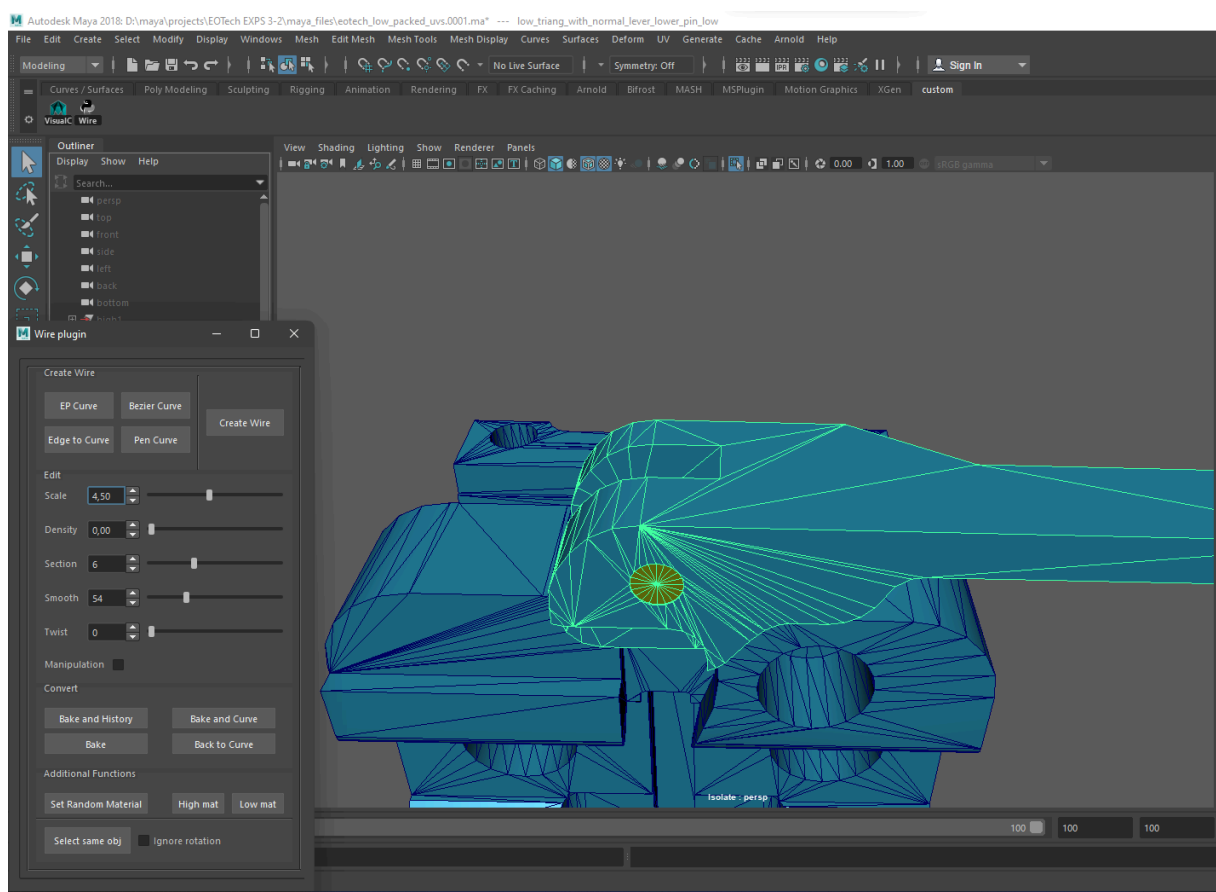


Рисунок 5.15 – Назначення випадкового кольору

Select Same Obj дає можливість швидко вибирати об'єкти з однаковими параметрами. Чекбокс Ignore rotation дозволяє ігнорувати будь-який поворот об'єкта при виборі однакових мешей. Ця функція може бути корисна, наприклад,

для текстурування лоу-полі гітари. Щоб не виділяти кожну струну одна за одною, виділяємо тільки одну та натискаємо на кнопку Select Same Obj, далі можна наприклад задати їм один колір кнопкою Set Random Material для створення ID карти, або обравши всі струни застосувати авторозгортку. Таким чином економимо дорогоціний час, який є важливим критерієм при створенні будь-якого асета, чи то для гри, чи для візуалізації інтер'єру.

Наприклад, створимо 10 однакових кубів та спробуємо обрати їх всіх за допомогою кнопки Select Same Obj (рисунок 5.16).

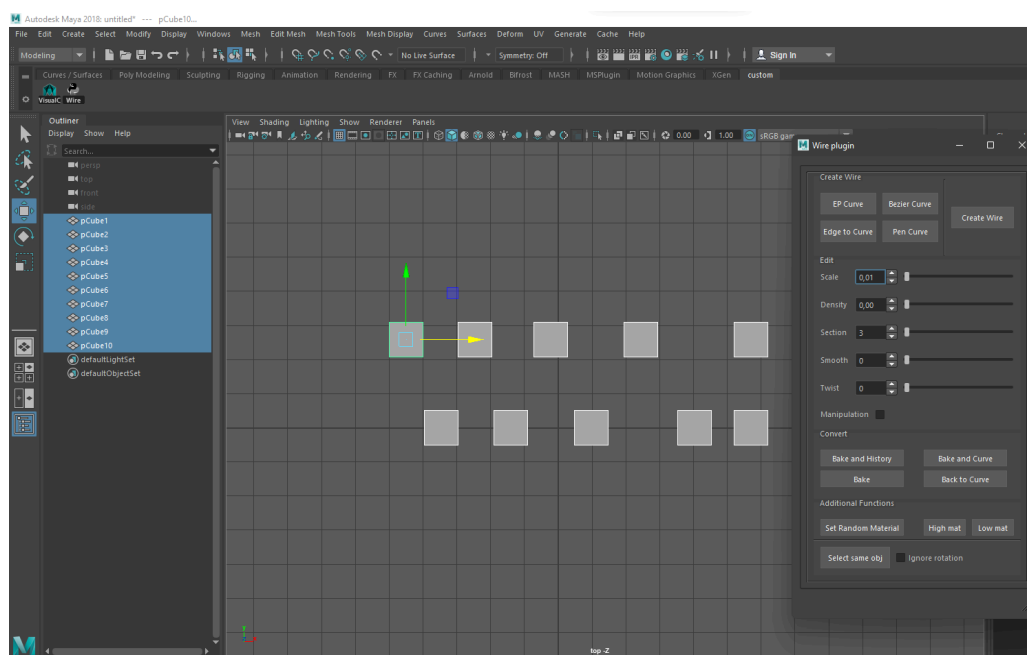


Рисунок 5.16 – Результат вибору однакових об'єктів

Для експерименту змінимо розташування двох вершин одного з кубів та перевіримо, чи коректно буде працювати вибір всіх однакових об'єктів. Як бачимо, при виборі куб, як початкового об'єкта, виділяються тільки куби (рисунок 5.17), отже, все працює вірно. Якщо у нас об'єкти будуть мати змінену орієнтацію, тоді будуть обиратися тільки ті об'єкти, в яких в параметрах ротеиту стоїть нульове значення (рисунок 5.18).

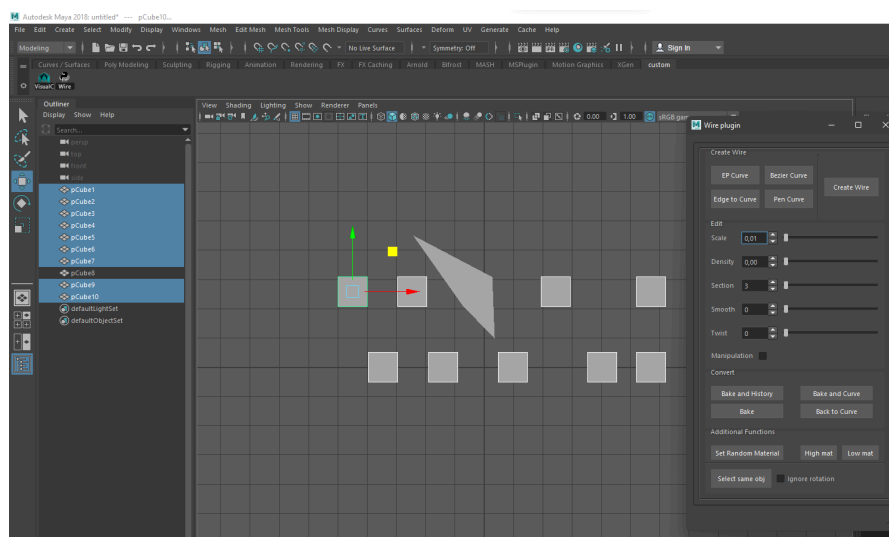


Рисунок 5.17 – Демонстрація функції виділення однакових об'єктів

На рисунку 5.18 бачимо, що обрано було тільки ті куби, які не мають зміненої орієнтації в просторі, при цьому також бачимо, що чекбокс Ignore Rotation не є відміченим.

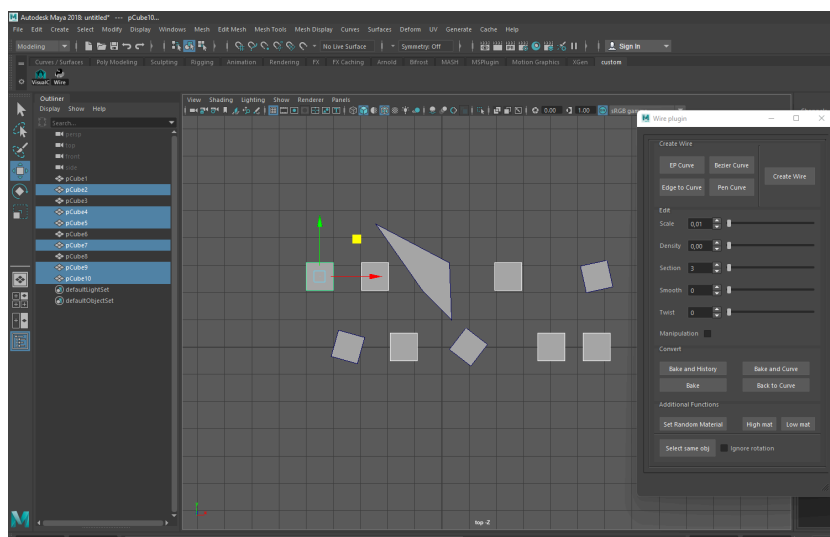


Рисунок 5.18 – Виділення ідентичних мешів з урахування орієнтації об'єктів у просторі

Якщо є необхідність у виділення всіх однакових об'єктів незважаючи на значення ротейту, тоді для цього є вищезазначений відповідний чекбокс Ignore

rotation, відмітимо його та знову спробуємо, обираючи за початковий об'єкт куб, виділити однакові об'єкти (рисунк 5.19). В результаті бачимо, що були виділені всі однакові куби.

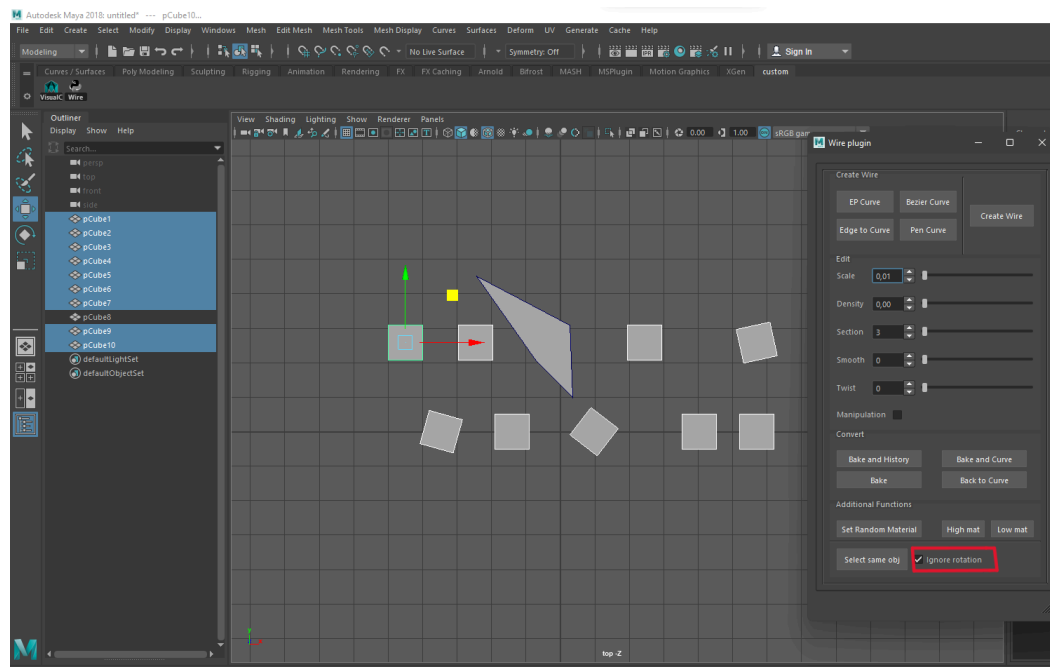


Рисунок 5.19 – Виділення ідентичних мешів без урахування орієнтації об'єктів у просторі

Отже, окрім функціоналу по створенню дротів, що є основною проблемою, яку вирішує плагін, також були реалізовані корисні вищезазначені функції, які доволі ефективно економлять час на розробку моделі.

ВИСНОВКИ

У ході кваліфікаційної роботи було проведено ретельний аналіз різних методів та підходів до створення складних тривимірних об'єктів у програмному середовищі Autodesk Maya. Було розроблено плагін, який значно спрощує процеси створення, маніпуляції та управління тривимірними кабелями та дротами, підвищуючи точність та ефективність моделювання. Крім того, були визначені ключові технології та інструменти, що необхідні для інтеграції плагіну в середовище Maya, щоб забезпечити його високу продуктивність та сумісність.

У роботі також описано інструменти розробки, які використовувалися для реалізації плагіну, зокрема мову програмування Python, Maya Embedded Language (MEL), Maya Python API та інструмент Qt Designer для створення графічного інтерфейсу користувача. Для користувачів були надані відповідні інструкції з встановлення, перевірки працездатності та огляду функціональних можливостей системи.

У результаті проведених досліджень та розробки плагіну можна зробити наступні висновки.

1. Було проведено аналіз сучасних методів створення кабелів і дротів в Autodesk Maya, що дозволило оцінити рівень досягнень у даній галузі та вибрати найбільш ефективні рішення для подальшого використання.

2. Було розроблено програмний плагін для створення кабелів і дротів в Autodesk Maya, який демонструє високу точність та швидкодію у процесах моделювання.

3. Результати експериментів показали, що використання передових програмних інструментів і алгоритмів сприяє покращенню якості моделювання.

4. Розроблений плагін може бути застосований у різних галузях, таких як кінематограф, відеоігри, архітектурна візуалізація та інженерне моделювання, де вимагається деталізація та точність тривимірних конструкцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Maya rigging wiki: веб-сайт. URL: <https://sites.google.com/site/mayariggingwiki/python/basic-structure?authuser=0> (дата звернення: 15.04.2024).
2. Vernon C. Python Scripting for Maya Artists: веб-сайт. URL: <https://www.chadvernon.com/python-scripting-for-maya-artists/> (дата звернення: 15.04.2024).
3. MEL Commands documentation: веб-сайт. URL: https://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__Commands_index_h (дата звернення: 16.04.2024).
4. Qt Designer Manual: веб-сайт. URL: <https://doc.qt.io/qt-6/qtdesigner-manual.html> (дата звернення: 15.04.2024).
5. Qt Designer: веб-сайт. URL: <https://www.pythontutorial.net/pyqt/qt-designer/> (дата звернення: 17.04.2024).
6. Working with PySide in Maya: веб-сайт. URL: https://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__files_GUID_3F96AF53_A47E_4351_A86A_396E7BFD6665_hm (дата звернення: 18.04.2024).
7. Galanakis R. Practical Maya Programming with Python. Birmingham, UK: Packt Pub Ltd, 2014. 352 p.
8. Lengyel E. Mathematics for 3D Game Programming and Computer Graphics, Third Edition. Boston, MA, USA: Cengage Learning PTR, 2011. 566 p.
9. Gallier J. Curves and Surfaces in Geometric Modeling: Theory & Algorithms. San Francisco, California, USA: Morgan Kaufmann Publishers, 2000. 502 p.
10. Hearn D., Baker P., Carithers W. Computer Graphics with Open GL, 4th edition. Boston, MA, USA: Pearson, 2010. 819 p.
11. Advanced Lighting: веб-сайт. URL: <https://learnopengl.com/Advanced-Lighting/Advanced-Lighting> (дата звернення: 19.04.2024).

12. Shading: підручник. Toronto: Alias. 394 с. URL: <https://courses.cs.washington.edu/courses/cse459/06wi/help/mayaguide/Complete/Shading.pdf> (дата звернення: 19.04.2024)

13. Autodesk Maya API | White Paper: веб-сайт. URL: https://images.autodesk.com/adsk/files/maya_api_whitepaper.pdf (дата звернення: 16.04.2024).

14. Python Commands for Autodesk Maya: веб-сайт. URL: https://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__CommandsPython_index_html (дата звернення: 17.04.2024).

ДОДАТОК А

Реалізація основних функцій плагіна

Програмні засоби:

- мова програмування – Python;
- мова для написання скриптів – MEL;
- модуль Python для доступу до команд MEL – cmds;
- безпосередня взаємодія з Autodesk Maya – Maya API;
- створення інтерфейсу – Qt Designer.

Програмний код:

```
# SET correct brush -----
version = mc.about(v=True) # command to get Maya version you are now
running
OS = mc.about(os=True)
if OS == 'win64':
    if version == '2018':
        mel.eval('visorPanelBrushPressCallback files\VisorEd "C:/Program
Files/Autodesk/Maya2018/Examples/Paint_Effects/Trees/birchLimb.mel";')
        mel.eval('setToolTo $gMove;')
    # Set default settings for the tube
def set_default_tube_settings(self):
    mc.setAttr("birchLimb.globalScale", 1)
    #mc.setAttr("birchLimb.brushWidth", 0.1)
    mc.setAttr("birchLimb.forwardTwist", 0)
    mc.setAttr(("birchLimb.color1"), 0.51, 0.51, 0.51, type='double3')
    mc.setAttr(("birchLimb.specularColor"), 0, 0, 0, type='double3')
    mc.setAttr(("birchLimb.tubeSections"), 3)
def apply_blinn_material_low(self):
    selected_objects = cmds.ls(selection=True)
```

```

if not selected_objects:
    cmds.warning("No object selected.")
    return
material_name = 'low_Material'
if not cmds.objExists(material_name):
    material_name = cmds.shadingNode('blinn', asShader=True,
name=material_name)
    cmds.setAttr(material_name + ".color", 0.440, 0.440, 0.440,
type='double3')
    cmds.select(selected_objects)
    cmds.hyperShade(assign=material_name)

def apply_blinn_material_high(self):
    selected_objects = cmds.ls(selection=True)
    if not selected_objects:
        cmds.warning("No object selected.")
        return
    material_name = 'high_Material'
    if not cmds.objExists(material_name):
        material_name = cmds.shadingNode('blinn', asShader=True,
name=material_name)
        cmds.setAttr(material_name + ".color", 0.146, 0.555, 0.750,
type='double3')
        cmds.select(selected_objects)
        cmds.hyperShade(assign=material_name)
def create_ep_curve(self):
    mc.EPCurveTool()
def create_bezier_curve(self):
    mc.CreateBezierCurveTool()
def create_pen_curve(self):

```

```

mc.PencilCurveTool()
def create_edge_curve(self):
    mc.polyToCurve(form= 3, degree= 1, conformToSmoothMeshPreview= 1)

def attach_cable(self):
    mel.eval('AttachBrushToCurves')
    mel.eval('convertCurvesToStrokes')
    mel.eval('setToolTo $gMove;')

def closeWindow(self):
    """
    Close window.
    """
    print ('closing window')
    self.destroy()

#-----Scale-----
def scale_val(self):
    slider_value = self.ui.Scale_slider.value()
    scaled_value = slider_value / 10.0 # Convert to floating point for Maya
usage
    selection = mc.ls(sl=True, fl=True, dag=True, type='stroke')
    buffer = mc.listConnections(selection, d=True, scn=True, type='brush')
    for each in buffer:
        mc.setAttr(each + ".globalScale", scaled_value)
def update_spin_box_scale(self, value):
    double_value = value / 100.0
    self.ui.Scale_spinbox.setValue(double_value)
def update_slider_scale(self, value):
    int_value = int(value * 100)
    self.ui.Scale_slider.setValue(int_value)

```

```

#-----Section-----

def section_val(self):
    slider_value = self.ui.Section_slider.value()
    selection = mc.ls(sl = True, fl = True, dag = True, type= 'stroke')
    buffer = mc.listConnections(selection, d = True, scn=True, type= 'brush')
    for each in buffer:
        mc.setAttr(each + ".tubeSections", slider_value)
def update_spin_box_section(self, value):
    self.ui.Section_spinbox.setValue(value)
def update_slider_section(self, value):
    self.ui.Section_slider.setValue(value)

#-----Density-----

def density_val(self):
    slider_value = self.ui.Density_slider.value()
    density_value = slider_value / 10.0 # Convert to actual density
    selection = mc.ls(sl=True, fl=True, dag=True, type='stroke')
    for each in selection:
        mc.setAttr(each + ".sampleDensity", density_value) # Applying lower
density

def update_spin_box_density(self, value):
    double_value = value / 10.0
    self.ui.Density_spinbox.setValue(double_value)
    self.density_val(double_value)
def update_slider_density(self, value):
    int_value = int(value * 10)
    self.ui.Density_slider.setValue(int_value)
    self.density_val(value)

#-----Smooth-----

def smooth_val(self):

```

```

slider_value = self.ui.Smooth_slider.value()
selection = mc.ls(sl = True, fl = True, dag = True, type= 'stroke')
for each in selection:
    mc.setAttr(each + ".smoothing", slider_value)
def update_spin_box_smooth(self, value):
    # Update the spinbox value based on the slider's position
    self.ui.Smooth_spinbox.setValue(value)
    self.smooth_val(value)

def update_slider_smooth(self, value):
    # Update the slider's position based on the spinbox's value
    self.ui.Smooth_slider.setValue(value)
    self.smooth_val(value)
#-----Twist-----
def update_spin_box_twist(self, value):
    # Update the spinbox value based on the slider's position
    self.ui.Twist_spinbox.setValue(value)
    self.twist_val(value)

def update_slider_twist(self, value):
    # Update the slider's position based on the spinbox's value
    self.ui.Twist_slider.setValue(value)
    self.twist_val(value)
def twist_val(self, twist_rate):
    # Apply the twist rate to selected objects in Maya
    selection = mc.ls(sl=True, fl=True, dag=True, type='stroke')
    buffer = mc.listConnections(selection, d=True, scn=True, type='brush')
    for each in buffer:
        mc.setAttr(each + ".twistRate", twist_rate)

```