МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система для моделювання роботи маніпуляторів»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Власенко О.В.**

Студента групи ІН-63 **Чірва Д.М.**

СУМИ 2020

РЕФЕРАТ

Записка: 50 стор., 35 рис., 2 табл., 16 джерел.

Об’єкт дослідження — інформаційної системи та інтерфейсу оператора для конструкторського бюро виробничого підприємства.

Мета роботи — розробка інформаційної системи та інтерфейсу оператора для конструкторського бюро виробничого підприємства, якими дозволяє конструювати, та маніпулювати промисловими роботами.

Методи дослідження — проведено аналіз вимог технічного завдання, розглянуті різні види інтерфейсів для виконання розробки особливостей інтерфейсу для промислових роботів.

Результати — розроблено функціональну блок-схему алгоритму управління промисловим роботом та алгоритм роботи інтерфейсу. До складу інтерфейсу входять графічна та програмна частини.

ІНТЕРФЕЙС, ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЮВАННЯ, ТЕХНОЛОГІЧНИЙ ПРОЦЕС, АЛГОРИТМ, ВІРТУАЛЬНЕ СЕРЕДОВИЩЕ, ПРОМИСЛОВИЙ РОБОТ, МАНІПУЛЯТОР, ADOBE AFTER EFFECTS, AUTODESK 3DS MAX, UNITY.

ЗМІСТ

ВСТУП

Сучасне вітчизняне конструкторське бюро повинно розвиватися в напрямку автоматизації виробництва з широким використанням ЕОМ та роботів, впровадження гнучких технологій, що дозволяють швидко і ефективно перебудовувати технологічні процеси на виготовлення нових виробів. Автоматизація проектування технології та управління виробничими процесами – один з основних шляхів інтенсифікації виробництва, підвищення його ефективності та якості продукції.

Автоматизація технологічної підготовки виробництва на підприємстві є важливим кроком до скорочення витрат на випуск нових видів виробів. Для сучасної системи автоматизованого технологічного проектування висувають високі вимоги до універсальності, комплексності, інтегрованості з існуючими на підприємстві базами даних і системами, за умов відносно простоти в адаптації та експлуатації, поширення методики автоматизованого проектування на різні види виробництв та підтримки технології «клієнт-сервер».

На даний час на підприємствах також гостро постало питання про необхідність швидкої оцінки трудовитрат і матеріальних ресурсів, необхідних для виготовлення продукції. Підприємствам треба швидко визначати, чи зможуть вони виконати нове замовлення, і який прибуток буде отримано. Наслідком цього є необхідність реальної інтеграції системи проектування технологічних процесів з системою автоматизованого проектування, конструювання і автоматизовані системи управління підприємства.

Слід мати на увазі, що робота технолога в розробці технологічного процесу поки ніяк не може бути замінена комп'ютерними програмами, тому зазначені системи вирішують завдання моделювання прийнятих технологічних рішень. Технолог у цьому випадку отримує можливість побачити наслідки і результати технологічного процесу, не вдаючись до

натурного його запуску, що передбачає виготовлення дорогого оснащення.

Система САПР ТП повинна бути створена як засіб, що істотно прискорює і спрощує проектування технології, розрахунок режимів і норм, розрахунок технологічних розмірних ланцюгів, формування текстів переходів, вибір необхідної оснастки та інструментів, формування документації та операційних ескізів.

Актуальними напрямками діяльності конструкторського бюро є:

- модернізація і розширення наявного асортименту продукції;

розробка апаратів і науково-технічних винаходів нових видів технічно досконалих конструкцій електричних машин;

- авторський супровід, доведення та випробування дослідних зразків нових виробів на стендах і в експлуатації,

конструктивний супровід серійного виробництва;

- проведення повних періодичних і типових випробувань серійної та нової продукції;

- розробка проектної документації за технічними вимогами замовника.

Таким чином, метою роботи є розробка інтерфейсу, для конструювання маніпулятора і симуляції виконання роботи маніпулятора в віртуальній середовище.

Для досягнення даної мети необхідно вирішити наступні завдання:

- проаналізувати предметну область: розглянути аналогічних інтерфейсів;

- проаналізувати технологічний процес робота маніпулятора, розробити функціональну блок-схему алгоритму;

- проаналізувати в яких програмах розробити компоненти інтерфейсу;

- проаналізувати оптимальні умови роботи в лабораторії, застосувати заходи проти підвищення значення напруги в електричній ланці.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ І ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз вимог технічного завдання

Згідно ТЗ необхідно розробити інтерфейс оператора для конструкторського бюро виробничого підприємства. Інтерфейс повинен мати такі параметри:

- функції інтерфейсу: конструювання промислових роботів та маніпулювання ними;
- вимоги до ОС: Windows 7 x86/x64, RAM 512 мБ;
- роздільна здатність екрану: від 512×384 до 1366×768 пікселів;
- параметри промислових роботів – робоча зона, вантажопідйомність, кут повороту, число ступенів рухливості.

Основні функції і задачі конструкторського бюро є:

- розробка документації на модернізацію діючого обладнання;
- розробка технічних завдань на засоби механізації та автоматизації ручної праці;
- розробка технічних завдань;
- здійснення авторського нагляду за виготовленням розробленого обладнання;
- конструкторська підготовка виробництва;
- розробка проектів нових дослідних і промислових установок, нестандартного устаткування та пристроїв у зв'язку з реконструкцією об'єктів, автоматизацією виробництва та механізацією трудомістких процесів;
- участь у роботі комісії з приймання в експлуатацію знову виготовленого нестандартного обладнання та засобів механізації та автоматизації;
- участь у проведенні атестації і раціоналізації робочих місць;
- підготовка висновків та розробка документації на раціоналізаторські

пропозиції та винаходи.

Промислові роботи призначені для заміни людини при виконанні основних і допоміжних технологічних операцій у процесі промислового виробництва. При цьому вирішується важлива соціальна завдання – звільнення людини від робіт, пов'язаних з небезпеками для здоров'я або з важкою фізичною працею, а також від простих монотонних операцій, що не потребують високої кваліфікації. Гнучкі автоматизовані виробництва, що створюються на базі промислових роботів, що дозволяють вирішувати задачі автоматизації на підприємствах з широкою номенклатурою продукції при дрібносерійному та штучному виробництві. Копіюють маніпулятори, керовані людиною-оператором, необхідні при виконанні різних робіт з радіоактивними матеріалами. Крім того, ці пристрої незамінні при виконанні робіт в космосі, під водою, в хімічно активних середовищах. Таким чином, промислові роботи і копіюють маніпулятори є важливими складовими частинами сучасного промислового виробництва.

1.2 Аналіз основних особливостей інтерфейсу оператора

Інтерфейс – система правил і засобів, регламентує та забезпечує взаємодію програми з користувачем.

Зокрема інтерфейс являє собою сукупність технічних, програмних і методичних (протоколів, правил, угод) засобів сполучення в обчислювальній системі користувачів із пристроями і програмами, а також пристроїв з іншими пристроями і програмами.

Інтерфейс в широкому сенсі слова – це спосіб (стандарт) взаємодії між об'єктами. Інтерфейс в технічному сенсі слова задає параметри, процедури і характеристики взаємодії об'єктів. Розрізняють:

- інтерфейс користувача – набір методів взаємодії комп'ютерної програми і користувача цієї програми;
- програмний інтерфейс – набір методів для взаємодії між програмами;
- фізичний інтерфейс – спосіб взаємодії фізичних пристроїв.

Найчастіше мова йде про комп'ютерні порти.

Основу такої взаємодії користувача з комп'ютером складають діалоги. Під діалогом в даному випадку розуміють регламентований обмін інформацією між людиною і комп'ютером, який здійснюється в реальному масштабі часу і спрямований на спільне вирішення конкретної задачі. Кожен діалог складається з окремих процесів вводу / виводу, які фізично забезпечують зв'язок користувача і комп'ютера. Обмін інформацією здійснюється за рахунок передачею повідомлення.

В основному користувач генерує повідомлення наступних типів:

- запит інформації;

- запит допомоги;

- запит операції або функції;

- введення або зміна інформації.

У відповідь користувач отримує підказки чи довідки, інформаційні повідомлення, що потребують відповіді; накази, які потребують дії; повідомлення про помилки та іншу інформацію.

Інтерфейс користувача комп'ютерної програми включає:

- засоби відображення інформації, відображувану інформацію, формати і коди;

- командні режими, мови (користувач-інтерфейс);

- пристрої і технології введення даних;

- діалоги, взаємодію та транзакції між користувачем і комп'ютером, зворотний зв'язок із користувачем;

- підтримку прийняття рішень у конкретній предметній області;

- порядок використання програми та документацію на неї.

Користувальницький інтерфейс часто розуміють тільки як зовнішній вигляд програми. Насправді користувач сприймає через нього всю програму в цілому, а значить, таке розуміння є занадто вузьким. Програмна інженерія об'єднує в собі всі елементи і компоненти програми, які здатні впливати на

взаємодію користувача з програмним забезпеченням .

До цих елементів відносяться:

- набір задач, які користувач розв'язує за допомогою системи;

- використовувана системою метафора (наприклад, робочий стіл MS Windows ®);

- елементи управління системою;

- навігація між блоками системи;

- візуальний (і не тільки) дизайн екранів програми;

1.3 Види інтерфейсів

Інтерфейс – це, перш за все, набір правил. Як будь-які правила, їх можна узагальнити, зібрати в «кодекс», згрупувати за спільною ознакою.

Сучасними видами інтерфейсів є:

- командний інтерфейс. Командний інтерфейс називається так тому, що в цьому виді інтерфейсу людина подає «команди» комп'ютера, а комп'ютер їх виконує і видає результат людині. Командний інтерфейс реалізований у вигляді пакетної технології і технології командного рядка;

- WIMP-інтерфейс (Window – вікно, Image – образ, Menu – меню, Pointer – покажчик). Характерною особливістю цього виду інтерфейсу є те, що діалог з користувачем ведеться не за допомогою команд, а за допомогою графічних образів – меню, вікон, інших елементів. Хоча і в цьому інтерфейсі подаються команди машині, але це робиться «опосередковано», через графічні образи. Цей вид інтерфейсу реалізований на двох рівнях технологій: простий графічний інтерфейс і «чистий» WIMP-інтерфейс;

- SILK-інтерфейс (Speech – мова, Image – образ, Language – мова, Knowledge – знання). Цей вид інтерфейсу найбільш наближений до звичайної, людської форми спілкування. В рамках цього інтерфейсу йде звичайний «розмова» людини і комп'ютера. При цьому комп'ютер знаходить для себе команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд він також перетворює у зрозумілу людині

форму. Цей вид інтерфейсу найбільш вимогливий до апаратних ресурсів комп'ютера, і тому його застосовують в основному для військових цілей.

Інтерфейси користувача бувають двох типів:

- процедурно-орієнтовані (примітивні меню з вільною навігацією);

- об'єктно-орієнтовані (прямого маніпулювання).

Процедурно-орієнтований інтерфейс використовує традиційну модель взаємодії з користувачем, засновану на поняттях «процедура» і «операція». В рамках цієї моделі програмне забезпечення надає користувачеві можливість виконання деяких дій, для яких користувач визначає відповідність даних і наслідком виконання яких є отримання бажаного результату.

Об'єктно-орієнтовані інтерфейси використовують модель взаємодії з користувачем, орієнтовану на маніпулювання об'єктами предметної області.

В рамках цієї моделі користувачеві надається можливість безпосередньо взаємодіяти з кожним об'єктом та ініціювати виконання операцій, у процесі яких взаємодіють кілька об'єктів. Завдання користувача формулюється як цілеспрямована зміна деякого об'єкта. Об'єкт розуміється в широкому сенсі слова – модель база даних, системи і т. д.

Об'єктно-орієнтований інтерфейс припускає, що взаємодія з користувачем здійснюється за допомогою вибору і переміщення піктограм відповідної об'єктно-орієнтованої області. Розрізняють однодокументні (SDI) та багатодокументні (MDI) інтерфейси.

Особливості процедурно-орієнтованих інтерфейсів:

- забезпечують користувачеві функції, необхідні для виконання завдань;

- акцент робиться на завдання;

- піктограми представляють програми, вікна або операції;

- зміст папок і довідників відображається з допомогою таблиці-списку.

Особливості об'єктно-орієнтованих інтерфейсів:

- забезпечують користувачеві можливість взаємодії з об'єктами;

- акцент робиться на вхідні дані і результати;

- піктограми представляють об'єкти;

- папки та довідники є візуальними контейнерами об'єктів.

Примітивним називається інтерфейс, який організовує взаємодію з користувачем і використовується в консольному режимі. Єдине відхилення від послідовного процесу, який забезпечується даними, полягає в організації циклу обробки декількох наборів даних.

Інтерфейс «Меню», на відміну від примітивного інтерфейсу, дозволяє користувачеві вибирати операцію зі спеціального списку, що виводиться йому програмою. Ці інтерфейси припускають реалізацію безлічі сценаріїв роботи, послідовність дій в яких визначається користувачами. Деревовидна організація меню передбачає строго обмежену реалізацію.

При цьому можливі два варіанти організації меню:

- кожне вікно меню займає весь екран;

- на екрані одночасно присутні кілька різнорівневих меню (Windows).

В умовах обмеженої навігації, незалежно від варіанту реалізації, пошук пункту більш ніж двохрівневого меню виявляється досить складним завданням.

Інтерфейс з вільною навігацією (графічний інтерфейс) підтримує концепцію інтерактивної взаємодії з візуальним зворотним зв'язоком з користувачем і можливість прямого маніпулювання об'єктом (кнопки, індикатори, рядком стану). На відміну від інтерфейсу «Меню», інтерфейс з вільною навігацією забезпечує можливість здійснення будь-яких допустимих в конкретному стані операцій, доступ до яких можливий через різні інтерфейсні компоненти («гарячі» клавіші і т. д.).

1.4 Аналіз аналогічних інтерфейсів

Програма RoboticArm

Завдяки сучасному розвитку САПР можна навести безліч програмних забезпечень для використання у конструкторському бюро.

RoboticArm – багатофункціональна і проста програма для керування маніпулятором.

Вона виконує роботизовані рухи. Містить найпростіші способи управління:

- клавіатуру;

- мишку;

- джостік.

Програма RoboticArm проста в управлінні, потрібно просто навести на частину рукоятки (на екран), становище якого потрібно змінити і натиснути відповідну кнопку. Так можливо запрограмувати серію рухів.

Компоненти інтерфейсу поділяються на:

- вікно із записом і очищенням алгоритму команд управління;

- Filearea (файлове поле), де є кнопка «створити», «відкрити», «зберегти» і «вихід»;

- інструменти з кнопками:

1) «старт» – алгоритм команд виконується;

2) «зупинка» – алгоритм команд припиняє свою роботу;

3) «перемотування назад» – повертає алгоритм команд у вихідне положення;

4) «видалити» – алгоритм видаляється з пам'яті програми.

Інтерфейс програми RoboticArm зображена на рис. 1.1.

Рисунок 1.1 – Інтерфейс програми RoboticArm

До переваг системи можна віднести наступні:

- простота у використанні;

- можливість встановлювати різні операційні системи;

- використання в різних режимах «програмному» і «основному».

SprutCAM – програмне забезпечення для розробки керуючих програм для устаткування з ЧПУ.

SprutCAM російська CAM-система, що підтримує розробку УП для багатокоординатних, електроерозійного, токарно-фрезерного обладнання і промислових роботів з урахуванням повної кінематичної 3D-моделі вузлів.

SprutCAM дозволяє створювати 3D-схеми верстатів і всіх його вузлів і проводити попередню віртуальну обробку з контролем кінематики і 100 % вірогідністю, що дозволяє наочно програмувати складне багатокоординатне обладнання.

Інтерфейс програми SprutCAM зображений на рис. 1.2.

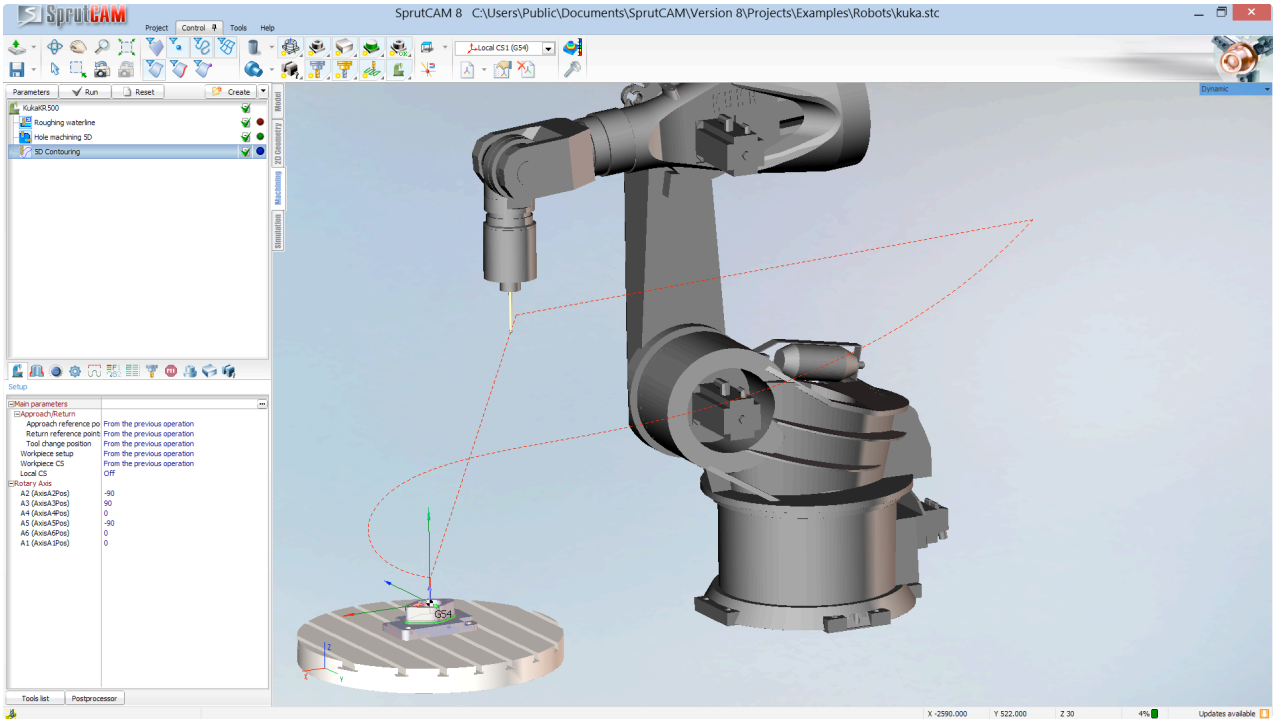

Рисунок 1.2 – Інтерфейс програми SprutCAM

Важливими відмінними рисами системи є:

- розвинуті засоби імпорту та перетворення геометричної моделі;

- коректна обробка розривів і напуску між формотворчими поверхнями;

- наскрізна передача стану заготовки між етапами і різними видами обробки;

- розширений набір функцій керування параметрами технологічних операцій;

- безліч методів оптимізації обробки;

- обов'язковий контроль на підрізування на всіх стадіях розрахунку траєкторії;

- реалістичне моделювання обробки;

- простота в освоєнні і використанні;

- зручний інтерфейс, практично виключає потреба у використанні документації.

2 ФУНКЦІОНАЛЬНА БЛОК-СХЕМА АЛГОРИТМУ УПРАВЛІННЯ ПРОМИСЛОВИМ РОБОТОМ І ІНТЕРФЕЙСУ

2.1 Технологічні процеси у виробництві та застосування роботів маніпуляторів

Використання розроблюваного інтерфейсу передбачає у конструкторському бюро, тому розглянемо особливості автоматизованих пристроїв для роботи на заводах представлені в широкому асортименті. Роботи-маніпулятори успішно використовуються в металургії, машинобудуванні, легкої і харчової промисловості. Вони здатні замінити людину у важких і небезпечних умовах праці. Забезпечуючи високі швидкість, точність, якість, а також високу окупність виробництва.

Роботи-маніпулятори бувають:

- універсальні роботи-маніпулятори – ці пристрої являють собою автоматизований механізм, обладнаний спеціальним розпізнавальним інструментом – так званої «рукою» маніпулятора. Вона слугує основним діючим органом в різних цілях. Якщо це робот для зварювання, рука-маніпулятор виконує зварювальні операції, якщо робот-укладальник, рука слугує для укладання та пакування продукції;

- роботи-зварювальники. Зварювальні маніпулятори являють собою комплекс передових технологій та комплектуючих деталей, запрограмованих на виконання дугової та точкової зварювання об'єктів. Маніпулятори слугують для зварювання ємностей, кранів, балок і цистерн;

- роботи-збирачі. Складальні роботи-маніпулятори в основному являють собою 6-ти осьові пристрої з 6-ма ступенями свободи, які приводяться в дію за рахунок системи сервоприводів;

- роботи-різьбярі;

- роботи-малярі. Дані машини оснащені спеціальними пульверизаторами для фарбування деталей і володіють підвищеною гнучкістю для захисту шлангів при подачі в робочу зону барвника від механічних впливів, скручування і зламу, забруднення та запилення, що

просто неможливо для виконання людьми вручну;

- згинальні роботи. Пристрої здатні здійснювати завантаження об'єкта в згинальну голівку, подачу, поворот об'єкта і вивантаження після згинання;

- роботи-вантажники. Виконують приймання і відправлення вантажу з високу швидкістю і ефективністю;

- роботи-пакувальники. Машини оснащені гнучкою рукою-маніпулятором, яка дозволяє їм зі спритністю і обережністю упаковувати навіть крихкі предмети, не розбиваючи їх на відміну від людей-пакувальників;

- роботи-сортувальники. Пристрої оснащені робочим інструментом і рядом датчиків визначення продукції для точного її сортування.

2.2 Алгоритм роботи управління промисловим роботом

На рисунках 2.1 та 2.2 представлені алгоритми роботи розроблюваного інтерфейсу оператора для конструкторського бюро виробничого підприємства

Алгоритми розроблені в середовищі Microsoft Office Visio 2007. На рисунку 2.1 представлена блок-схема алгоритму управління промисловим роботом, а на рисунки 2.2 представлена блок-схема алгоритму роботи інтерфейсу.

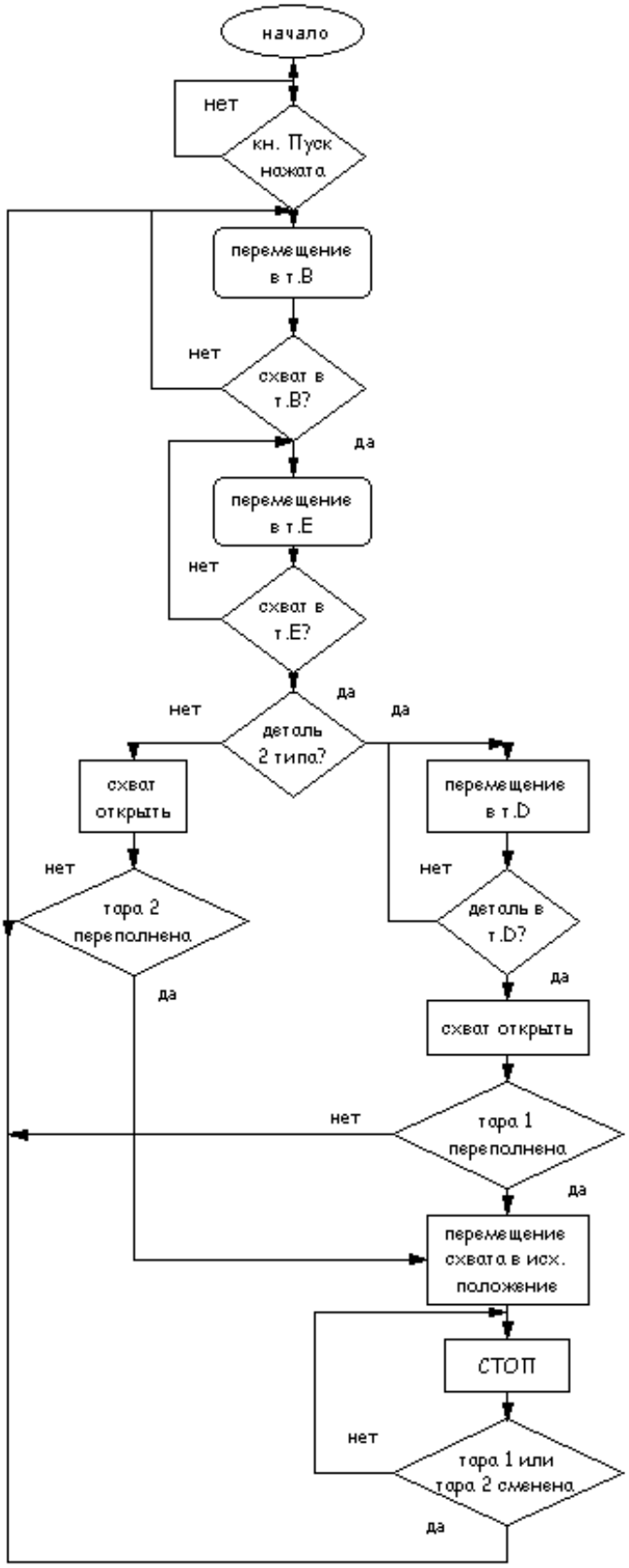

Рисунок 2.1 – Блок-схема алгоритму управління промисловим роботом

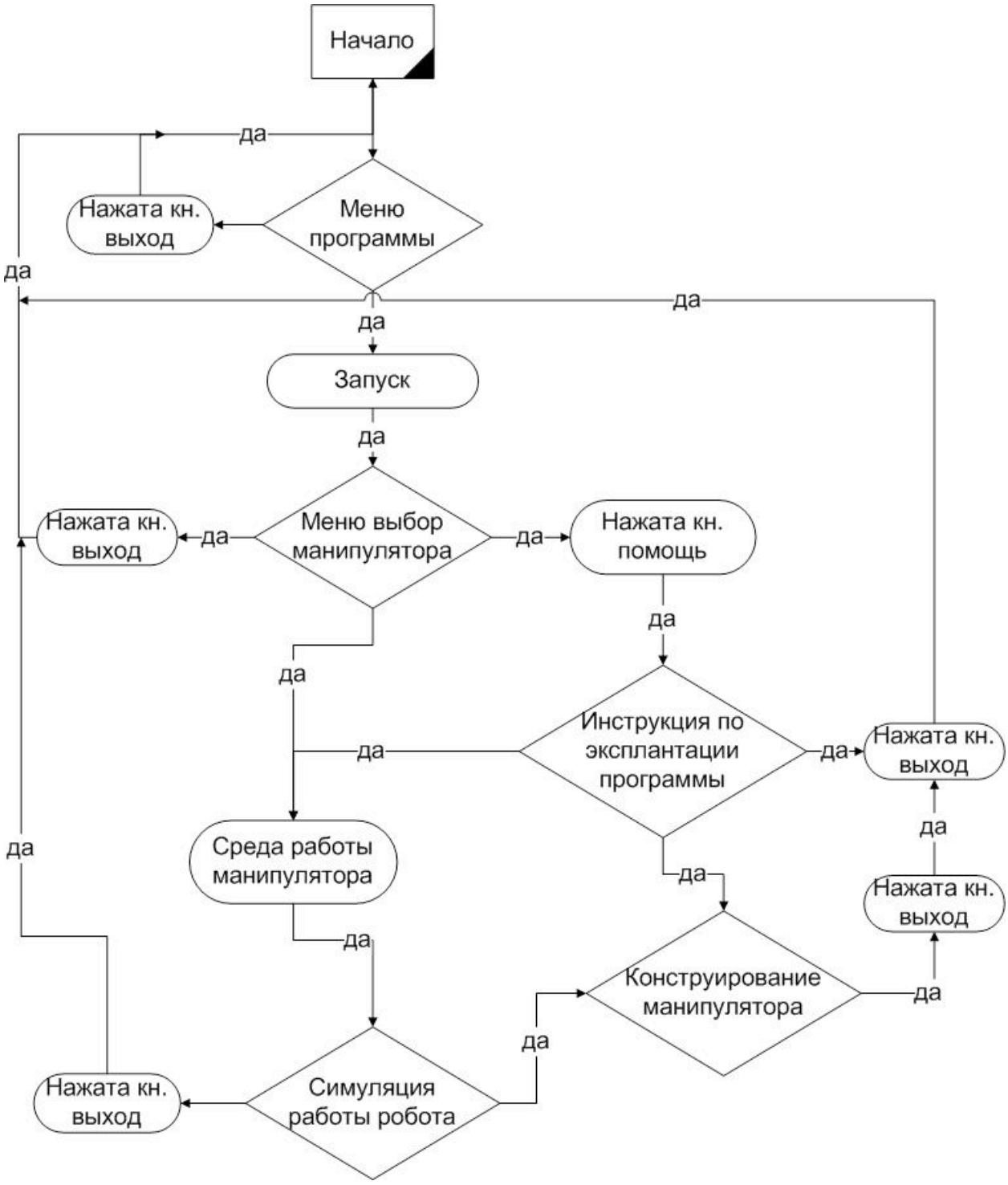

Рисунок 2.2 – Частина алгоритму роботи інтерфейсу

3 РОЗРОБКА КОМПОНЕНТІВ ІНТЕРФЕЙСУ

3.1 Програмна система Adobe After Effects

Adobe After Effects — програмне забезпечення компанії Adobe Systems для редагування відео та динамічних зображень, розробки композицій (композітінг), анімації та створення різних ефектів. Широко застосовується в обробці знятого відеоматеріалу (корекція кольору, пост-продакшн), під час створення рекламних роликів, музичних кліпів, у виробництві анімації (для телебачення і web), титрів для художніх і телевізійних фільмів, а також для цілої низки інших завдань, у яких потрібне використання цифрових відео-ефектів.

Під час створення інтерфейсу використовувалася моушен-графіка (Motion Graphics Design), яка вмістила понад сто шарів (рис. 3.1).

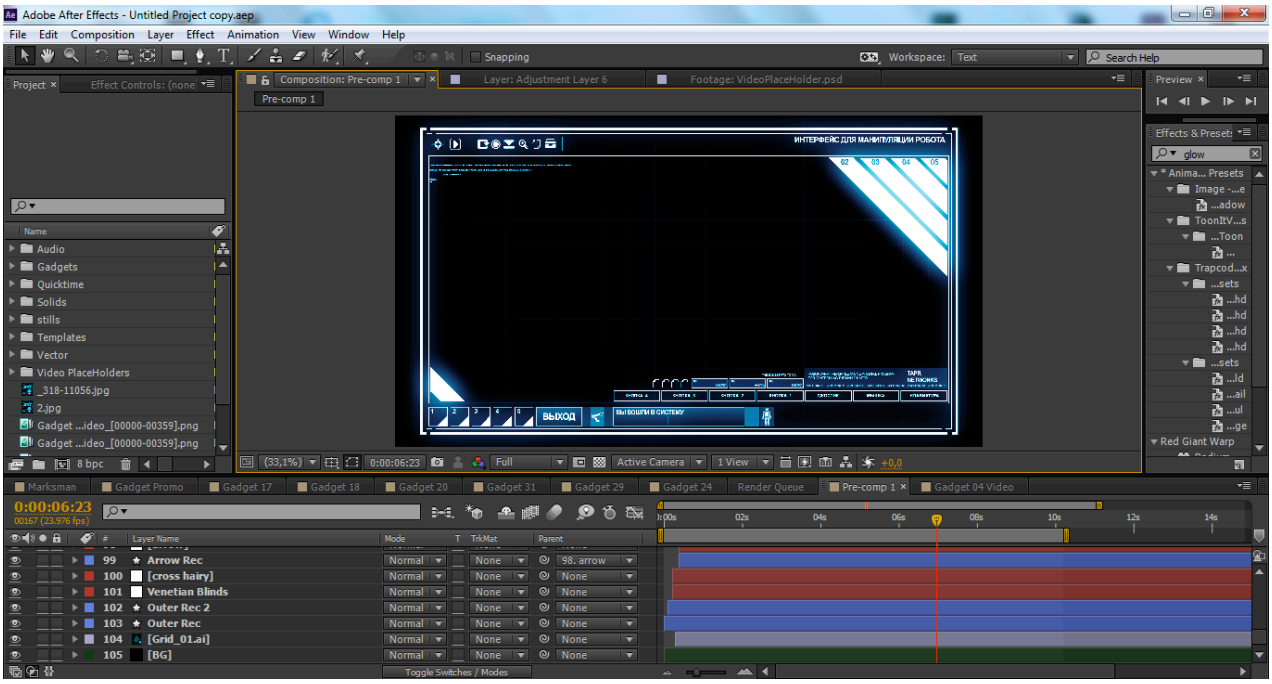

Рисунок 3.1 – Програмна середовище Adobe After Effects з використання моушен-графіки

Моушн-графіка — візуальне оформлення для відео, телебачення і кіно,

створене переважно за допомогою комп'ютерних технологій (рисунок 3.2).

Рисунок 3.2 – Приклад моушн-графіки

Так само використовувалися шари «shape layer», «adjustment layer», корекція кольору і текст.

Shape layer (фігурні шари) — інструмент для створення векторної графіки та анімації.

Adjustment layer (Коригувальний шар) — редагування шарів.

Корекція кольору – це регулювання колірних складових R (червоний), G (зелений), B (синій) з метою зміни загальної кольоровості і візуального стилю зображення.

Всі керуючі кнопки були створені й анімовані у програмі After Effects.

Програма поділяється на дві частини, основна частина, де розміщені всі функціональні кнопки, меню, вибір маніпулятора, конструювання маніпулятора та використання.

Меню представлено на рисунку 3.3 і основний інтерфейс представлений на рисунку 3.4.

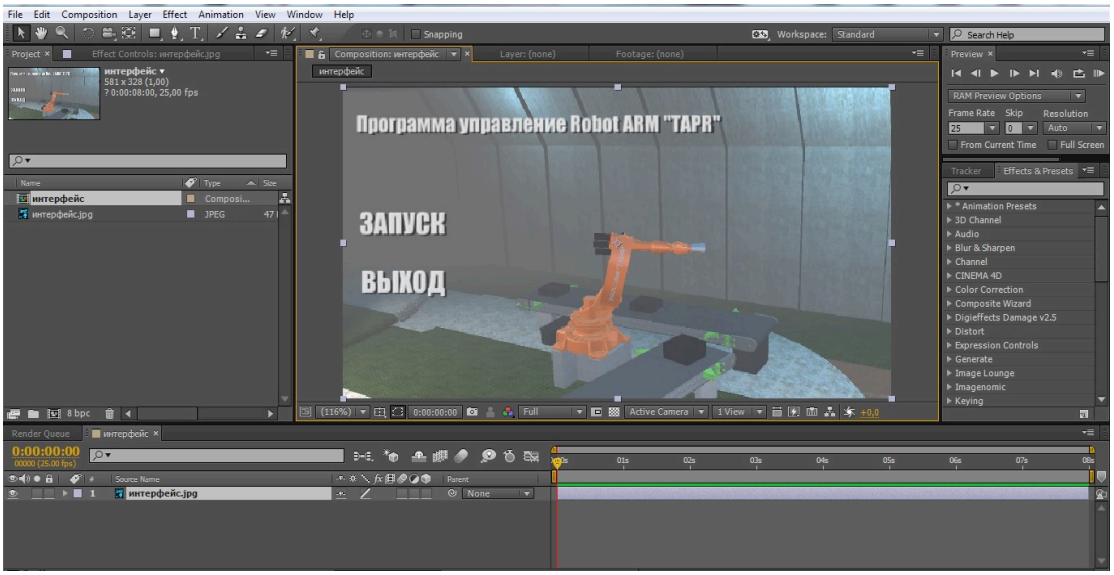

Рисунок 3.3 – Меню програми

Дизайн меню дуже простий і естетичний, в нього входить лише дві функціональні кнопки:

- запуск;
- вихід.

Рисунок 3.4 – Основна частина інтерфейсу

Основна частина інтерфейсу поділяються на дві категорії:

- графічну, з інформаційно керуючими розділами;

- тривимірну, де знаходиться сам маніпулятор, зі сценою.

У графічній частині розміщені такі кнопки:

- вихід;

- джойстик;

- мишка;

- клавіатура;

- зберегти.

Так само є не задіяні кнопки в даний момент, але можливе додавання по мірі вдосконалення інтерфейсу.

3.2 Програмна система Autodesk 3ds Max

Autodesk 3ds Max – повнофункціональна професійна програмна система для створення і редагування тривимірної графіки і анімації, розроблена компанією Autodesk. Містить найсучасніші засоби для художників і фахівців в області мультимедіа.

Під час створення маніпулятора, в задачі конвеєра і навколишнього середовища використовувалися високо-полігональні об'єкти, а також анімація цих об'єктів (рис. 3.5).

Високо-полігональне моделювання (high-poly моделювання) – це розробка абсолютно точної копії об'єкта, з використанням під час побудови моделі великої кількості полігонів.

Дві головних властивості всіх високо полігональних моделей:

- максимально точна передача форми та геометрії модельованого об'єкта;

- фінальний рендер (візуалізація кінцевого результату) здійснюється з великим навантаженням.

У даній сцені використовувалося більш ніж шістсот елементів (деталей) для відтворення максимально реалістичною виконання роботи

маніпулятора в конструкторському бюро (рис. 3.6).

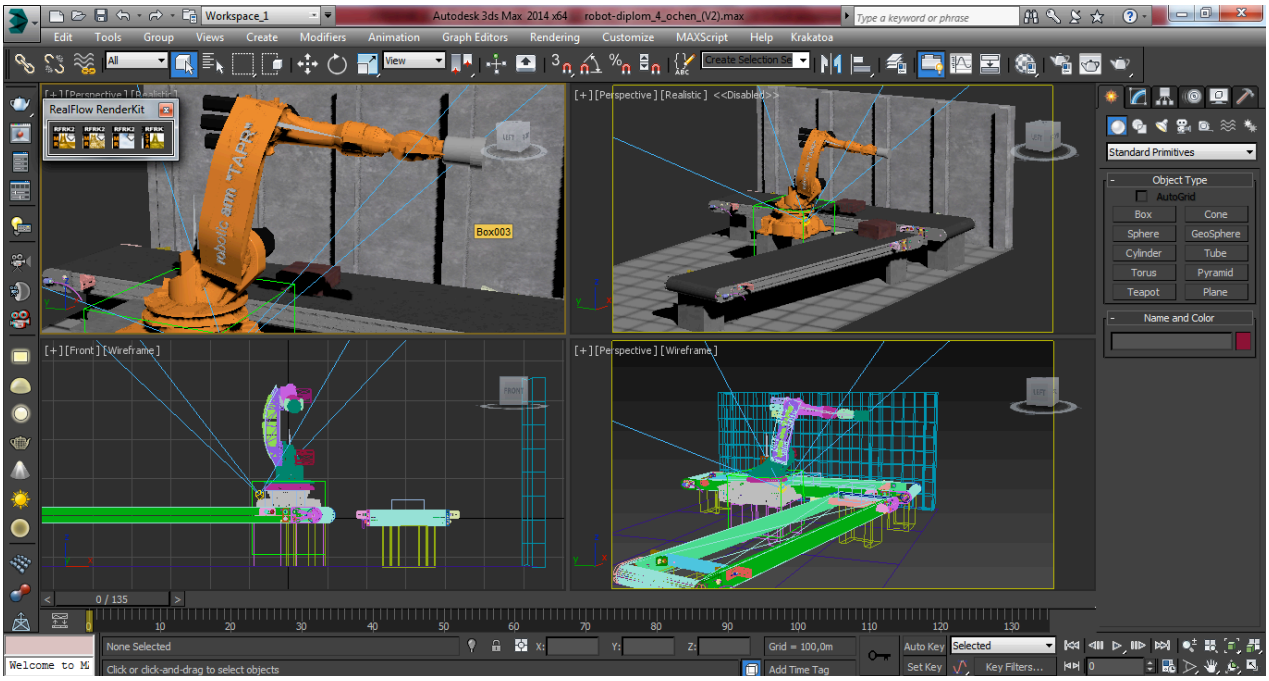

Рисунок 3.5 – Тривимірна модель в середовищі Autodesk 3ds Max

Сцена складається з таких елементів:

- маніпулятор;

- конвеєрна стрічка;

- стіна і підлога.

Маніпулятор в свою чергу складається:

- основи на якій розміщена вся несуча частина, переміщується ліворуч та праворуч, вздовж осі Z;

- плечового суглоба, що переміщує вперед і назад вздовж осі X ;

- ліктьового суглоба, що переміщується вертикально вгору-вниз, так само вздовж осі X;

- електромагніту.

Модель маніпулятора була прив'язана до кісток (Bone), після чого був проведений Rig (створення скелета) і записані анімації виконання роботи маніпулятора.

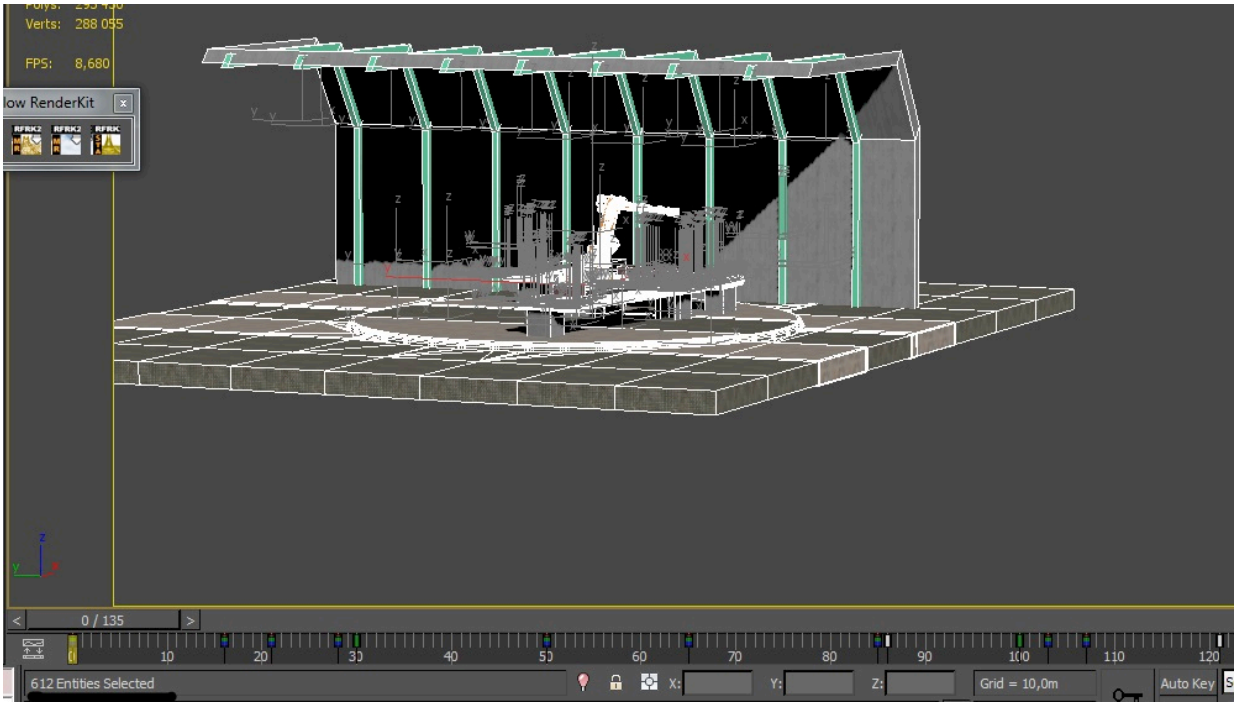

Рисунок 3.6 – Елемент в середовищі Autodesk 3ds Max

Кістки (Bones) є з'єднаними, ієрархічно зв'язаними системою окремих кісток, які можуть використовуватися для анімації інших об'єктів або ієрархій (рис. 3.7).

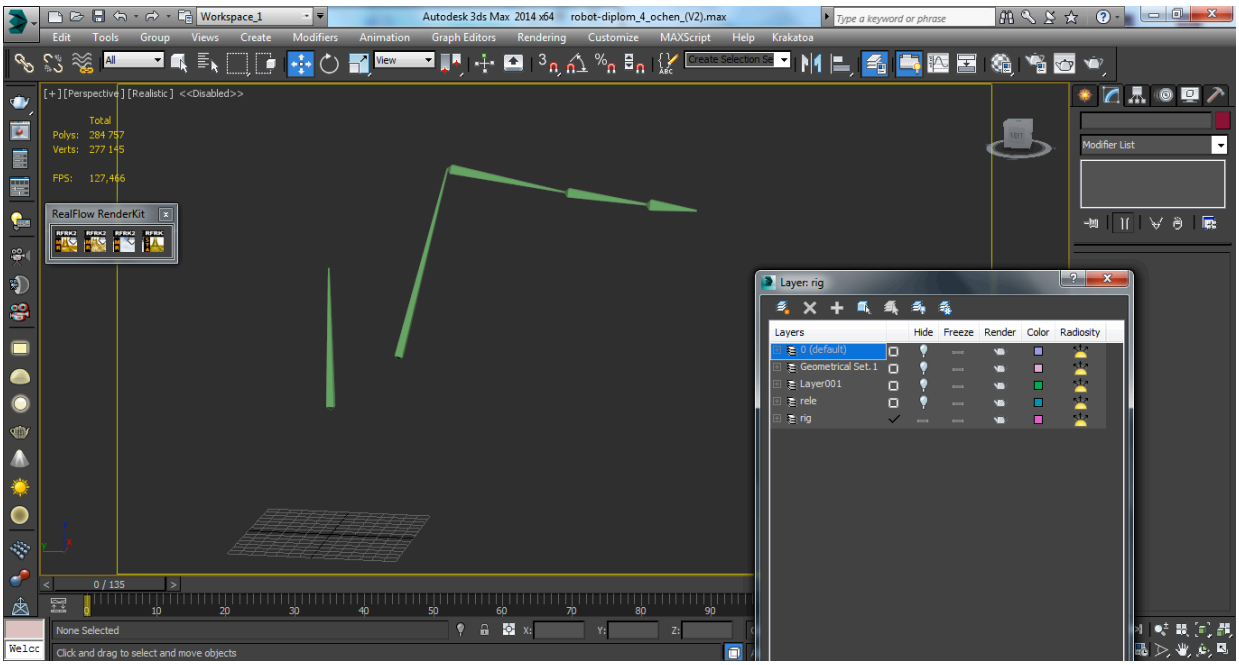

Рисунок 3.7 – Кістки маніпулятора в середовищі Autodesk 3ds Max

Риг (англ. rig — оснащення, пристосування) — термін у комп'ютерній анімації, який описує набір залежностей між керуючими і керованими елементами, створений таким чином, щоб керуючих елементів було менше, ніж керованих.

Комп'ютерна анімація — вид мультиплікації, створюваний за допомогою комп'ютера. На відміну від більш загального поняття «графіка CGI», що відноситься як до нерухомих, так і рухомих зображень, комп'ютерна анімація передбачає тільки рухомі.

Конвеєрна стрічка також поділяется на декілька елементів (рис. 3.8):

- привід (електродвигун з редуктором);
- завантажувальний і розвантажувальний пристрій;
- очисні пристрої;
- натяжна станція.

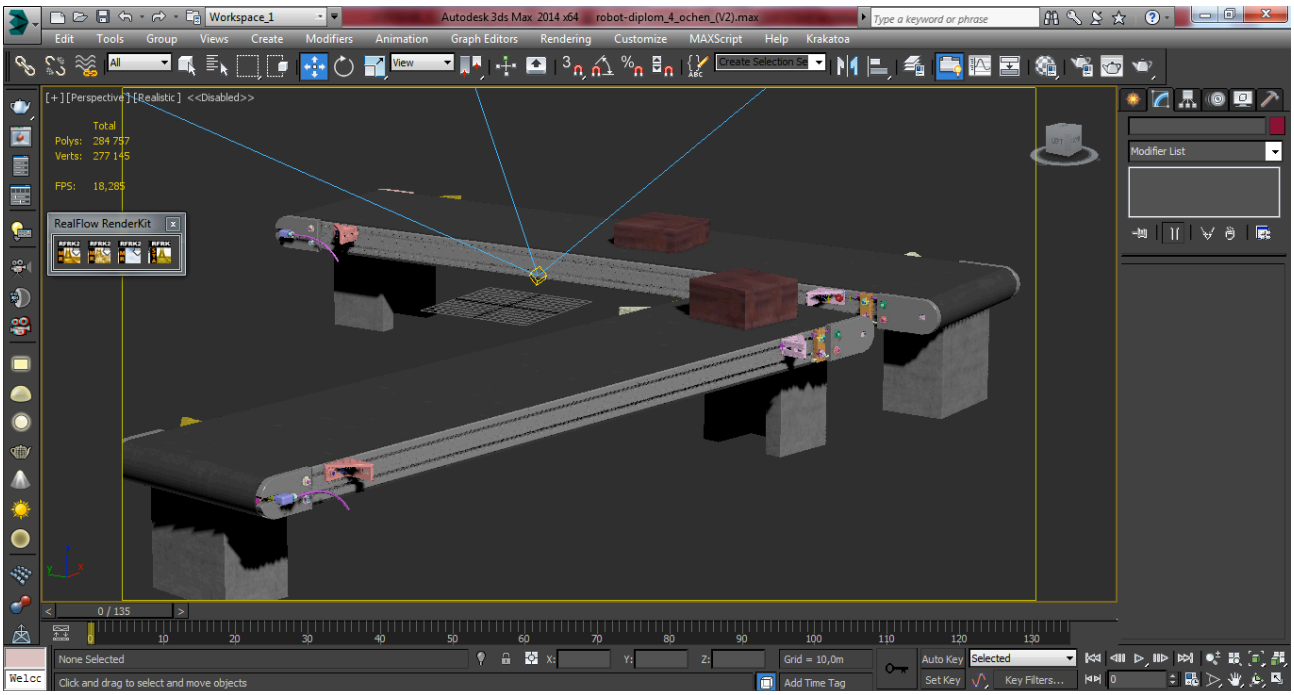

Рисунок 3.8 – Конвеєрна стрічка в середовищі Autodesk 3ds Max

Натяжна станція є компенсаційним подовженням для конвеєрної стрічки, оскільки стрічка в процесі тривалих експлуатацій здатна

витягнутися (розтягнутися).

Привід у свою чергу складається з таких частин: електродвигун, редуктор, гальмівний механізм, муфти з'єднання і приводні барабани.

Завантажувальний/розвантажувальний пристрій – лоток, на який надходить вантаж з іншого конвеєра. Служить розподільником вантажу за всією шириною конвеєрної стрічки.

Очисні пристрої служать очищувальним елементом від налипаючих часток вантажу на робочій стороні стрічки.

Стіна і підлога були відтворені найпростішими об'єктами, Plan і Box елементами.

Всі текстури були створені в багатофункціональному графічному редакторі Adobe Photoshop (рис. 3.9).

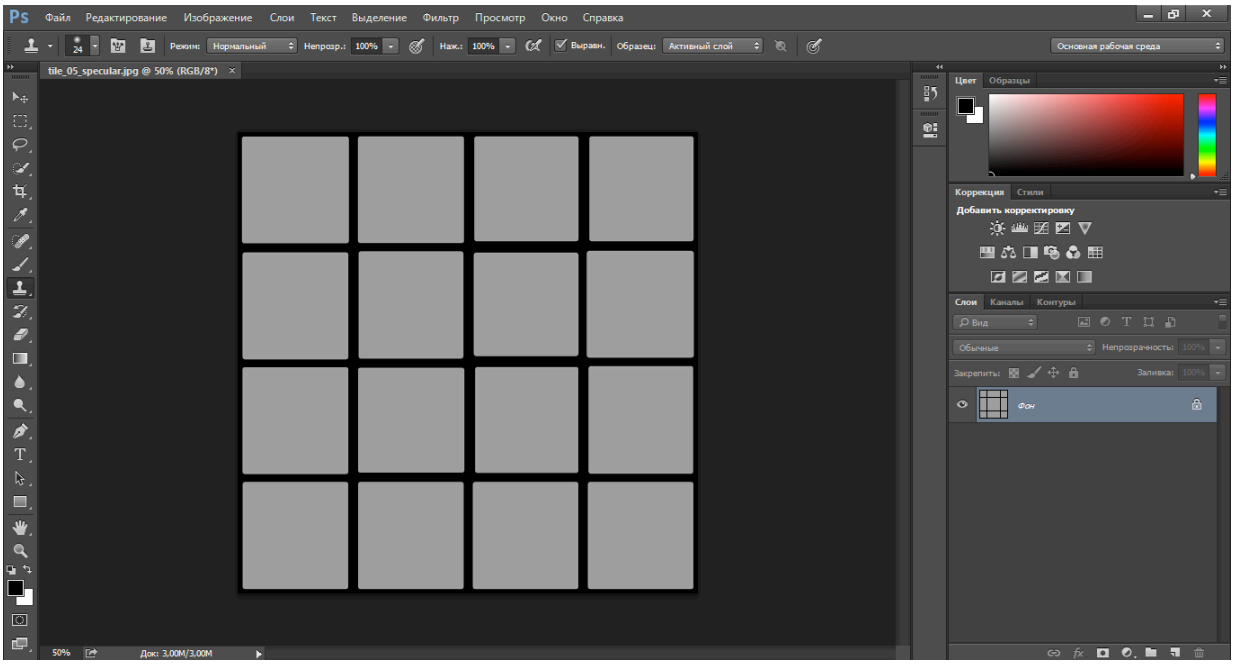

Рисунок 3.9 – Текстура в середовищі Adobe Photoshop

Для візуалізації зображення використовувалася система візуалізації V-Ray — це рейтрейсний рендерер, в якому є кілька алгоритмів прорахунку глобального освітлення (Global Illumination):

- Light Cache;

- Photon Map (фотонна карта);

- Irradiance Map;

- Brute Force (QMC).

Є можливість вибору різних алгоритмів для прорахунку відображення глобального освітлення.

Результати візуалізації зображення, представлені на рисунку 3.10.

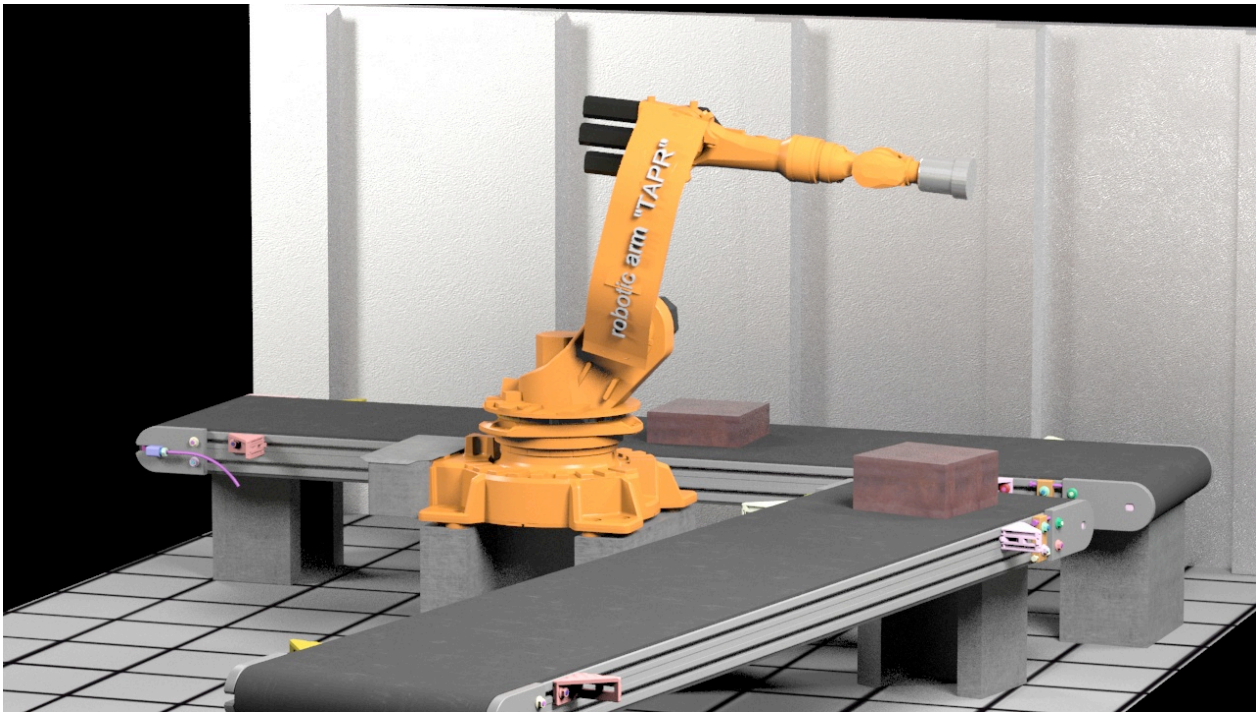

Рисунок 3.10 – Результат візуалізації зображення в програмі 3ds Max

Для створення маніпулятора і його середовища роботи використовувалося 284757 полігонів (polygins) і 277145 вершин (vertexs) що показано на рисунку 3.11.

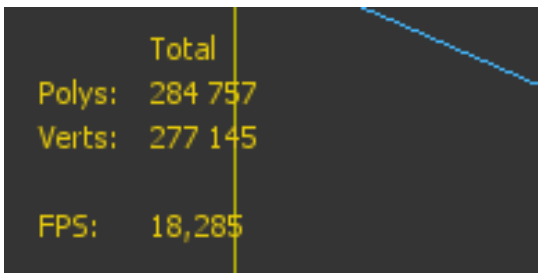

Рисунок 3.11 – Загальна кількість полігонів і вершин у програмі 3ds Max

3.3 Інструмент для розробки двохвимірних і тривимірних додатків Unity

Unity – це інструмент для розробки двох і тривимірних додатків і ігор, працює під операційними системами Windows і OS X.

Створені за допомогою Unity додатки працюють під операційними системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One (рис. 3.12).

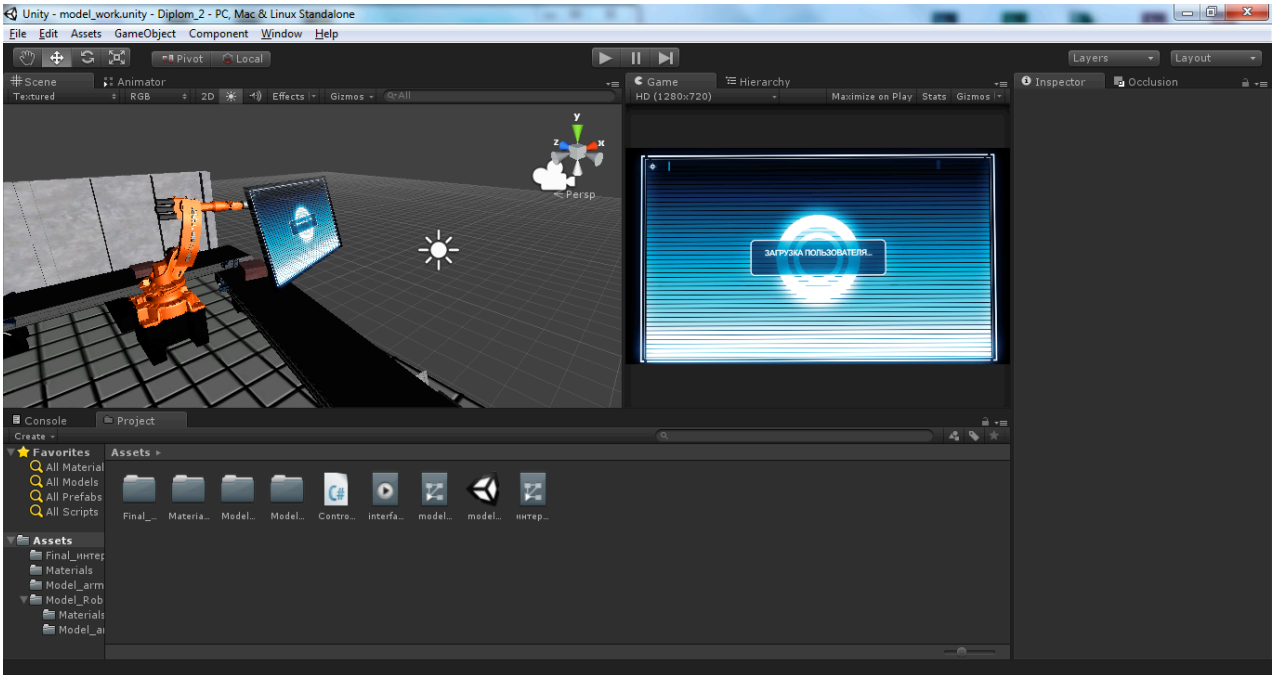

Рисунок 3.12 – Тривимірна модель з інтерфейсом в середовищі Unity

Для меню програми створено нову сцену, в яку помістили модель маніпулятора, приміщення в якому він працює, сам інтерфейс (рис. 3.13).

Створено новий C# скрипт, в якому прописаний код управління меню (рис. 3.14).

Першим ділом потрібно підключити бібліотеку using UnityEngine.UI, для активації всіх класів з ім'ям UI.

UI (user interface) – це користувальницький інтерфейс.

Далі оголошуємо public Canvas quitMenu, public Button startText, public Button exitText, щоб відображалися в інспекторі.

Canvas quitMenu – для активації закриття меню.

Button startText – запуск програми на натискання кнопки.

Button exitText – вихід програми на натискання кнопки.

У void Start пишемо quitMenu = quitMenu.GetComponent<Canvas> (), коли в інспекторі використовується quitMenu, то він відразу шукає всіх компоненти з назвою Canvas.

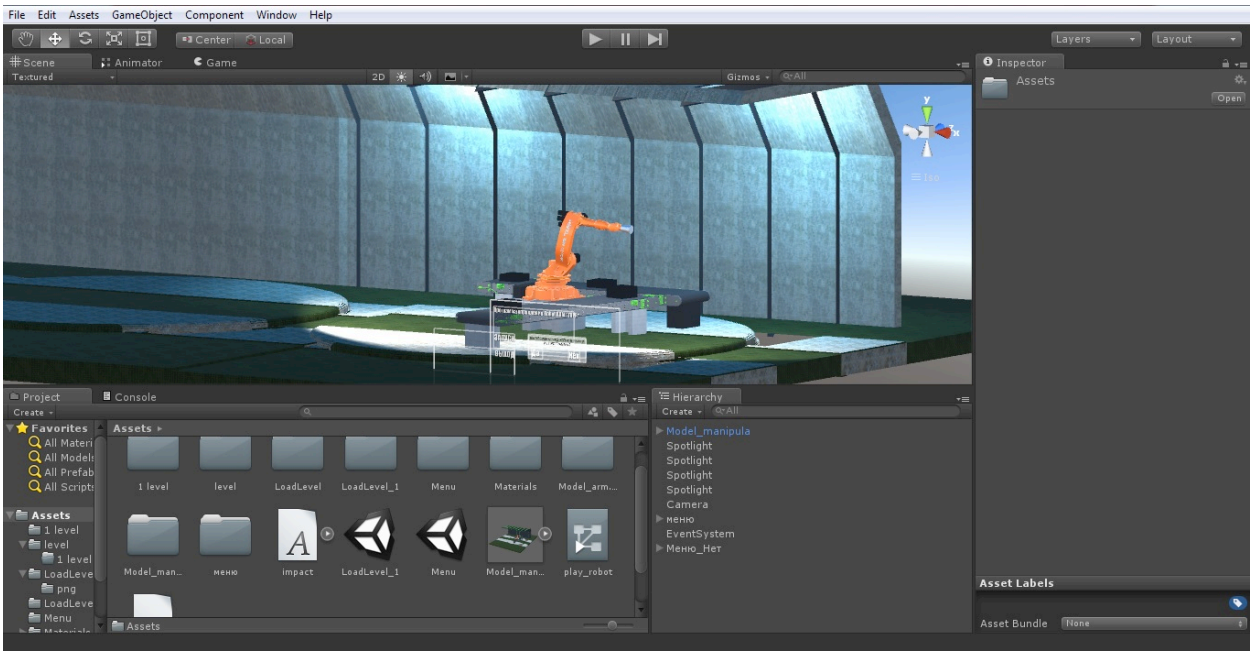

Рисунок 3.13 – Меню інтерфейсу програми в середовищі Unity

Так само прописуємо startText = startText.GetComponent<Button> (), і exitText = exitText.GetComponent<Button> () , що виконує ті ж самі функції тільки з компонентів Button.

Закриваємо фігурні дужки і після оголошуємо public void ExitPress() (натискання вихід):

```
{           quitMenu.enabled = true;
    startText.enabled = false;
    exitText.enabled = false; }
```

Що означає quitMenu.enabled = true роздільна запуск, в даному випадку вихід, startText і exitText – забороняється.

меню.cs

No selection

```
1  using UnityEngine;
2  using UnityEngine.UI;
3  using System.Collections;
4
5  public class меню : MonoBehaviour {
6
7
8      public Canvas quitMenu;
9      public Button startText;
10     public Button exitText;
11
12
13     // Use this for initialization
14     void Start () {
15
16         quitMenu = quitMenu.GetComponent<Canvas> ();
17         startText = startText.GetComponent<Button> ();
18         exitText = exitText.GetComponent<Button> ();
19         quitMenu.enabled = false;
20     }
21
22     public void ExitPress()
23     {
24         quitMenu.enabled = true;
25         startText.enabled = false;
26         exitText.enabled = false;
27     }
28
29     public void NoPress()
30     {
31         quitMenu.enabled = false;
32         startText.enabled = true;
33         exitText.enabled = true;
```

Рисунок 3.14 – Код меню

Таким же способом прописуємо public void NoPress() (якщо натиснемо немає), де startText і exitText = true дозволяємо запуску, а quitMenu забороняється.

Оголошуємо public void StartLevel() завантаження нового рівня, у фігурних дужках прописуємо Application.LoadLevel (1) завантаження нового

рівня з номер рівня в дужках.

Далі оголошуємо public void ExitGame() вихід їх програми у фігурних дужках прописуємо Application.Quit () закриття додаток.

Код управління меню закінчений, після чого переходимо в Unity, вибираємо нашу основну частину меню і інспектор, перетягуємо в C# скрипт (рис. 3.15).

У розділах quitMenu розміщуємо спливаюче віконце після натискання клавіші «ВИХІД», та «startText» розміщуємо кнопку «Запуск» і в exitText поміщаємо «Вихід».

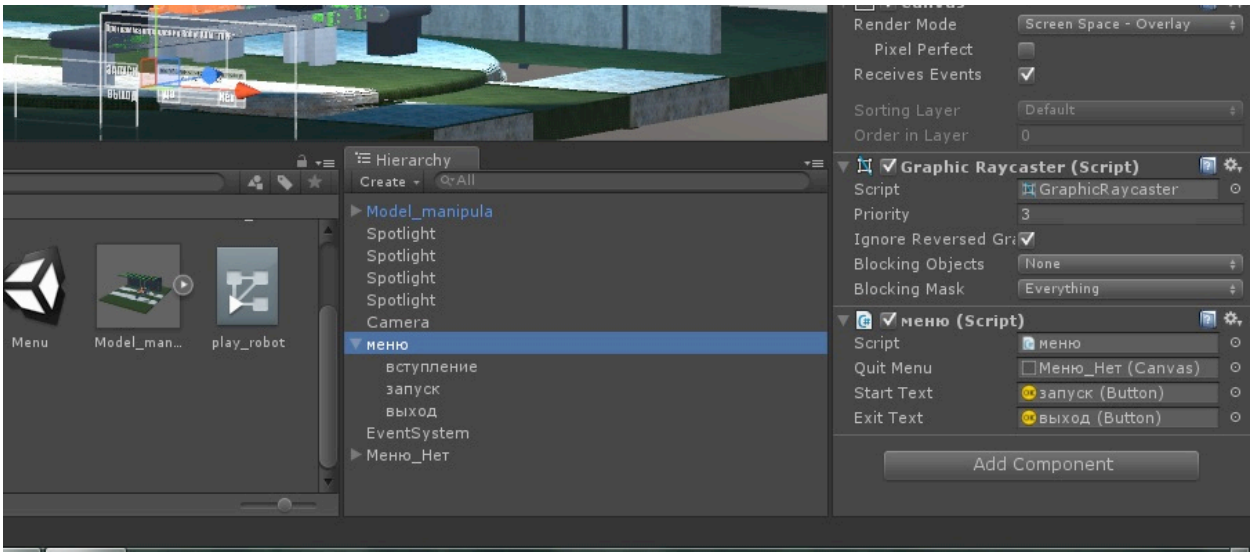

Рисунок 3.15 – Основна частина меню со скриптом

У розділах «Запуск», «Вихід», «так», «ні» переходимо до інспектора, знаходимо компонент «Button» в розділі On Click, вибираємо наш скрипт (рис. 3.16)

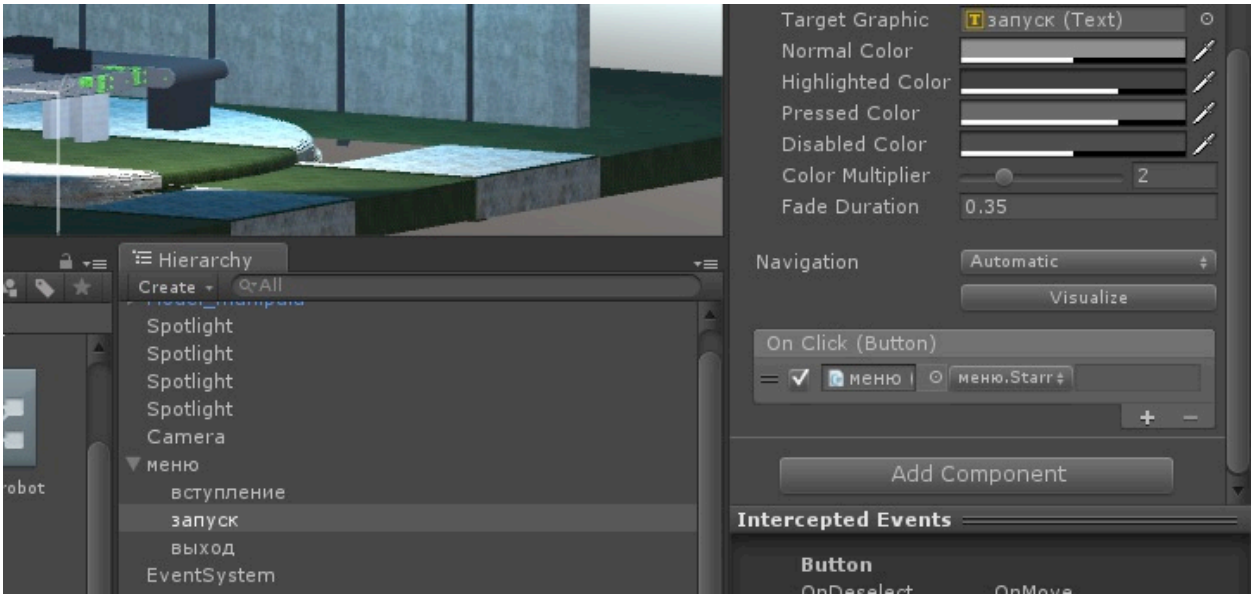

Рисунок 3.1 – Додавання сценарію в компонент «Button»

В розділі Edit Scene – Render Settings встановлюємо загальні параметри візуалізації (рис. 3.17).

Підключимо туманність для реалістичності картинки, в Fog Mode в розділах Linear (лінійний), та Linear Fog Start (початок туманності) встановимо рівень 293 і в Linear Fog End (кінець туманності) – 3325.

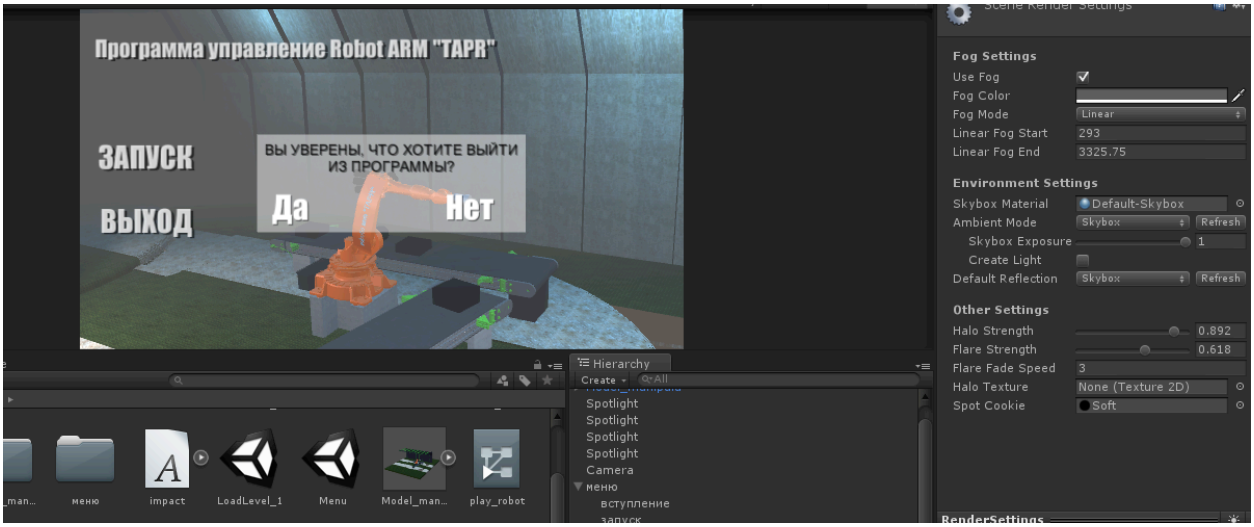

Рисунок 3.17 – Загальні параметри візуалізації інтерфейсу

Далі створюємо нову сцену, зберігаємо під назвою «Вибір маніпулятора» (рис. 3.18).

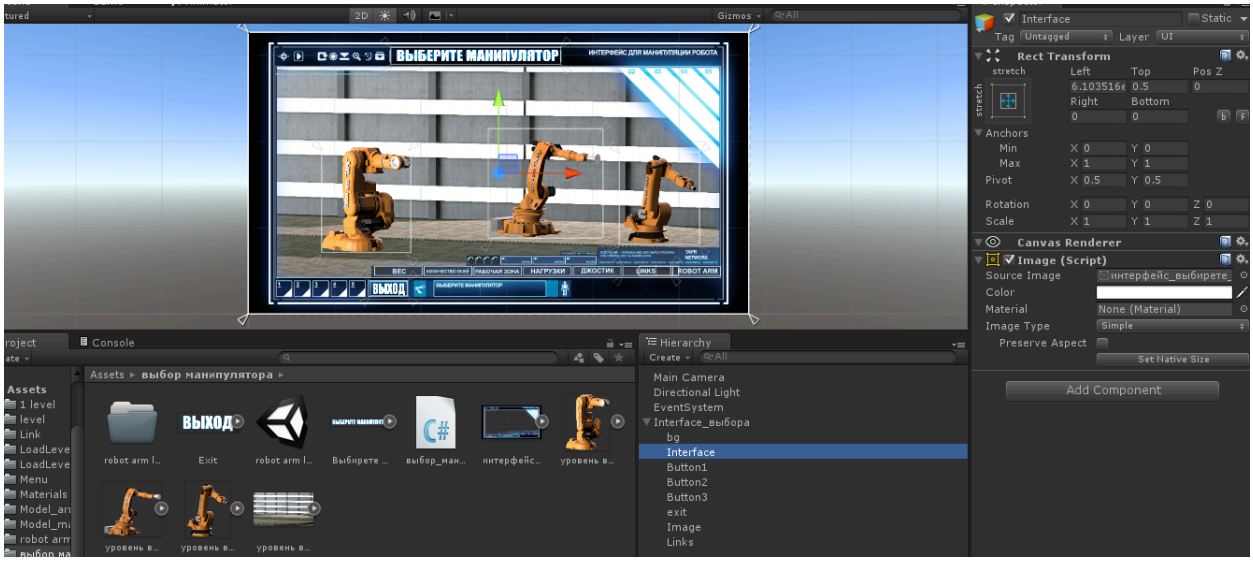

Рисунок 3.18 – Сцена під назвою «Вибір маніпулятора»

В ієрархію додаємо програмний об'єкт Canvas, до нього підключаємо сім компонентів «Image»:

- задній фон;
- інтерфейс;
- маніпулятор 1;
- маніпулятор 2;
- маніпулятор 3;
- вихід;
- links.

Завантажуємо графічний об'єкт Sprite для даної сцени, задній фон, інтерфейс, маніпулятор 1, 2, 3 і вихід (рис. 3.19).

Вибираємо інтерфейс, переходимо до інспектору, у вкладці «image», знаходимо Source Image і в нього перетягуємо «Sprite» інтерфейсу, у вкладці Rect Transform встановлюємо прив'язку по всіх краях Sprite-інтерфейсу.

Таким же чином додаємо задній фон, маніпулятор 1, маніпулятор 2, маніпулятор 3, вихід і links.

Таким же чином додаємо задній фон, маніпулятор 1, маніпулятор 2, маніпулятор 3, вихід і links.

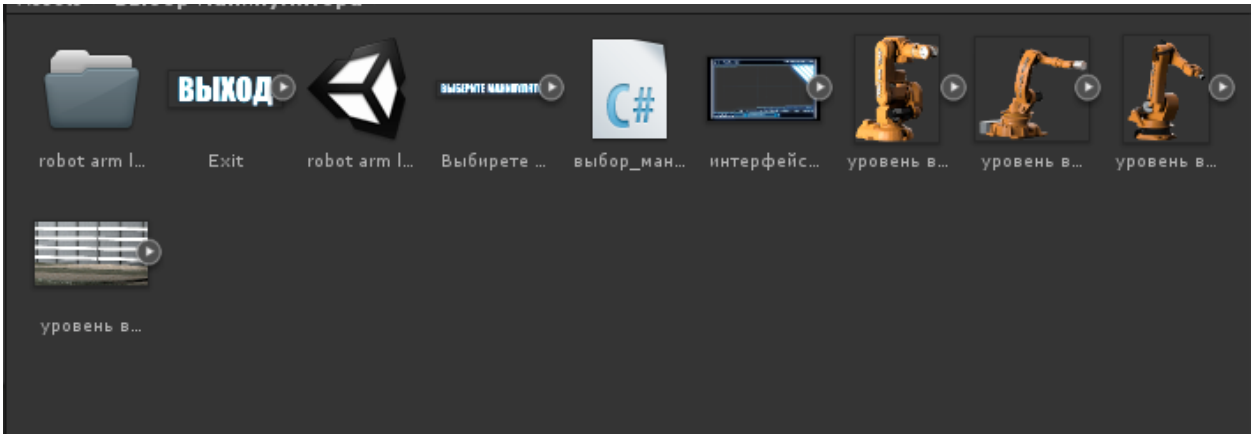

Рисунок 3.19 – Графічний об'єкт sprite для сцени під назвою «Вибір маніпулятора»

Створюємо C# скрипт з назвою «вибір маніпулятора», переходимо в програмне середовище розробки для написання коду «Mono Development».

Першим ділом потрібно підключити бібліотеку using UnityEngine.UI для активації всіх класів з ім'я ям UI.

Оголошуємо в скрипті public Button Manim1; public Button Manim2; public Button Manim3; public Button Exit_level; public Button Link.

Переходимо в void Start () та описуємо, по натискання кнопки Button з іменем Manim1, 2, 3, Exit_level, Link.

```
{
        Manim1 = Manim1.GetComponent<Button> ();
    Manim2 = Manim2.GetComponent<Button> ();
    Manim3 = Manim3.GetComponent<Button> ();
    Exit_level = Exit_level.GetComponent<Button> ();
    Link = Link.GetComponent<Button> ();
}
```

Оголошуємо, завантаження нових рівнів:

- Manim1Level завантаження 3 рівня з маніпулятором;

- Manim2Level завантаження 2 рівня з маніпулятором;
- Manim3Level завантаження 4 рівня з маніпулятором;
- за допомогою ExitLevel переходимо головне меню програми;
- LinkLevel рівень вибору деталей маніпулятора.

Код завантаження рівнів:

```
public void Manim1Level()
{
	Application.LoadLevel (3);
}

public void Manim2Level()
{
	Application.LoadLevel (2);
}


public void Manim3Level()
{
	Application.LoadLevel (4);
}


public void ExitLevel()
{
	Application.LoadLevel (0);
}

public void LinkLevel()
{
```

```
        Application.LoadLevel (5);
    }
```

За аналогією з головним меню прив'язуємо кнопки Manim1, Manim2, Manim3, Exit_level, Link з C# скриптом (рис. 3.21).

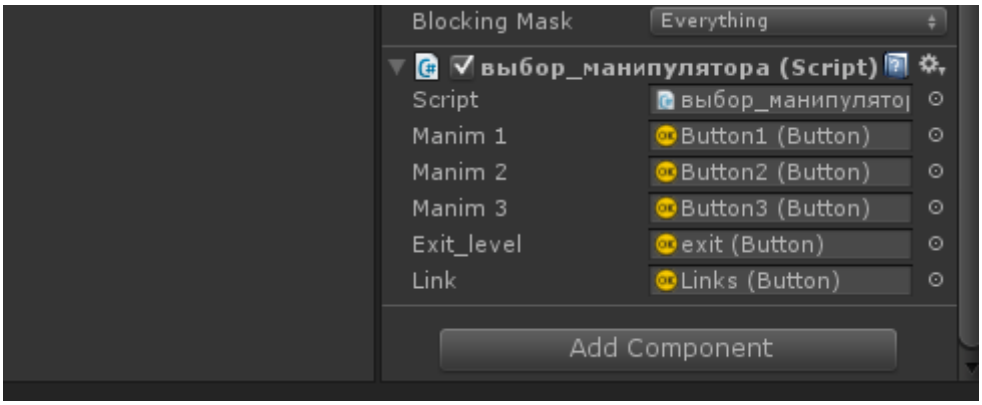

Рисунок 3.21 – Прив'язка клавіш

До вкладок «маніпулятору» 1, 2, 3, «вихід», «links», додаємо в інспекторі кнопки Button (Script), після чого до кожного з них прив'язуємо нашу C# скрипт (рис. 3.22).

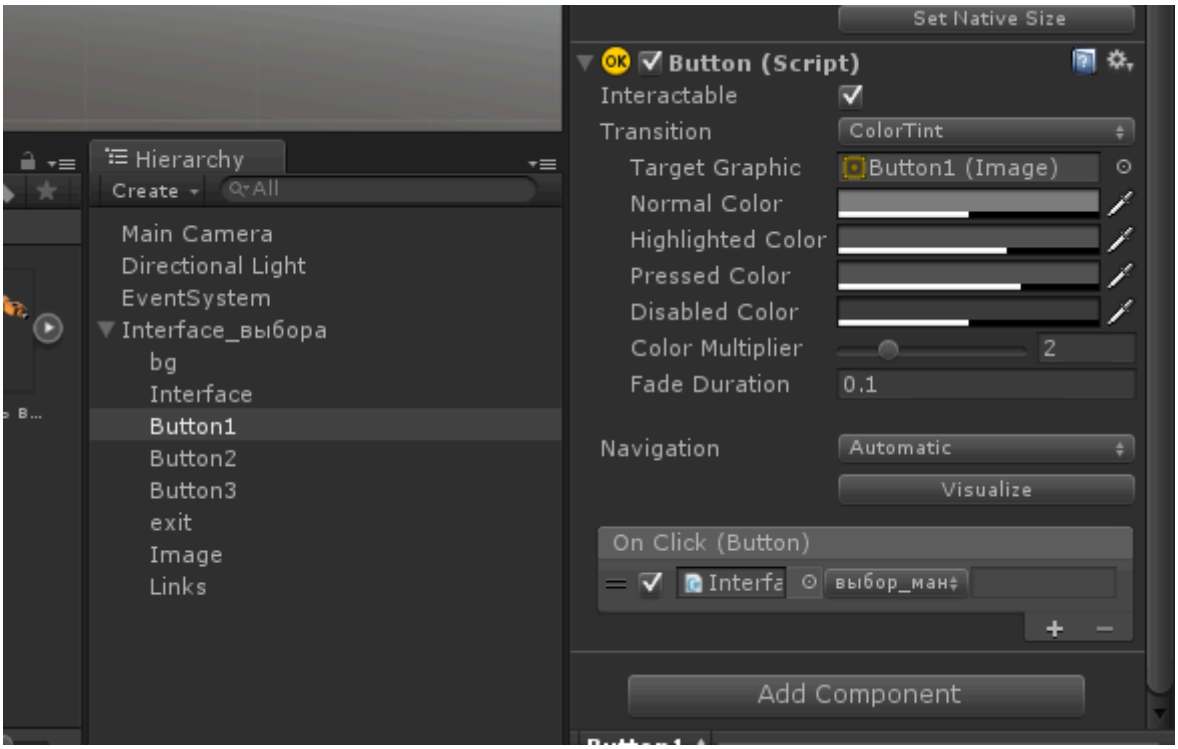

Рисунок 3.22 – Прив'язка компонента Button (Script) в C#

Таким же чином додаємо скрипт в Button «Exit» і «Links», зберігаємо цю сцену і створюємо кілька нових з назвами «Robot arm 1», «Robot arm 2», «Robot arm 3» (рис. 3.23).

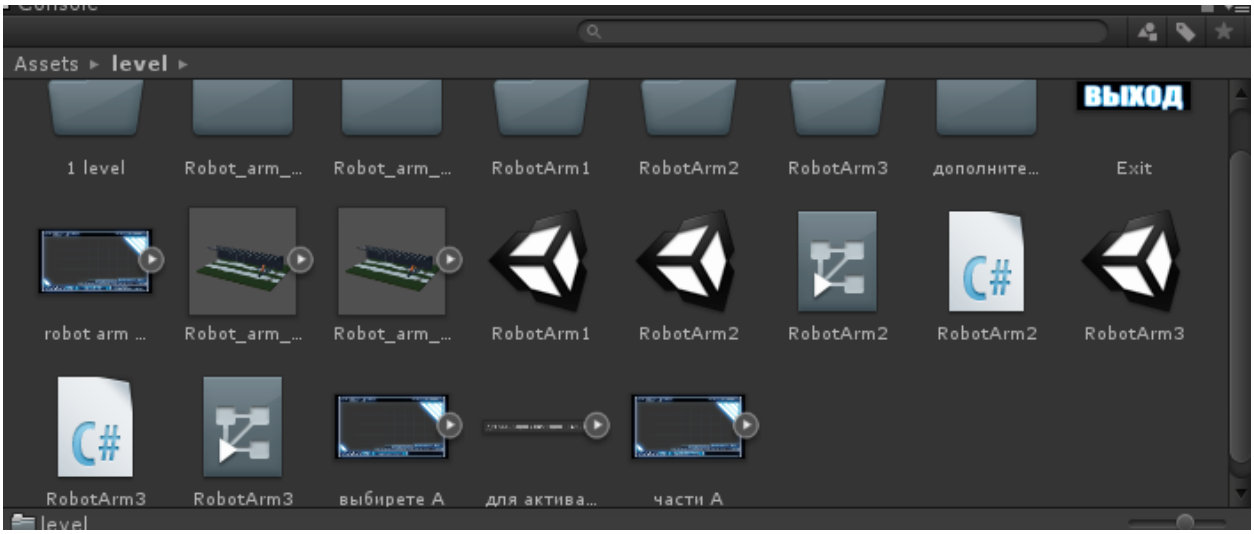

Рисунок 3.22 – Сцени під назвою «Robot arm»

У сцені «Robot arm 1» завантажуємо маніпулятор з його середовищем роботи (рис. 3.23).

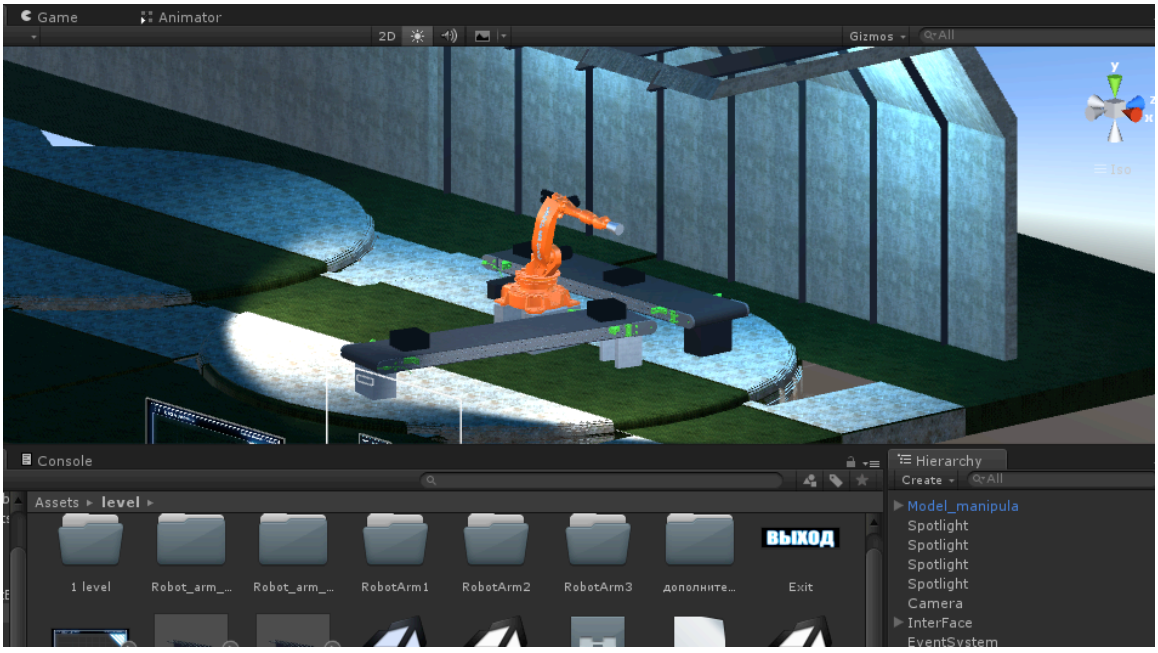

Рисунок 3.23 – Завантаження маніпулятора в сцену

Додаємо кілька джерел світла з різною інтенсивністю освітлення (рис. 3.24).

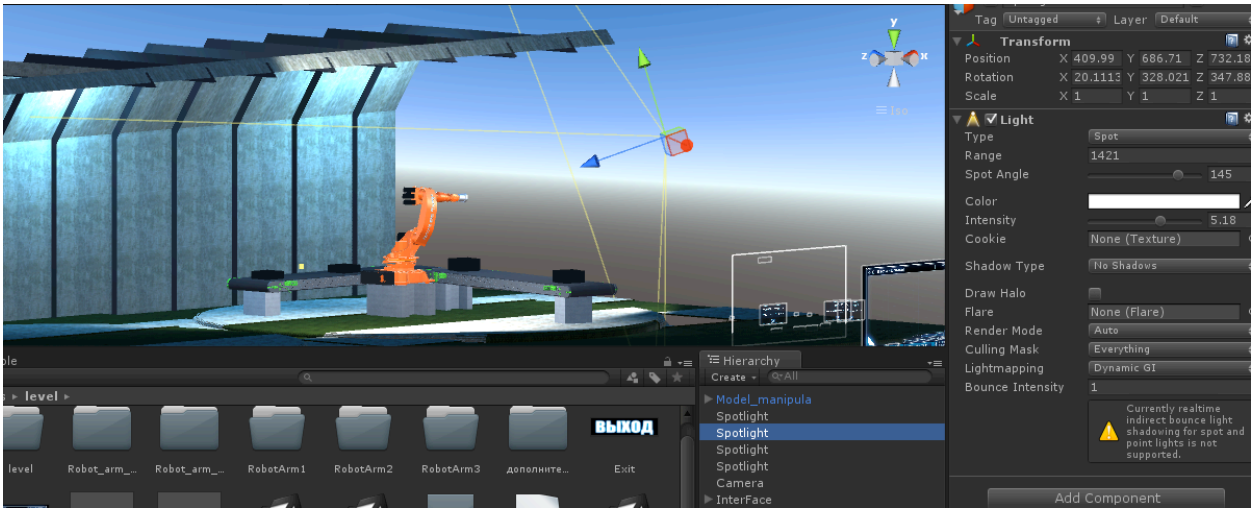

Рисунок 3.24 – Розміщення джерел світла

Створюємо кілька Canvas з назвою «InterFace», «Вантажопідйомність», «Число ступенів свободи», «Кут повороту руки», в кожен з них додаємо компонент «Image». Для «InterFace» створюємо вісім компонентів «Image» з назвами «Interface», «Links», «Exit», «RobotArm», «Навантаження», «Число ступенів повороту», «Кут повороту руки» і «Допомога».

Для шару «Вантажопідйомність» створюємо компонент Image з назвою «Грузоподъемность_1» і клавішу «ОК», таким же чином створюємо ці елементи і для «Число ступенів свободи», «Кут повороту руки» (рис. 3.25).

Створюємо C# скрипт з назвою «Функ. кнопки», переходимо в програмне середовище розробки для написання коду Mono Development.

Так само підключаємо бібліотеку using UnityEngine.UI, публікуємо такі змінні

public Canvas quitMenu; // для активації закриття спливаючого вікна 1

public Canvas quitMenu1; // для активації закриття спливаючого вікна 2

```
public Canvas quitMenu2; // для активації закриття спливаючого вікна 3
public Button RobotArm; // запуск програми після натискання кнопки
public Button exitText; // вихід програми після натискання кнопки
public Button Link; // запуск програми після натискання кнопки
public Button startText; // запуск програми після натискання кнопки
public Button HelpText;
```

У старті прописуємо натискання кнопки Button void Start ()

```
{
RobotArm = RobotArm.GetComponent<Button> (); // пошук компонента Button з назва RobotArm
startText = startText.GetComponent<Button> ();
HelpText = HelpText.GetComponent<Button> ();

Link = Link.GetComponent<Button> ();

quitMenu = quitMenu.GetComponent<Canvas> ();
quitMenu.enabled = false;

quitMenu1 = quitMenu1.GetComponent<Canvas> ();
quitMenu1.enabled = false;

quitMenu2 = quitMenu2.GetComponent<Canvas> ();
quitMenu2.enabled = false;
}
```

Canvas – елемент, призначений для створення растрового двомірного зображення за допомогою скриптования.

Після закриття фігурних дужок в Start прописуємо нижче змінниї з їх функціями:

```
public void ExitPress() // при натискання кнопки відкривається
```

спливаюче вікно

```
        {
                quitMenu.enabled = true;
                startText.enabled = false;
                exitText.enabled = false;
        }

        public void ExitPress1()
        {
                quitMenu1.enabled = true;
                startText.enabled = false;
                exitText.enabled = false;
        }
        public void ExitPress2()

        {
                quitMenu2.enabled = true;
                startText.enabled = false;
                exitText.enabled = false;
        }
```

public void NoPress2() // при натискання кнопки закриється спливаюче вікно

```
        {
                quitMenu2.enabled = false;
                startText.enabled = true;
                exitText.enabled = true;
        }
        public void NoPress1()
        {
```

```
        quitMenu1.enabled = false;
        startText.enabled = true;
        exitText.enabled = true;
    }

    public void NoPress()
    {
        quitMenu.enabled = false;
        startText.enabled = true;
        exitText.enabled = true;
    }
    public void ExitAndLevel() // завантаження 0-го рівня
    {
        Application.LoadLevel (0);
    }
    public void RobotArmLevel () // завантаження 1-го рівня
    {
        Application.LoadLevel (1);
    }
    public void LinkLevel () // завантаження 5-го рівня
    {
        Application.LoadLevel (5);
    }
    public void HelpButtun()// завантаження 6-го рівня
    {
        Application.LoadLevel (6);
    }
```

Переходимо в програмне середовище Unity і проводимо аналогічні дії для ранніх рівнів.

Для управління і перевірки роботи маніпулятора, створюємо C# скрипт з назвою «Управління маніпулятора» і додаємо в модель маніпулятора (рис. 3.26).

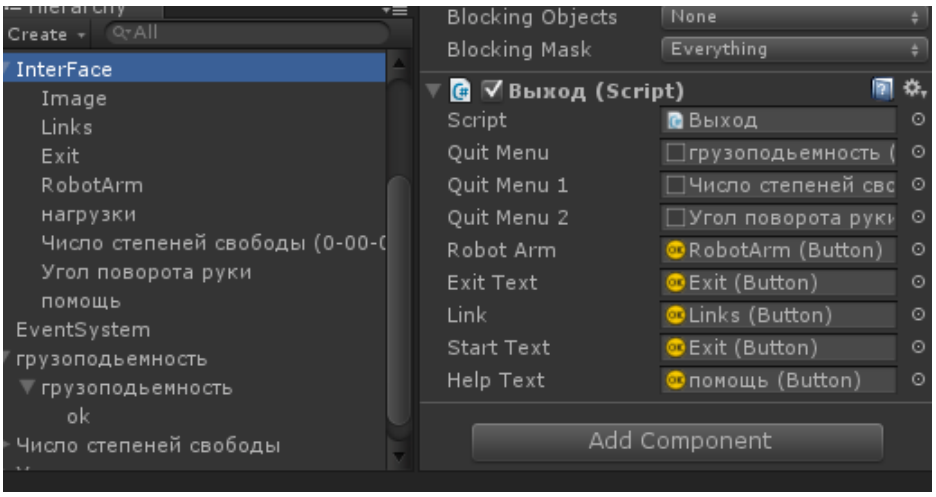

Рисунок 3.25 – Створення Canvas елементів

Створюємо управління анімації Animator Controller для програвання анімації робота, подвійним кліком переходимо в Animator.

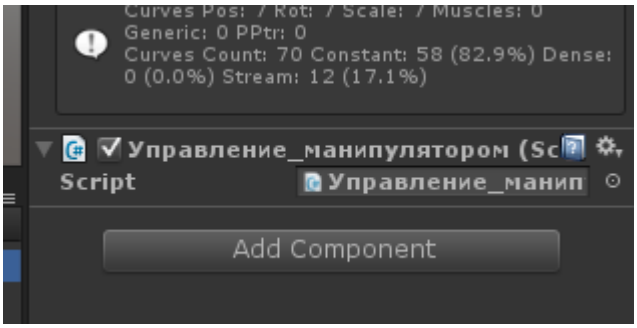

Рисунок 3.26 – Додавання C# скрипта в модель маніпулятора

Додаємо анімацію з маніпулятора (Take 001) і створюємо стандартний елемент Empty, встановлюємо його типове значення (set as default), робимо переходи між Take 001 і Empty (рис. 3.27).

У налаштуваннях Empty знімаємо всі галочки і встановлюємо значення тривалості переходу (Transition duration) 0,1.

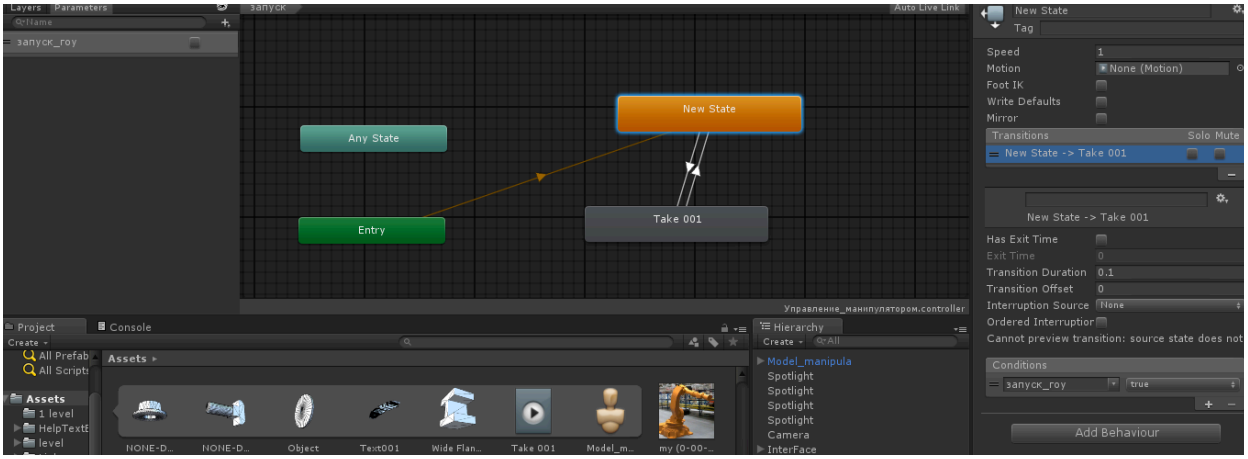

Рисунок 3.27 – Створення анімації в аніматоре

Відкриваємо C# з назвою «Управління маніпулятора», прописуємо код:

```
private Animator Play; // розпізнання аніматора
public void Reset()
{
Play.SetBool ("запуск_roy", false);
}
void Start () {
        Play = GetComponent<Animator>(); // знаходження Аніматора
}

// Update is called once per frame
void Update () {
        if (Input.GetKeyDown(KeyCode.T)) // після натискання кнопки «T» запускається анімація
        {
                Play.SetBool("запуск_roy", true);
                Invoke("Reset", 2);
        }
```

}

Для кнопок «Robot arm 2», «Robot arm 3» виконуємо такі ж дії, що і для «Robot arm 1».

Створюємо новий проект під назвою «Links» (рис. 3.28).

Рисунок 3.28 – Сцена вибору деталей маніпулятора

Завантажуємо деталі, задній фон і інтерфейс в сцену, створюємо програмний об'єкт Canvas, з ним кілька компонентів «Image» с назвами «задній фон», «інтерфейс», «панель», «вихід», «RobotArm».

До інспектора підключаємо компоненти Scroll Rect і Scrollbar, для ефекту прокручування сторінки.

Створюємо C# скрипт для активація кнопок «RobotArm», «вихід».

Перейшовши у настройки програми Build Setting додаємо всі сцени по порядку (рис. 3.29).

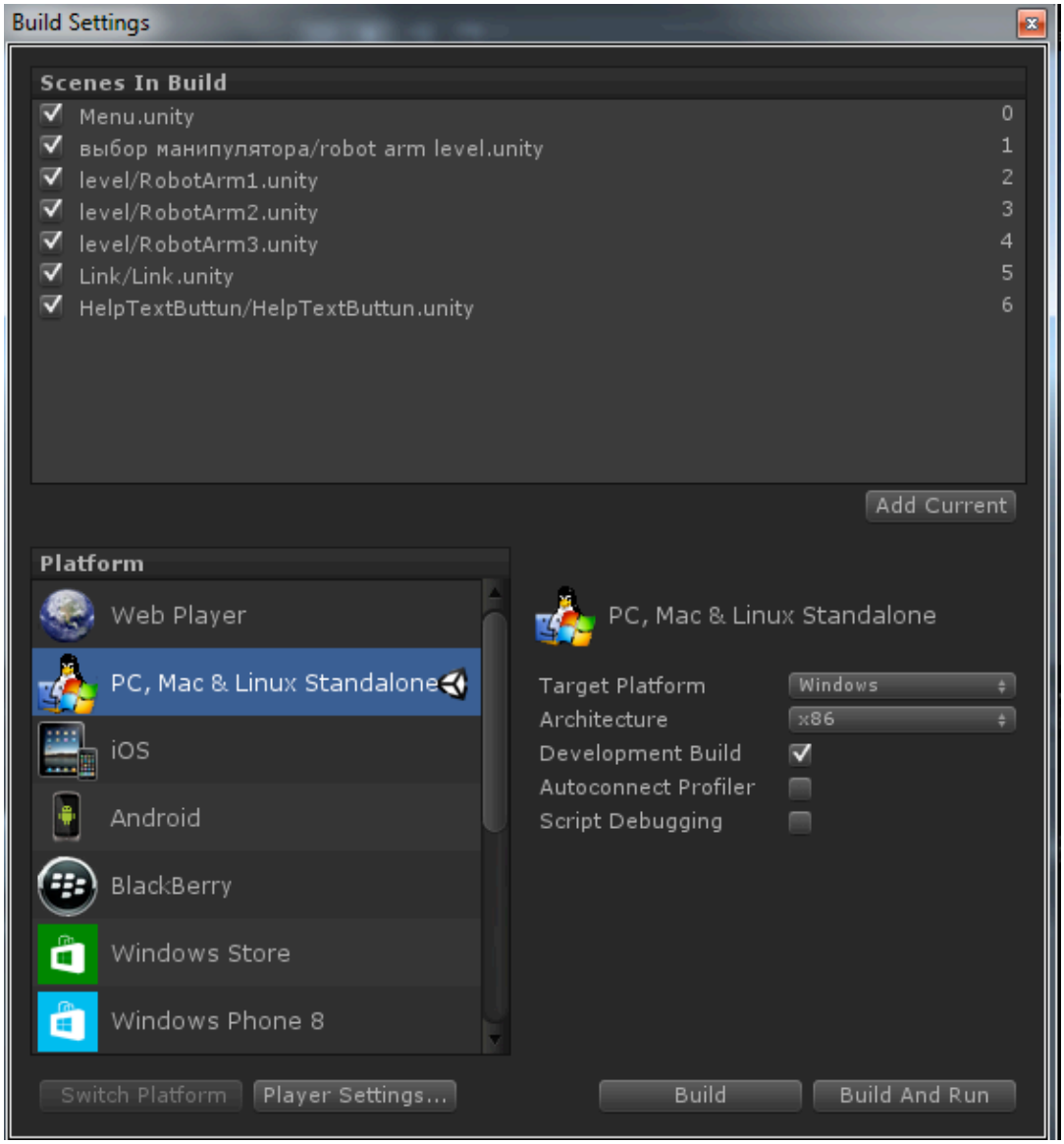

Рисунок 3.29 – Складання програми

Після цього збираємо і запускаємо нашу програму для перевірки функціональності її роботи (рис. 3.30).

Вибираємо оптимальні налаштування для комп'ютера розширення екрану «Screen resolution», та деталізація програми «Graphics quality» і запускаємо програму.

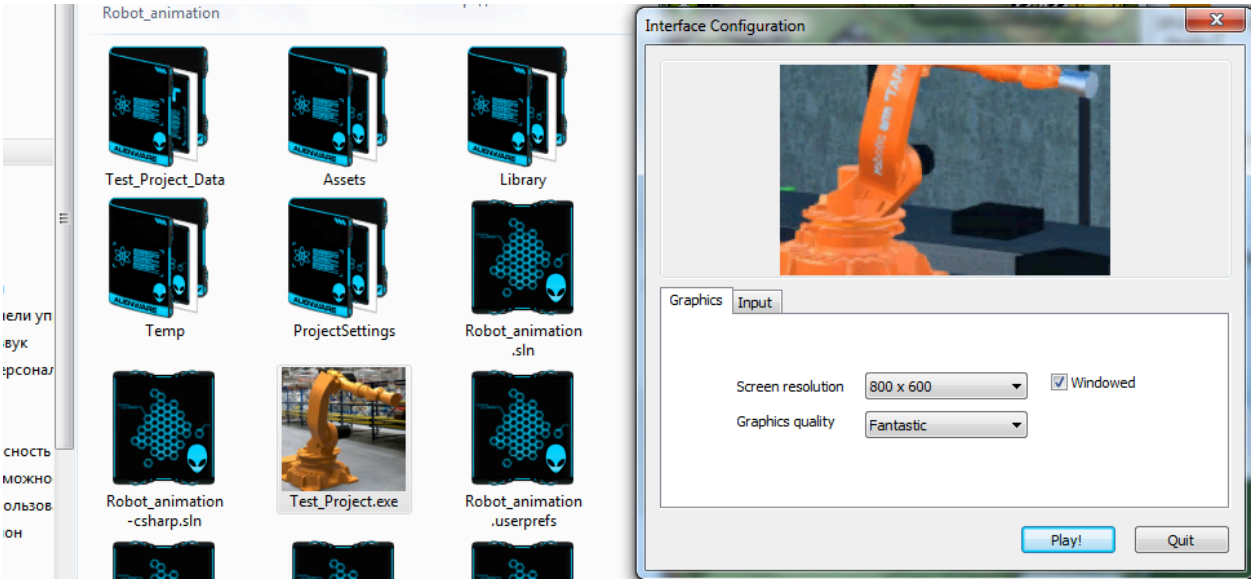

Рисунок 3.30 – Перевірка програми на працездатність

ВИСНОВКИ

У випускній роботі бакалавра виконано розробку інтерфейса та інформаційної системи оператора для конструкторського бюро виробничого підприємства.

Був проведений аналіз аналогічних інтерфейсів, обрано оптимальні параметри для програми.

Основними функціями інтерфейсу є конструювання і моделювання роботи маніпулятора у віртуальному середовищі.

Інтерфейс створено за допомогою кількох програм, такі як Unity, Adobe After Effects CC, 3ds Max 2014, Adobe Photoshop CC 2014, Mono Development, у яких були проведені роботи щодо створення декількох видів маніпуляторів, їх сцен роботи, створення самого інтерфейсу, написання коду управління всіх функцій і маніпуляції, а також комбінування 3Д моделі з кодуванням.

Зокрема з використання запровадженого інтерфейсу можливо:

- конструювати різних типів маніпулятори, для використання в різних умовах експлуатації;
- перевіряти роботу маніпулятора в віртуальній середовищі;
- визначати кут повороту маніпулятора;
- максимальній вантажопідйомність;
- кількість осей використання маніпулятором.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АСУ – автоматизована система управління;

БД – база даних;

ЕОМ – електронна обчислювальна машина;

КІ – користувальницький інтерфейс;

ПЗ – програмне забезпечення;

ПІ – програмна інженерія;

САПР – система автоматизованого проектування;

ТЗ – технічне завдання;

ТП – технологічний процес;

MS Windows – Microsoft Windows.

СПИСОК ЛІТЕРАТУРИ

1. 1. Невлюдов, І. Ш. Методичні вказівки до підготовки випускної кваліфікаційної роботи бакалавра для студентів усіх форм навчання напрямку підготовки 6.050902 «Радіоелектронні апарати» [Текст] / І. Ш. Невлюдов, С. П. Новоселов, Г. В. Пономарьова. – Х. : ХНУРЕ, 2010. – 39 с.

2. 2. Невлюдов, І. Ш. Основи виробництва електронних апаратів [Текст] / І.Ш. Невлюдов. – Х. : Компанія СМІТ, 2005. – 592 с.

3. Тео, М. Разработка пользовательского интерфейса [Текст] / М. Тео. – М.: ДМК пресс, 2001. – 150 с.

4. Тереза, Н. Проектирование веб-интерфейсов [Текст] / Н. Тереза. – М.: Символ-Плюс, 2010. – 210 с.

5. Тидвелл, Д. Разработка пользовательских интерфейсов [Текст] / Д. Тидвелл. – М.: 2008. – 185 с.

6. Разработка интерфейса пользователя [Электронный ресурс] / Режим доступа: http://sea1608.narod.ru/interface.html.

7. Разработка интерфейса пользователя [Электронный ресурс] – Режим доступа: http://pandia.ru/text/78/247/74988.php.

8. Hansen. Пользователь инженерных принципов для интерактивных систем [Текст] / Пер. с англ. - М.: AFIPS Press, 1971. – 523 с.

9. Heckel, Paul. Элементы понятным программным обеспечением дизайн [Текст] / Пер. с англ. - М.: WarnerBooks, 1984. – 345 с.

10. Гультяев, А.К. Проектирование и дизайн пользовательского интерфейса [Текст] / А.К. Гультяев, В.А. Машин. – М.: КОРОНА принт, 2000. – 155 с.

11. Елена, К. Самоучитель After Effects [Текст] / К. Елена. – Санкт-Петербург : Adobe Press 2004. – 150 с.

12. Christiansen, M. Adobe After Effects CS4 Visual Effects and Compositing Studio Techniques [Текст] / M. Christiansen. – M.: Adobe Press 2009. – 497 с.

13. Верстак, В. 3ds max [Текст] / В. Верстак. – Санкт-Петербург: Autodesk Press 2008. – 250 с.

14. Шишанов, А. Дизайн интерьеров в 3ds Max 2008 [Текст] / А. Шишанов. – М.: Autodesk Press 2008. – 350 с.

15. Creighton, H.R. Unity 3D Game Development by Example Beginner's Guide [Текст] / H.R. Creighton. - . : Unity Press 2010. – 480 p.

16. Pereira, V. Learning Unity 2D Game Development by Example [Текст] / V. Pereira. – . : Unity Press 2014. – 295 p.