Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Циганок Євгена Дмитровича*
(ПІБ)

академічної групи *121М-19-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

на тему: *Методи, алгоритми і програмне забезпечення для розширення функцій САПР моделей реальних об'єктів в режимі доповненої реальності.*

_____ *Є.Д. Циганок*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	Проф. Мороз Б.І.			
економічний	Доц. Касьяненко Л.В.			

Рецензент				

Нормоконтролер	Доц. Сироткіна О.І.			

Дніпро
2020

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 20 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121М-19-1* *Циганок Євгену Дмитровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Методи, алгоритми і програмне забезпечення для розширення функцій САПР моделей реальних об'єктів в режимі доповненої реальності.*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 22.10.2020 р. № 888-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес демонстрації моделей створених в механічному САПР в режимі доповненої реальності.

Предмет досліджень – методи демонстрації моделей створених в САПР Autodesk Inventor в режимі доповненої реальності.

Мета роботи – підвищення ефективності роботи САПР Autodesk Inventor для демонстрації тривимірних об'єктів будь-якої складності в режимі доповненої реальності.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна отриманих результатів кваліфікаційної роботи визначається тим, що були вдосконалені методи, що дозволяють розглядати тривимірні моделі створені в САПР Autodesk Inventor

Практична цінність полягає в тому, що розроблений в рамках роботи додаток дозволяє розглядати створені моделі в режимі доповненої реальності і дозволяє демонструвати розробляємий продукт на всіх етапах його проєктування без

необхідності його виготовлення, тим самим зменшуючи витрати на виготовлення і транспортування вироби.

4 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	23.09.2020-30.09.2020
Дослідження існуючих підходів в розробці додатків використовують функціонал доповненої реальності	01.10.2020-30.10.2020
Створення плагіна для САПР який дозволяє переглядати моделі в режимі доповненої реальності	1.11.2020-04.12.2020

Завдання видав _____ *Мороз Б.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Циганок Є.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *23.09.2020 р.*

Термін подання кваліфікаційної роботи до ЕК *07.12.2020*

РЕФЕРАТ

Пояснювальна записка: 88 с., 25 рис., 1 табл., 4 дод., 70 джерел.

Об’єкт дослідження: процес демонстрації тривимірної моделі в режимі доповненої реальності.

Предмет дослідження: методи демонстрації моделі створеної в САПР Autodesk Inventor в режимі доповненої реальності.

Мета магістерської роботи: підвищення ефективності роботи САПР Autodesk Inventor для демонстрації тривимірних об'єктів будь-якої складності, в режимі доповненої реальності.

Методи дослідження. При вирішенні поставлених завдань виконано аналіз і наукове узагальнення літературних джерел по вихідним посилам досліджень.

Наукова новизна отриманих результатів дипломної роботи визначається тим, шо вперше розроблена система що дозволяє розглядати тривимірні моделі створені в САПР Autodesk Inventor.

Практична цінність полягає в тому, що розроблений в рамках роботи додаток дозволяє розглядати створені моделі в режимі доповненої реальності дозволяє демонструвати розробляється продукт на всіх етапах проєктування без необхідності його виготовлення, тим самим зменшуючи витрати на виготовлення і транспортування вироби.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: плагін, САПР, доповнена реальність, валідація, конвертація.

ABSTRACT

Explanatory note: 88 p., 25 fig., 1 tabl., 4 app., 70 sources.

Objects of research: process of demonstration of three-dimensional model in the Augmented reality mode.

Subject of research: methods demonstrating three-dimensional models, created in Autodesk Inventor CAD tool, in the Augmented reality mode.

Purpose of the master's work: increase in the efficiency of Autodesk Inventor CAD tool for demonstration of three-dimensional models of any complexity in the Augmented Reality mode.

Methods of research: in the solution of the tasks set, analysis and scientific generalization of the literature on the initial premises of the research was carried out.

Scientific novelty: system which allows to view three dimensional models created in Autodesk Inventor CAD tool was developed for the first time.

Practical significance: application developed during this thesis allows viewing created models in Augmented reality mode, also it allows to demonstrated product under development on all the stages of development without the necessity to produce it, henceforth lowering the expenses on production and shipment of object.

In the section "Economics" the calculations of the complexity of software development, the cost of creating software and the duration of its development, as well as marketing studies of the market for the created software product were conducted.

List of keywords: plugin, add-in, cad, augmented reality, validation, conversation.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

AR – Augmented reality;

HMD - Head-mounted display;

XML – eXtensible Markup Language;

UML – Unified modelling language;

САПР – Система автоматизованого проєктування;

API – Application programming interface;

QR – Quick Response Code;

DLL – Dynamic Link Library.

ВСТУП

Актуальність дослідження. В рамках кваліфікаційного проєкту на тему “методи алгоритми і програми для розширення функцій САПР і відображення моделей реальних об'єктів в режимі доповненої реальності” досліджується поняття доповненої реальності і розробляється плагін для САПР Autodesk Inventor здатний використовувати доповнену реальність. Плагін розроблений в процесі роботи над дипломним проєктом дозволить істотно спростити розробку певних деталей, машин або механізмів, над якими працюють кілька команд інженерів які знаходяться в різних місцях, в подібному випадку розробляємо ПЗ поліпшить комунікації між командами, і нівелює необхідність транспортування частини роботи зробленої однією командою, іншій команді.

Основним призначенням розробленого програмного забезпечення являється надання користувачеві можливості затверджувати, конвертувати і експортувати моделі створені в САПР Autodesk Inventor, для відображення та взаємодії з ними в режимі доповненої реальності.

Розроблене програмне забезпечення допоможе істотно спростити розробку певних деталей, машин або механізмів, над якими працюють кілька команд інженерів які знаходяться в різних місцях, в подібному випадку розробляємо ПО поліпшить комунікації між командами, і нівелює необхідність транспортування частини роботи зробленої однією командою, іншій команді, також розроблення інформаційна система може буду використана і інших галузях, коли треба уявити, як ще не існуюча річ, буде виглядати у реальному оточенні.

Мета магістерської роботи: підвищення ефективності роботи САПР Autodesk Inventor для демонстрації тривимірних об'єктів будь-якої складності в режимі доповненої реальності.

Об’єкт дослідження: процес демонстрації САПР моделі в режимі доповненої реальності.

Предмет дослідження: методи демонстрації моделі створеної в САПР Autodesk Inventor в режимі доповненої реальності.

Методи дослідження: при вирішенні поставлених завдань виконано аналіз і наукове узагальнення літературних джерел по вихідним посилам досліджень.

Наукова новизна полягає у тому, що вперше розроблена система що дозволяє розглядати тривимірні моделі створені в САПР Autodesk Inventor.

Практичне значення: полягає в тому, що розроблений в рамках роботи додаток дозволяє розглядати створені моделі в режимі доповненої реальності та дозволяє демонструвати розробляємий продукт на всіх етапах проєктування без необхідності його виготовлення, тим самим зменшуючи витрати на виготовлення і транспортування вироби.

Особистий внесок автора полягає в розробці теоретичної та практичної частини магістерської роботи, в дослідженні і систематизації знання про існуючі методики та реалізації додатку, який використовує технологію доповненої реальності.

Структура та обсяг дипломної роботи. Робота складається з вступу, трьох розділів і висновків. Містить 88 сторінок друкованого тексту, в тому числі 58 сторінок тексту основної частини з 25 рисунками, списку використаних джерел з 70 найменуваннями на 5 сторінках, 3 додатків на 13 сторінках.

РОЗДІЛ 1
ДОПОВНЕНА РЕАЛЬНІСТЬ ЯК ФОРМА ПОДАННЯ ІНФОРМАЦІЇ

1.1. Визначення доповненої реальності

Доповнена реальність - це інтерактивний досвід реального середовища, де об’єкти, що перебувають у реальному світі, доповняються генеровану комп'ютером інформацією, іноді через різні сенсорні модальності, включаючи зорові, слухові, тактильні, сомато-сенсорні та нюхові. Доповнену реальність можна визначити як систему, яка виконує три основні функції: поєднання реального та віртуального світів, взаємодію в реальному часі та точну 3D-реєстрацію віртуальних та реальних об’єктів. Накладена сенсорна інформація може бути конструктивною (додавати інформацію до природного середовища) або деструктивною (тобто маскувати природне середовище). Цей досвід безперешкодно переплітається з фізичним світом так, що сприймається як захоплюючий аспект реального середовища. Таким чином, доповнена реальність змінює постійне сприйняття реального середовища, тоді як віртуальна реальність повністю замінює реальне середовище користувача на модельоване. Доповнена реальність пов’язана з двома в основному синонімічними термінами: змішаною реальністю та компьютерно-опосредованною реальністю.

Першочерговою цінністю доповненої реальності є спосіб, в якому компоненти цифрового світу поєднуються у сприйнятті людиною реального світу не як простий показ даних, а завдяки інтеграції захоплюючих відчуттів, які сприймаються як природні частини навколишнє середовище. Найдавніші функціональні системи AR, що забезпечували вражаючий досвід змішаної реальності для користувачів, були винайдені на початку 1990-х років, починаючи з системи віртуальних світильників, розробленої в лабораторії Армстронг ВПС США в 1992 році. Комерційний досвід доповненої реальності вперше був представлений у сфері розваг та ігор. Згодом додатки доповненої реальності охопили комерційні галузі, такі як освіта, зв'язок, медицина та розваги. В освіті

доступ до вмісту можна отримати шляхом сканування або перегляду зображення за допомогою мобільного пристрою або за допомогою безмаркерових AR-методів.

Доповнена реальність використовується для покращення природного середовища або ситуацій та надання збагаченого сприйняттям досвіду. За допомогою передових AR-технологій (наприклад, додавання комп'ютерного зору, включення AR-камер у програми для смартфонів та розпізнавання об'єктів) інформація про навколишній реальний світ користувача стає інтерактивною та цифровою. Інформація про навколишнє середовище та його об'єкти накладається на реальний світ. Ця інформація може бути віртуальною або реальною, наприклад бачення іншої реальної відчутої або виміряної інформації, такої як електромагнітні радіохвилі, накладені в точному вирівнюванні з тим, де вони насправді знаходяться в просторі. Розширена реальність також має великий потенціал у зборі та обміні мовчазними знаннями. Доповнена реальність використовується, як правило, у реальному часі та в семантичному контексті з елементами навколишнього середовища. Імерсійна інформація про сприйняття іноді поєднується з додатковою інформацією, як-от оцінки в прямому ефірі відео спортивної події. Це поєднує в собі переваги як технології доповненої реальності, так і технології прозорих дисплеїв.

1.1.1. Відмінність віртуальної реальності від доповненої

У віртуальній реальності сприйняття реальності користувачами повністю базується на віртуальній інформації. У доповненій реальності користувачеві надається додаткова інформація, що генерується комп'ютером, та покращує їх сприйняття реальності. Наприклад, в архітектурі VR можна використовувати для створення прохідного моделювання внутрішньої частини нової будівлі; а AR можна використовувати для відображення структур та систем будівлі, накладених на реальний погляд. Інший приклад - використання службових

програм. Деякі додатки AR, такі як Augment, дозволяють користувачам застосовувати цифрові об'єкти в реальних середовищах, дозволяючи компаніям використовувати пристрої доповненої реальності як спосіб попереднього перегляду своїх продуктів у реальному світі. Подібним чином, доповнену реальність також можна використовувати для демонстрації того, як можуть виглядати товари в середовищі для споживачів, як це демонструють такі компанії, як Mountain Equipment Co-op або Lowe, які використовують доповнену реальність, щоб дозволити клієнтам переглянути, як можуть виглядати їхні товари вдома завдяки використанню 3D-моделей.

Доповнена реальність відрізняється від віртуальної реальності тим, що в доповненої реальності частина навколишнього середовища насправді є «реальною» і просто додає прошарок віртуальних об’єктів до реального середовища. З іншого боку, в віртуальної реальності навколишнє середовище є абсолютно віртуальним. Демонстрація того, як AR-шари об'єктів у реальному світі можна побачити за допомогою ігор з доповненою реальністю. WallaMe - це програма для доповненої реальності, яка дозволяє користувачам приховувати повідомлення в реальному середовищі, використовуючи технологію геолокації, щоб дозволити користувачам приховувати повідомлення, де б вони не побажали у світі. Такі програми мають багато застосувань у світі, в тому числі для активізму та художньої виразності.

1.2. Характеристики доповненої реальності

У цьому розділі розглядаються характеристики систем доповненої реальності та проблеми проєктування, що виникають при побудові системи доповненої реальності. Розділ 1.2.1 описує основні характеристики доповненої реальності. Існує два способи здійснити це доповнення: оптичні або відео технології. У розділі 3.2 розглядаються їх характеристики та відносні сильні та слабкі сторони. Поєднання реального та віртуального створює проблеми з

фокусом та контрастом (Розділ 3.3), а деякі програми вимагають, щоб портативні системи AR були справді ефективними (Розділ 3.4). Нарешті, розділ 3.5 узагальнює характеристики шляхом порівняння вимог AR до вимог до віртуального середовища.

1.2.1. Доповнення

Окрім додавання об’єктів до реального середовища, Доповнена реальність також може їх видалити. Поточна робота зосереджена на додаванні віртуальних об'єктів до реального середовища. Однак графічні накладення можуть також використовуватися для видалення або приховування частин реального середовища від користувача. Наприклад, прибрати робочий стіл у реальному навколишньому середовищі, намалювати зображення справжніх стін і підлоги за письмовим столом і «намалювати» це над справжнім письмовим столом, ефективно видаляючи його з поля зору користувача. Зробити це інтерактивно в AR-системі буде набагато складніше, але для ефективного видалення може не знадобитися фото реалістичність.

Розширена реальність може стосуватися всіх органів чуття, а не лише зору. Наразі дослідники зосереджувались на поєднанні реальних та віртуальних зображень та графіки. Однак доповнену реальність можна розширити, включивши звук. Користувач повинен носити навушники, оснащені мікрофонами зовні. Навушники додадуть синтетичний, спрямований тривимірний звук, тоді як зовнішні мікрофони будуть виявляти вхідні звуки від навколишнє середовище. Це дало б системі можливість замаскувати або приховати вибрані реальні звуки з навколишнього середовища, генеруючи маскуючий сигнал, який точно скасовував вхідний реальний звук. Хоча зробити це буде непросто, можливо, це можливо. Інший приклад - хаптика. Рукавички з пристроями, що забезпечують тактильний зворотний зв'язок, можуть збільшити реальні сили в навколишньому середовищі. Наприклад, користувач може проведіть рукою по поверхні справжнього письмового столу. Моделювати таку

тверду поверхню віртуально досить складно, але насправді це легко зробити. Тоді тактильні ефектори в рукавичці можуть посилити відчуття письмового столу, можливо, змушуючи його відчувати себе шорстким в певних місцях. Ця можливість може бути корисною в деяких додатках, наприклад, надання додаткової сигналізації про те, що віртуальний об'єкт знаходиться в певному місці на реальному столі.

1.2.2. Оптичні і відеотехнічні системи доповненої реальності

Основним проєктним рішенням у побудові системи AR є те, як здійснити поєднання реального та віртуального. Доступні два основні варіанти: оптична та відеотехніка. Кожен має певні переваги та недоліки. Цей розділ порівнює ці два і зазначає компроміси. Прозорий HMD - це один пристрій, що використовується для поєднання реального та віртуального. Стандартні HMD із закритим видом не дозволяють мати прямий погляд на реальний світ. На відміну від цього, пронизаний HMD дозволяє користувачеві бачити реальний світ за допомогою віртуальних об'єктів, накладених за допомогою оптичних або відеотехничних засобів.

Оптичні прозорі HMD працюють, розміщуючи оптичні суматори перед очима користувача. Ці комбайнери є частково пропускаючими, так що користувач може дивитись прямо через них, щоб побачити реальний світ. Оптичні суматори також частково відображають, так що користувач бачить, як віртуальні зображення відбиваються від комбайнерів від моніторів, встановлених на голові. Цей підхід за своєю суттю схожий з головними дисплеями (HUD), які зазвичай використовуються у військових літаках, за винятком того, що комбайнери прикріплені до голови. Таким чином, оптичні прозорі HMD іноді описуються як "HUD на голові". На рисунку 11 представлена концептуальна схема оптичного прорізу HMD. На рисунку 12 показані два оптичні прозорі HMD, виготовлені Х'юзом Електроніка.

Оптичні комбайнери зазвичай зменшують кількість світла, яке користувач бачить із реального світу. Оскільки комбайнери діють як напівсріблені дзеркала, вони пропускають лише частину світла з реального світу, щоб вони могли відображати частину світла від моніторів в очі користувача. Наприклад, HMD, описаний в, передає близько 30% надходить світла з реального світу. Вибір рівня змішування - проблема дизайну. Більш складні комбайнери можуть змінювати рівень внесків залежно від довжини хвилі світла. Наприклад, такий комбайн може бути встановлений для відображення всього світла певної довжини хвилі і жодної на будь-якій іншій довжині хвилі. Це було б ідеально з монохромним монітором. Практично все світло від монітора буде відбиватися в очах користувача, тоді як майже все світло з реального світу (за винятком певної довжини хвилі) досягне очей користувача. Однак більшість існуючих оптичних прозорих HMD дійсно зменшуються кількість світла з реального світу, тому вони відірвані від сонячних окулярів, коли відключено живлення.

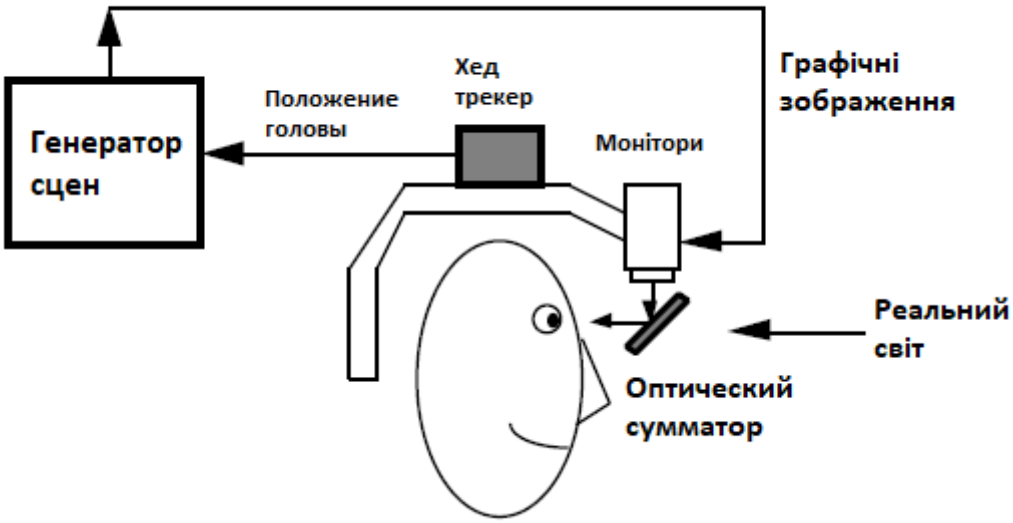

Рис. 1.1. Концептуальна схема оптичного прозорого наголовного дисплея

На відміну від цього, відеопрозорі HMD працюють, поєднуючи HMD із закритим видом з однією або двома відеокамерами, встановленими на голові. Відеокамери надають користувачеві погляд на реальний світ. Відео з цих камер

поєднується з графікою зображення, створені генератором сцен, поєднуючи реальне та віртуальне. Результат є надсилається на монітори перед очима користувача в HMD із закритим видом. Малюнок 13 показує концептуальну схему відеопрозорого HMD. На малюнку 14 показано фактичний відеопрозорий HMD, з двома відеокамерами, встановленими поверх льотного шолома.

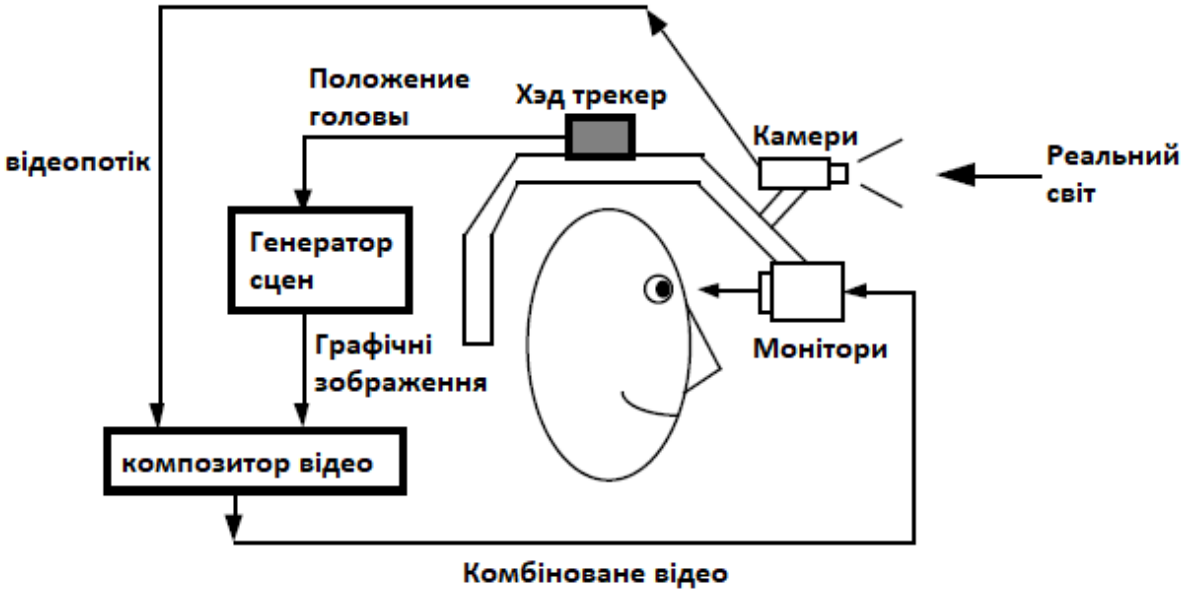

Рис. 1.2. Концептуальна схема наголовного відео дисплея

Композиція відео може бути виконана більш ніж одним способом. Простий спосіб - використовувати кольорову маніпуляцію: техніку, що використовується у багатьох відео спецефектах. Тло графічних зображень комп'ютера встановлено в певний колір, скажімо зелений, який не використовує жоден з віртуальних об'єктів. Потім крок комбінування замінює всі зелені зони на відповідні частини з відео реального світу. Це має наслідком накладання віртуальних об'єктів на реальний світ. Більш досконала композиція використовує інформацію про глибину. Якби система мала інформацію про глибину на кожному пікселі для зображень у реальному світі, вона могла б поєднувати реальні та віртуальні зображення за допомогою піксельного порівняння глибини. Це дозволило б реальним об'єктам покривати віртуальні об'єкти і навпаки.

Системи AR також можна будувати, використовуючи конфігурації на основі монітора, замість прозорих HMD. На малюнку 15 показано, як може бути побудована система на основі монітора. У цьому випадку одна або дві відеокамери оглядають навколишнє середовище. Камери можуть бути статичними або мобільними. У мобільному корпусі камери можуть рухатися, прикріплені до робота, з відстеженням їх розташування. Відео реального світу та графічні зображення, сформовані генератором сцен, поєднуються, як і у випадку з відео HMD, і відображаються на моніторі перед користувачем. Користувач не носить пристрій відображення. За бажанням, зображення можуть відображатися на стерео на моніторі, що вимагає, щоб користувач носив пару стереоокулярів. Малюнок 16 показує зовнішній вигляд системи ARGOS, яка використовує конфігурація на основі монітора.

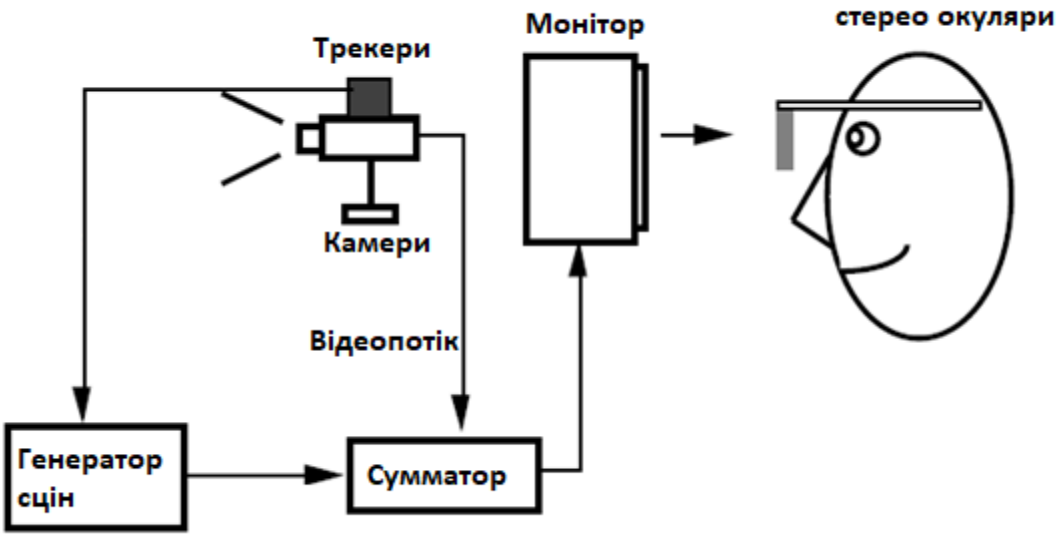

Рис 1.3. Концептуальна схема AR на основі монітора

Нарешті, можлива також оптична конфігурація на основі монітора. Це схоже на малюнок 11, за винятком того, що користувач не носить монітори або комбайнери на голові. Натомість монітори та комбайни закріплені в просторі, і користувач позиціонує голову, щоб дивитись через комбайнери. Це типово для

головних дисплеїв на військових літаках, і принаймні одна така конфігурація була запропонована для медичного застосування.

Решта цього розділу порівнює відносні переваги та недоліки оптичного та відео підходу, починаючи з оптичного. Оптичний підхід має наступні переваги перед відеопідходом:

1) Простота: Оптичне змішування простіше і дешевше, ніж змішування відео. Оптичні підходи мають лише один "потік" відео, про який слід турбуватися: графічні зображення. Реальний світ бачиться безпосередньо через комбайнери, і така затримка часу, як правило, становить кілька наносекунд. З іншого боку, змішування відео повинно мати справу з окремими відеопотоками для реальних та віртуальних зображень. Обидва потоки мають властиві затримки в десятки мілісекунд. Оцифровка відеозображень зазвичай додає принаймні один час затримки кадру до відеопотоку, де час кадру - це скільки часу потрібно для повного оновлення зображення. Монітор, який повністю оновлює екран із частотою 60 Гц, має час кадру 16,67 мс. Два потоки реальних та віртуальних зображень повинні бути належним чином синхронізовані або в результаті тимчасових спотворень. Крім того, оптичні прозорі HMD з вузькими комбінованими полями зору пропонують види реального світу, які мало спотворюють. Відеокамери майже завжди мають деяку кількість спотворень, які необхідно компенсувати, разом з будь-якими спотвореннями від оптики перед пристроями відображення. Оскільки для відео потрібні камери та комбінатори, в яких не потрібні оптичні підходи, відео, ймовірно, буде дорожчим та складнішим у побудові, ніж оптичні системи.

2) Розширення: Змішування відео обмежує роздільну здатність того, що бачить користувач, як реальну, так і віртуальну, до роздільної здатності пристроїв відображення. На поточних дисплеях ця роздільна здатність набагато менша, ніж роздільна здатність фовеї. Оптичний прозорий екран також показує графічні зображення з роздільною здатністю пристрою відображення, але погляд користувача на реальний світ не погіршується. Таким чином, відео зменшує

роздільну здатність реального світу, в той час як оптичне просвічування - ні. 3) Безпека: відеопрозорі HMD - це, по суті, модифіковані HMD із закритим видом. Якщо живлення відключено, користувач фактично сліпий. Це загрожує безпеці в деяких додатках. На відміну від цього, коли живлення віднімається від оптичного HMD, користувач все ще має прямий погляд на реальний світ. Потім HMD стає парою важких сонцезахисних окулярів, але користувач все ще може бачити.

Змішування відео має наступні переваги перед оптичним змішуванням:

1) Гнучкість у композиційних стратегіях: Основна проблема оптичного проникнення полягає в тому, що віртуальні об'єкти не закривають повністю об'єкти реального світу, оскільки оптичні комбайнери пропускають світло як з віртуальних, так і з реальних джерел. Скласти оптичний прозорий HMD, який може вибірково відключати світло від реального світу, складно. У звичайній оптичній системі об’єкти створені для фокусування лише в одній точці оптичного шляху: оці користувача. Будь-який фільтр, який вибірково блокує світло, повинен бути розміщений в оптичному тракті в точці, де зображення знаходиться у фокусі, що, очевидно, не може бути оком користувача. Отже, оптична система повинна мати два місця, де зображення знаходиться у фокусі: в оці користувача та в точці гіпотетичного фільтра. Це робить оптичну конструкцію набагато складнішою та складнішою. Жоден існуючий оптичний прозорий HMD таким чином не блокує надходить світло. Таким чином, віртуальні об’єкти здаються привидними та напівпрозорими. Це пошкоджує ілюзію реальності, оскільки оклюзія є однією з найсильніших глибинних ознак. На відміну від цього, прозоре відео набагато гнучкіше, як воно об’єднує реальні та віртуальні зображення. Оскільки і реальне, і віртуальне доступні в цифровій формі, композитори, що пропускають відео, можуть на основі пікселів за пікселем взяти реальне, віртуальне або якесь поєднання між ними для імітації прозорості. Завдяки цій гнучкості, відеопрозорість може в кінцевому підсумку створити більш привабливе середовище, ніж оптичні прозорі підходи.

2) Широке поле зору: Спотворення в оптичних системах є функцією радіальної відстані від оптичної осі. Чим далі відволікає погляд від центру огляду, тим більшими є спотворення. Оцифроване зображення, зроблене через спотворену оптичну систему, може бути не спотворене, застосовуючи методи обробки зображення для деформації зображення, за умови, що оптичні спотворення добре охарактеризовані. Це вимагає значних обчислень, але це обмеження буде менш важливим у майбутньому, оскільки комп'ютери стануть швидшими. Важче побудувати широкі дисплеї поля зору за допомогою оптичних прозорих методів. Будь-які спотворення уявлення користувача про реальний світ повинні бути виправлені оптично, а не цифровим способом, оскільки система не має оцифрованого зображення реального світу для маніпулювання. Складна оптика дорога і додає ваги HMD. Широкі системи поля зору є винятком із загальної тенденції оптичних підходів бути простішими та дешевшими, ніж відеопідходи.

3) Реальні та віртуальні затримки перегляду можна зрівняти: Відео пропонує підхід для зменшення або уникнення проблем, спричинених тимчасовим невідповідністю реального та віртуального зображень. Оптичні прозорі HMD пропонують майже миттєвий погляд на реальний світ, але відкладений погляд на віртуальний. Це тимчасове невідповідність може спричинити проблеми. За допомогою відеопідходів можна затримати відео реального світу відповідно до затримки віртуального потоку зображень.

4) Додаткові стратегії реєстрації: В оптичному прозорому режимі єдина інформація, яку система має про розташування голови користувача, надходить від головного трекера. Змішування відео забезпечує ще одне джерело інформації: оцифроване зображення реальної сцени. Це оцифроване зображення означає, що відеопідходи можуть використовувати додаткові стратегії реєстрації, недоступні для оптичних підходів. Розділ 4.4 описує їх більш докладно.

5) Простіше підібрати яскравість реальних та віртуальних об'єктів: це обговорюється в розділі 3.3. І оптичні, і відеотехнології мають свою роль, і вибір

технології залежить від вимог до програми. У багатьох прототипах механічного складання та ремонту використовуються оптичні підходи, можливо, через проблеми із вартістю та безпекою. У разі успіху обладнання довелося б тиражувати у великій кількості для оснащення робітників на заводському поверсі. На відміну від цього, більшість прототипів для медичних застосувань використовують відеопідходи, ймовірно, для гнучкості поєднання реального та віртуального та для додаткових запропонованих стратегій реєстрації.

1.2.3. Фокус і контраст

Фокус може бути проблемою як для оптичного, так і для відеопідходу. В ідеалі, віртуальний повинен відповідати реальному. У системі, що базується на відео, об'єднане віртуальне та реальне зображення буде проєктуватися на однакову відстань за допомогою монітора або оптики HMD. Однак, залежно від глибини різкості та налаштувань фокусування, частина реального світу може не знаходитись у фокусі. У типовому графічному програмному забезпеченні все рендериться за допомогою моделі-обскури, тому всі графічні об'єкти, незалежно від відстані, знаходяться у фокусі. Щоб подолати це, графіка може бути відтворена для імітації обмеженої глибини різкості, а відеокамера може мати об'єктив з автофокусом.

В оптичному корпусі віртуальне зображення проєктується на деякій відстані від користувача. Цю відстань можна регулювати, хоча вона часто є фіксованою. Тому, хоча реальні об’єкти знаходяться на різній відстані від користувача, усі віртуальні об’єкти проєктуються на однакову відстань. Якщо віртуальна та реальна відстані не співпадають для конкретних об’єктів, які переглядає користувач, можливо, неможливо буде чітко переглянути одночасно обидва. Контраст - це ще одна проблема через великий динамічний діапазон у реальному середовищі та в тому, що може виявити людське око. В ідеалі яскравість реальних та віртуальних об’єктів повинна відповідати відповідним

чином. На жаль, у гіршому випадку це означає, що система повинна відповідати дуже великому діапазону рівнів яскравості. Око - це логарифмічний детектор, де найяскравіше світло, яке воно може впорати, приблизно на одинадцять порядків перевищує найменше, включаючи як пристосовані до темряв, так і пристосовані до світла очі.

У будь-якому одному адаптаційному стані око може охоплювати близько шести порядків. Більшість пристроїв відображення не можуть наблизитися до такого рівня контрасту. Це особлива проблема оптичних технологій, оскільки користувач має прямий погляд на реальний світ. Якщо реальне середовище занадто яскраве, це змиє віртуальне зображення. Якщо реальне середовище занадто темне, віртуальне зображення змиє реальний світ. Проблеми з контрастом не такі серйозні з відео, оскільки самі відеокамери мають обмежену динамічну реакцію, а вигляд як реального, так і віртуального генерується монітором, тому все повинно бути відсічене або стиснене в динамічний діапазон монітора.

1.2.4. Переносимість

Практично у всіх системах віртуального середовища користувачеві не рекомендується багато ходити. Натомість користувач здійснює навігацію, "літаючи" по навколишньому середовищу, гуляючи по біговій доріжці або керуючи макетом транспортного засобу. Незалежно від технології, результат полягає в тому, що користувач залишається на одному місці в реальному світі. Однак деякі додатки AR потребують підтримки користувача, який буде ходити по великому середовищу. AR вимагає, щоб користувач фактично знаходився там, де має відбутися завдання. "Політ", як виконується в системі VE, більше не є варіантом. Якщо механіку потрібно перейти на інший бік реактивного двигуна, вона повинна фізично переміщати себе та пристрої відображення, які вона носить.

Таким чином, AR-системи дадуть перевагу портативності, особливо здатності ходити на відкритому повітрі, далеко від контрольованого середовища. Генератор сцени, HMD та система відстеження повинні бути автономними та здатними переживати вплив навколишнього середовища. Якщо ця можливість буде досягнута, стане доступним багато інших додатків, які ще не були випробувані. Наприклад, здатність коментувати навколишнє середовище може стати в нагоді солдатам, туристам або туристам у незнайомому новому місці.

1.2.5. Порівняння з віртуальними середовищами

Загальні вимоги AR можна узагальнити, порівнявши їх із вимогами до віртуального середовища для трьох основних підсистем, які вони потребують.

1) Генератор сцен: На сьогодні рендеринг не є однією з основних проблем AR. Системи VE мають набагато вищі вимоги до реалістичних зображень, оскільки вони повністю замінюють реальний світ віртуальним середовищем. В AR віртуальні зображення лише доповнюють реальний світ. Отже, потрібно намалювати менше віртуальних об'єктів, і вони не обов'язково повинні бути реалістично відтворені, щоб задовольнити цілі програми. Наприклад, у програмах анотацій може бути достатньо тексту та тривимірних каркасних креслень. В ідеалі фотореалістичні графічні об’єкти можна було б легко об’єднати з реальним середовищем (див. Розділ 7), але спочатку потрібно вирішити більш основні проблеми.

2) Пристрій відображення: Пристрої відображення, що використовуються в AR, можуть мати менш жорсткі вимоги, ніж вимагають системи VE, знову ж тому, що AR не замінює реальний світ. Наприклад, монохромні дисплеї можуть бути достатніми для деяких додатків AR, тоді як практично всі системи VE сьогодні використовують повнокольорові. Оптичні прозорі HMD з малим полем зору можуть бути задовільними, оскільки користувач все ще може бачити реальний світ своїм периферійним зором; прозорий HMD не відключає звичайне

поле зору користувача. Крім того, роздільна здатність монітора в оптичному прозорому HMD може бути нижчою, ніж те, що дозволить користувач у програмі VE, оскільки оптичний прозорий HMD не зменшує роздільну здатність реального середовища.

3) Відстеження та зондування: Хоча в попередніх двох випадках AR вимагав нижчих вимог, ніж VE, це не стосується відстеження та зондування. У цій галузі вимоги до AR набагато жорсткіші, ніж до систем VE. Основною причиною цього є проблема реєстрації, яка описана в наступному розділі. Інші фактори, що підвищують вимоги до відстеження та зондування.

1.3. Розробка додатків доповненої реальності

Беручи до уваги той факт, що за своєю суттю додатки які використовують можливості доповненої реальності використовують розпізнавання образів, робить їх розробку, налагодження, тестування і підтримку неймовірно складним завданням.

Всі завдання пов'язані з відображенням и взаємодією з контентом в режимі доповненої реальності тісно пов'язані з розпізнаванням образів. Розпізнавання образів має дуже багато сфер застосування, та характеризується здатністю програми пов'язувати одержуване на вході зображення з якоюсь внутрішньою сутністю. Обробка зображення - дуже складаний и ресурсномісткій процес, тому для його спрощення варто розділити його на кілька груп.

Фільтрація: методи, які дозволяють виділити на збережений цікавлять області, без їх аналізу. Велика частина цих методів застосовує якесь єдине перетворення на всі точки зображення. На рівні фільтрації аналіз зображення не проводитися, но точки, які проходять фільтрацію, можна розглядати як області з особливими характеристиками.

Логічна обробка результатів фільтрації: на цьому етапі здійснюється перетворення дозволяють перейти від зображення до властивостей об'єктів, або до самих об'єктів.

Навчання: методи, які не працюють безпосередньо з збережений, но які дозволяють приймати рішення. В основному це різноманітні методи машинного навчання и прийняття рішень.

На сьогоднішній день існує безліч SDK, які значно спрощують розробку, додатків доповненої реальності, и дозволяють розробнику реалізовувати вище перераховані пункти, та використовувати досить простий API для роботи з зображеннями.

1.4. Висновки до першого розділу

Беручи до уваги все вищесказане, можна зробити висновок, що на сьогоднішній момент технологія віртуальної реальності виявляється на ранніх етапах свого розвитку. Не дивлячись на те, що дуже багато було зроблено для розвитку повної реальності, безліч проблем все ще вимагає рішень. Після вирішення основних проблем з доповнену реальністю кінцевою метою буде створення віртуальних об’єктів, настільки реалістичних, що їх практично неможливо відрізнити від реального середовища. Фотореалізм був продемонстрований у художніх фільмах, але досягти цього в інтерактивному додатку буде набагато складніше. Умови освітлення, відбиття поверхні та інші властивості повинні вимірюватися автоматично в режимі реального часу. Більш витончені можливості освітлення, текстурування та затінення повинні працювати з інтерактивною швидкістю у майбутніх генераторах сцен. Реєстрація повинна бути майже ідеальною, без втручання та коригування вручну. Хоча це складні проблеми, вони, мабуть, не нездоланні.

РОЗДІЛ 2
ПРОГРАМУВАННЯ ДОДАТКІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1. Основні поняття

Існує безліч способів розробки програми використовують можливості надання, у всіх є свої плюси і мінуси. Однак варто зазначити, що в переважній більшості випадку розробники використовують уже готові рішення Надаємо вже готові рішення в області вжиття доповненої реальності (див. Наступний розділ) такі рішення зазвичай являють собою динамічно скомпільованими бібліотеками (.dll) або іншими формами виконуваних файлів, які можуть містити в собі скомпільовані об'єкти і методи які дозволяють реалізувати трекінг об'єктів і отрисовку трьох мірних зображення в поле зору користувача. Існує величезна кількість причин по яких такий підхід кращий, оскільки розробка системи доповненої реальності має на увазі розробку системи що використовує комп'ютерний зір, що у свою чергу потребує використання достатньо низькорівневих API, таких як OpenCV.

Існує величезна кількість бібліотек класів, який надають розробникові доступ до функціонала доповнене реальності, проте в основі кожної з них будуть лежать одні й ті ж методи і алгоритми. І також усі ці бібліотеки забезпечують розробника одним і тим же функціоналом, наприклад всі вони надають можливість виводити тривимірну модель в поле зору користувача, а також здійснювати трекінг мітки (якщо така мається) для використання її в якості початку локальної системи координат.

2.1.1. OpenCV

Строго кажучи OpenCV НЕ являється бібліотекою або фреймворком доповненої реальність, але оскільки для розробки системи доповненої реальності

необхідне застосування технології комп'ютерного зору OpenCV часто лежить в основі бібліотек дозволяє працювати з доповненою реальністю. Конкчно ж вохможно користуватися тільки OpenCV для розробки теж функціоналу який надають бібліотеки представлені нижче, однак обсяг такої роботи буде колосальним.

2.1.2. EasyAR

EasyAR, на жаль, не надає ніяких інструментів, що полегшують життя розробнику. Все що ми маємо - це SDK, інструкції по запуску їх прикладів, невелика документація, що описує базові принципи розпізнавання об'єктів і документація для C ++, її у принципі досить для знайомства з класами SDK, тому що весь environment - це обгортка над плюсами.

Цільові об'єкти з себе представляють картинки і їх опис в форматі json. Для роботи нам потрібні CameraDevice для надання доступу до камери, CameraFrameStreamer для передачі даних з камери в трекер і сам ImageTracker. Для відстеження цільових об'єктів ми безперервно опитуємо статус трекера.

У наданому прикладі у них є якась сутність, назвемо її AREnvironment, що містить в собі ініціалізацію SDK і трекерів, роботу з камерою, ініціалізацію OpenGL. Також даний клас описує, що і де рендерить (вийшов якийсь God object, в якому намішано більшість роботи). Це AR оточення агрегує GLView, GLView в свою чергу викликається з активують в onResume і onPause.

2.1.3. Wikitude

Даний фреймворк залишив найкраще враження, ймовірно, тому він і коштує дорожче інших. Почнемо з того, що вони надають онлайн-студію для накладення простих статичних об'єктів доповненої реальності. Для цього потрібно завантажити цільове зображення в студію, додати об'єкти AR,

згенерувати JavaScript код і вставити в свій проєкт. Таким чином весь рендеринг лягає на ArchitectView від Wikitude JS SDK. Студія виглядає наступним чином:

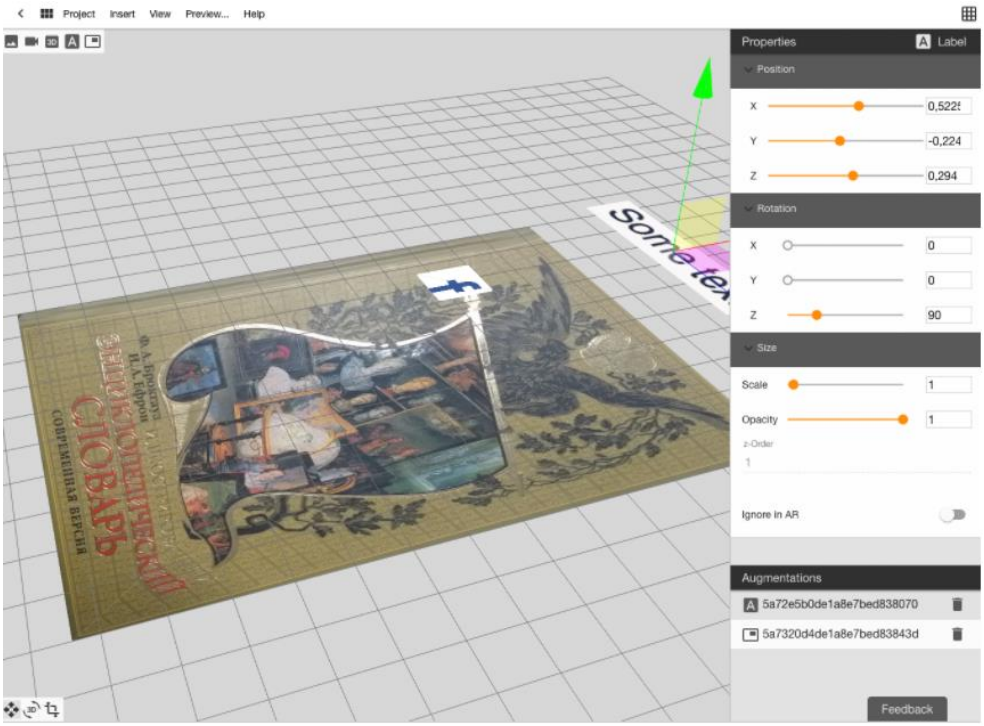

Рис. 2.1. Студія середовища розробки доповненої реальності Wikitude

Для розміщення простих статичних об'єктів досить в нашому UI компоненті форматувати ArchitectView з license key розробника і передати шлях до згенеровані студією JS AR environment. У найпростішому випадку, це все, що потрібно для розпізнавання Image targets і накладення доповненої реальності. В такому випадку Wikitude JS Android SDK бере на себе всю роботу, пов'язану розпізнаванням і рендерингом.

При необхідності можна з JS коду передавати json-об'єкти в нативний код. Для цього потрібно імплементувати ArchitectJavaScriptInterfaceListener і додати слухача в ArchitectView.

Але, що робити, якщо ми хочемо взяти на себе рендеринг і додати динаміку нашої AR або якось кастомизировать поведінку? Для цього можна написати свій JS код, використовуючи Wikitude JS SDK або ж, якщо нам потрібна

продуктивність і повний контроль над рендерингом, то Wikitude надає Native SDK.

Для роботи з даними розширенням нам буде потрібно передати wikitudeSdk реалізацію інтерфейсу ImageTrackerListener, який містить коллбекі розпізнавання об'єкта, втрати і т.д. і посилання на наш кастомний рендеринг (це може бути InternalRendering або Externalrendering). По суті це означає, що в internal реалізація GLSurfaceView надається Wikitude SDK, а в external весь openGL лягає на плечі розробника. Студія в такому випадку використовується тільки для генерації wtc бази цільових зображень, які ми потім відстежуємо.

Також даний фреймворк підтримує роботу з unity і інтеграцію C ++ плагінів.

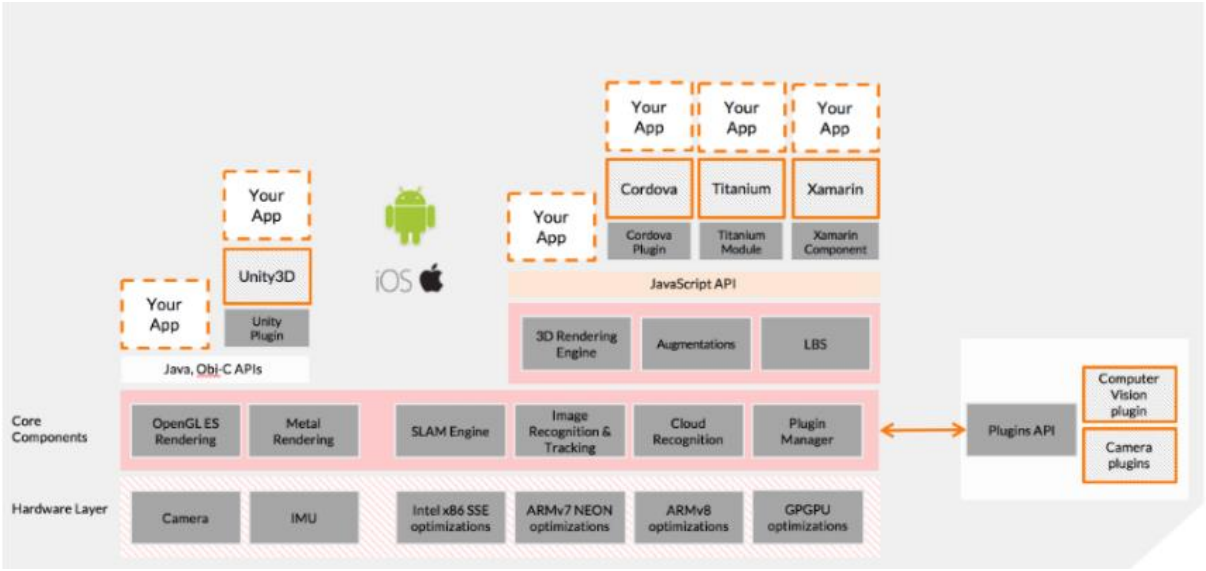

Рис. 2.2. Архитектура библиотеки Wikitude

2.2. Програмне забезпечення та алгоритми

Ключовим показником систем AR є те, наскільки реально вони інтегрують збільшення з реальним світом. Програмне забезпечення повинно отримувати координати реального світу, незалежно від зображень камери та камери. Цей процес називається реєстрацією зображень і використовує різні методи

комп'ютерного зору, в основному пов'язані з відстеженням відео. Багато методів комп’ютерного зору доповненої реальності успадковуються від зорової одометрії. Аугограма - це комп'ютерне зображення, яке використовується для створення AR. Аугографія - це наукова та програмна практика створення аугограм для AP.

Зазвичай ці методи складаються з двох частин. Перший етап полягає у виявленні точок інтересу, фідуціальних маркерів або оптичного потоку на зображеннях камери. На цьому кроці можна використовувати такі способи виявлення функцій, як виявлення кутів, виявлення крапок, виявлення або поріг країв та інші методи обробки зображень. Другий етап відновлює реальну систему координат з даних, отриманих на першому етапі. Деякі методи припускають, що на сцені присутні об'єкти з відомою геометрією (або довірчими маркерами). У деяких з цих випадків структуру 3D сцени слід розрахувати заздалегідь. Якщо частина сцени невідома, одночасна локалізація та відображення (SLAM) можуть відобразити відносні позиції. Якщо інформація про геометрію сцени відсутня, використовується структура з методів руху, наприклад, регулювання пучка. Математичні методи, що використовуються на другому етапі, включають: проєктивну (епіполярну) геометрію, геометричну алгебру, подання обертання з експоненціальною картою, фільтри Кальмана та частинок, нелінійну оптимізацію, надійну статистику.

У доповненій реальності розрізняють два різні режими відстеження, відомі як маркерні та маркерні. Маркери - це візуальні сигнали, які запускають відображення віртуальної інформації. Можна використовувати аркуш паперу з чітко вираженою геометрією. Камера розпізнає геометрію, визначаючи конкретні точки на кресленні. Безмаркерне відстеження, яке також називають миттєвим відстеженням, не використовує маркери. Натомість користувач розміщує об’єкт у вигляді камери, бажано в горизонтальній площині. Він використовує датчики в мобільних пристроях для точного виявлення реального середовища, наприклад розташування стін та точок перетину

Мова розмітки розширеної реальності (ARML) - це стандарт даних, розроблений в рамках Відкритого геопросторового консорціуму (OGC), який складається з граматики розширюваної мови розмітки (XML) для опису розташування та зовнішнього вигляду віртуальних об'єктів на сцені, а також прив'язок ECMAScript до дозволяють динамічний доступ до властивостей віртуальних об'єктів.

Щоб забезпечити швидкий розвиток додатків з доповненою реальністю, з’явилися деякі набори для розробки програмного забезпечення (SDK)

2.3. Висновки до другого розділу

Беручи до уваги все вищесказане можна зробити висновок про те, що проєктування, розробка, налагодження і тестування додатків використовують технологію доповненої реальності може бути дуже важкою і витратною завданням проте, з урахуванням того факту, що існує безліч вже готових рішень, розробка подібних програм не складає праці, єдина трудність закладається в пошуку бібліотеки, яка буде відповідати всім вимоги розробляється.

Також варто взяти до уваги, що завдяки бурхливому розвитку мікропроцесорної техніки, на момент написання даної роботи, існує можливість розробки додатків доповненої реальності в якості виконуваної платформи використовує мобільні пристрої, а також браузери. Однак варто точно розуміти область застосування розробляється і на підставі цього приймати рішення про те під яку платформу для якої операційної системи буде розроблятися додаток. Так, додаток аналізує вхідний відеопотік в режимі реального часу і відстежує безліч цілей (targets) може споживати достатньо велику кількість ресурсів і бути занадто вимогливим до обчислювальної потужності пристрою на якому воно буде виконуватися. Але навіть не дивлячись на це існує безліч областей які доповнена реальність зможе істотно поліпшити.

РОЗДІЛ 3
ПРАКТИЧНЕ ЗАСТОСУВАННЯ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1. Розробка плагіна для механічного САПР Autodesk Inventor

Беручи до уваги той факт, що доповнена реальність дозволяє з неймовірною чіткістю візуалізувати віртуальні тривимірні об'єкти в реальному оточення, було вирішено розробити додаток дозволяють, з застосуванням доповненої реальності, візуалізувати тривимірні моделі створені в механічному САПР Autodesk Inventor. Оскільки формат файлів застосовуваних у вищезгаданій САПР являться пропрієтарним, то для розробки даної програми необхідно використовувати прикладної інтерфейс програмування для Inventor, які надає об'єкти і методи за допомогою яких можливо здійснити конвертацію в формат файлу контент якого можливо відобразити в режимі доповненої реальності. Розроблене додаток, що є плагіном, дозволить істотно заощадити на розробці деталей машин і механізмів, оскільки, коли раніше було необхідні виготовляти розробляємий виріб, зараз потрібно лише скористатися плагіном розробленим в рамках дипломної роботи. У наступних розділах буде представлено опис САПР Autodesk Inventor, а також надана інформація щодо його API.

3.1.1. САПР Autodesk Inventor

Autodesk Inventor - САПР для створення і вивчення поведінки цифрових прототипів виробів і деталей. Розробник компанія Autodesk.

Використовується в основному в машинобудуванні. В комплект входить декілька продуктів: Autodesk Inventor Suite, Autodesk Inventor Routed Systems Suite (проєктування кабельних і трубопровідних систем, в том числі для розводки складних ділянок трубопроводів, електричних кабелів і проводів),

Autodesk Inventor Simulation Suite (засоби моделювання руху і аналізу навантажень, які спрощують вивчення поведінки виробу в реальних умовах ще на стадії проєктування).

Інструменти Inventor забезпечують повний цикл проєктування і створення конструкторської документації:

— 2D / 3D-моделювання; створення виробів з листового матеріалу та отримання їх розгорток;

— розробка електричних і трубопровідних систем;

— проєктування оснащення для лиття пластмасових виробів;

— динамічне моделювання; параметричний розрахунок напружено-деформованого стану деталей і зборок;

— візуалізація виробів;

— автоматичне отримання і оновлення конструкторської документації (оформлення по ЕСКД).

Компонувальні схеми, що поєднують окремі деталі і вузли. Користувачі можуть перевірити можливість збірки об'єкта, додати і позиціонувати нові частини, а також усунути перешкоди між частинами проєкту.

Ливарні форми і оснащення. Програма автоматизує ключові аспекти процесу проєктування ливарних форм під тиском. Користувачі можуть швидко створювати і перевіряти конструкції форм, а потім експортувати їх в Autodesk Moldflow.

Деталі з листового матеріалу. Спеціальна середу проєктування виробів з листового матеріалу автоматизує багато аспектів роботи. Користувачі можуть створювати деталі розгортки, гнуті профілі, формувати фланці шляхом 3D-моделювання і вставляти в деталі спеціалізовані кріпильні елементи.

Генератор рам служить для проєктування каркасів (рам) на основі стандартних профілів. Рами створюються шляхом розміщення стандартних сталевих профілів на каркасі. Формування кінцевих умов спрощується завдяки

наявності стандартних опцій для кутових з'єднань і з'єднань встик. Користувачі можуть створювати власні профілі і додавати їх в бібліотеку.

Кабельні та трубопровідні системи. Середа для створення трубопроводів допомагає проєктувати їх таким чином, щоб вписати в складну збірку або обмежений простір. Вона включає бібліотеку стандартних фітингів, труб і шлангів, і забезпечує створення складальних креслень, які оновлюються в міру змін вихідної 3D-моделі.

3.1.2. Інтерфейс прикладного програмування Autodesk Inventor

API, або інтерфейс прикладного програмування - це термін, який використовується для опису функціональних можливостей графічного редактора Inventor, що надається в рамках програми. Наприклад, ви можете використовувати API, щоб написати програму, яка реалізує повторювані операції. Autodesk Inventor є спільною системою САПР, а це означає, що він не призначений для будь-якої конкретної галузі або використовуються для моделювання тільки певних видів продукції. Наявність API дозволяє додавати функціональні можливості Inventor, специфічні для ваших індивідуальних потреб. Inventor API реалізується на основі COM технології і надає різні способи звернутися до даних Inventor за допомогою різних видів надбудовних додатків (Plug-in modules).

Білі поля представляють компоненти, які забезпечують API - Inventor і «Apprentice Server». Синій циліндр в підставі представляє Inventor дані, до яких Ви звертаєтесь - це деталі, збірки, і т.д. Все жовті поля представляють програми, які Ви записуєте. Коли одне поле включає інше поле, це вказує, що включене поле виконується в тому ж самому процесі як поле, що включає його. Наприклад, VBA виконується в тому ж самому процесі як Inventor. Програма, яка запускається в тому ж процесі, працює значно швидше, ніж програма, яка запускається «поза процесом».

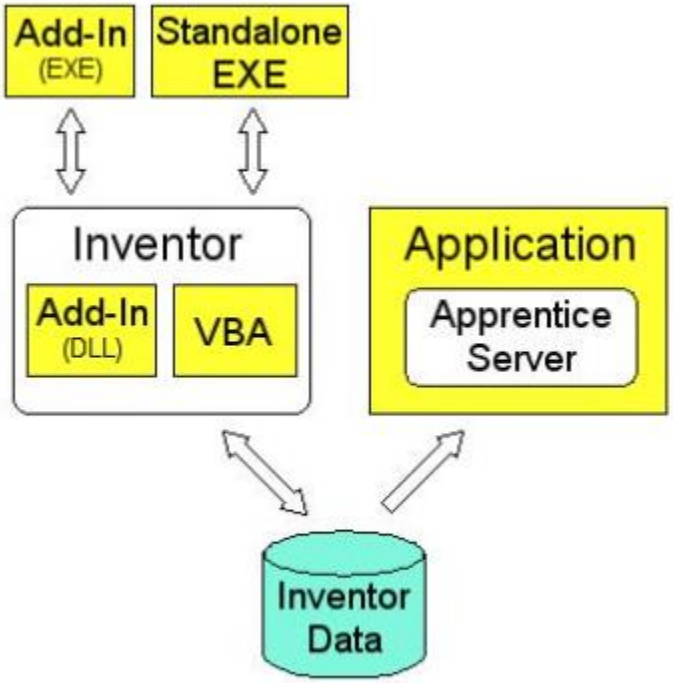

Рис. 3.1. Високорівневе уявлення Inventor API

Зазвичай, вибираючи VBA, керуються такими перевагами:

- VBA є середовищем програмування, яка доступна зсередини Inventor і не потрібно, щоб Ви купували додаткову мову програмування.
- Ви можете впровадити програми в межах документа, для обробки якого вони розроблені. При цьому Ви можете також зберегти програми в окремих файлах, щоб вони могли використовуватися в різних документах.
- VBA виконується в тому ж самому процесі як Inventor.
- VBA має той же самий доступ до всіх особливостей API як будь-який з інших способів звернення до API (за винятком Add-Ins).

Автономний EXE – програма, яка виконується самостійно і з'єднується з Inventor. Цей тип програми зазвичай використовується в разі, коли Ви маєте програму, яка використовує Inventor, але має власний інтерфейс і не вимагає, щоб користувач в інтерактивному режимі працював з Inventor. Наприклад, програма може контролювати нові записи, які будуть додані в базу даних. Коли

новий запис створена, програма запускає Inventor, відкриває бажаний документ і друкує його - все без будь-якої взаємодії з користувачем.

Apprentice сервер – підмножина Inventor, яке не має інтерфейсу користувача і запускається в одному процесі з іншим додатком. Єдиний спосіб взаємодіяти з Apprentice сервером - через його API. Apprentice сервер набагато більш ефективний, ніж використання Inventor, оскільки без інтерфейсу користувача операції виконуються швидше. Бібліотека типу Apprentice містить обмежений набір об'єктів, підтримуваних бібліотекою типу Inventor. Apprentice забезпечує доступ до структури збірки, B-Rep геометрії і рендер стилям - тільки для читання. Доступ до посилань файлу, атрибутам і властивостями документа - для читання і запису.

Add-In – особливий вид додатка, який автоматично завантажується при запуску Inventor, має високу продуктивність і представляється користувачеві як би частиною Inventor. Inventor's Add-In - це спосіб з'єднається з Inventor і використовувати його API. За допомогою Add-In додатків можна створювати команди. На діаграмі Add-In перерахований двічі - як DLL і як EXE. DLL додатки працюють в тому ж самому процесі як Inventor, EXE додатки зручні для налагодження.

3.2. Проєктування та розробка програмного продукту

3.2.1. Функціональне призначення програми

Призначений для створення програмного забезпечення представляє собою плагін для САПР Autodesk Inventor. Плагін розроблений в процесі роботи над кваліфікаційною роботою дозволити істотно спростити розробку питань комерційної торгівлі деталей, машин або механізмів, над якими працюють кілька команд інженерів які знаходяться в різних місцях, в подібному випадки розробляємо ПЗ поліпшіть комунікації між командами, и нівелює необхідність транспортування частини роботи зробленої однією командою , іншій команді.

Основним призначенням розробленого програмно забезпечення є надання користувачеві можливості затверджувати, конвертувати і експортувати моделі створені в САПР Autodesk Inventor, для відображення та взаємодії з ними в режимі доповненої реальності.

Розроблене програмне забезпечення допоможе істотно спростити розробку певних деталей, машин або механізмів, над якими працюють кілька команд інженерів які знаходяться в різних місцях, в подібному випадку розробляємо ПО поліпшить комунікації між командами, і нівелює необхідність транспортування частини роботи зробленої однією командою, іншій команді, також розроблення інформаційна система може буду використана і інших галузях, коли треба уявити, як ще не існуюча річ, буде виглядати у реальному оточенні.

3.2.2. Опис застосованих математичних методів

В даній інформаційній системі ні під час розробки, ні під час тестування та роботи додатка, не використовуються та не розглядаються жодні математичні методи.

3.2.3. Опис використаної архітектури та шаблонів проєктування

Виходячи з розміру програми, кількості частин з якого воно складається, а також філософії використання деяких зовнішніх залежностей таких як API Autodesk Inventor та Unity, практичні неможливо виділити якусь єдину архітектуру.

Однак, на найвищому рівні виділити кілька модулів з яких складається система, на фізичному рівні модулі являють собою Динамічна бібліотека і виконуваний файл, що розглядається система складається з трьох динамічно підключених бібліотек.

— InventorPluginUI служить точкою входу додаток;

— PluginCore включає в себе інтерфейси і базові класи;

— UtilsLib містить велику кількість сервісних класів, які використовується в більшості модулів програми;

— UnityARCore - включає в себе сценарії движка Unity.

В процесі роботи над додаток були використані шаблони проєктування всіх типів, а саме: які породжують, структурні і поведінкові.

В процесі роботи над обробкою користувальницького введення в частині AR потрібна добивати можливість здійснювати певну кількість маніпуляцій з моделлю: збільшувати модель, зменшувати модель, пересувати, а також скасовувати більш ранні дії, після вивчення існуючих шаблонів проєктування, було вирішено використовувати шаблон проєктування команда.

Команда - шаблон проєктування, відноситься до класу шаблонів поведінки. Також відомий як Дія, Транзакція

Призначенням цього шаблона проєктування є інкапсуляцій запита у формі об’єкта дозволяючи тим самим задавати параметри клієнтів для обробки відповідних запитів, ставити запити у чергу або протоколювати їх, а також підтримувати скасовування операцій

В процесі розробки провідника налаштувань доповненої реальності виникла необхідність запам'ятовувати частина стану об'єкта, а саме матрицю трансформації і матрицю обертання, оскільки при наявності наскрізної навігації по меню потрібно відновлювати стан об'єкта.

Знімок — це шаблон проєктування, що відноситься до класу шаблонів поведінки і забезпечує можливість відновлення об'єкта до збереженого (попереднього) стану.

Не порушуючи інкапсуляції, фіксує та виносить за межі об'єкта його внутрішній стан так, щоб пізніше можна було відновити з нього об'єкт.

Слід використовувати шаблон Знімок у випадках, коли: необхідно зберегти миттєвий знімок стану об'єкта (або його частини), щоб згодом об'єкт можна було відтворити у тому ж самому стані;

На ранніх етапах розробки плагіна для Autodesk Inventor, сутність, метою якої було створення призначеного для користувача інтерфейсу, була сінглтоном, проте в подальшому було вирішено відмовитися від використання даного патерну.

Одинак — шаблон проєктування, відноситься до класу твірних шаблонів. Гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра.

Для деяких класів важливо, щоб існував тільки один екземпляр. Наприклад, хоча у системі може існувати декілька принтерів, може бути тільки один спулер. Повинна бути тільки одна файлова система та тільки один активний віконний менеджер.

Глобальна змінна не вирішує такої проблеми, бо не забороняє створити інші екземпляри класу.

Рішення полягає в тому, щоб сам клас контролював свою «унікальність», забороняючи створення нових екземплярів, та сам забезпечував єдину точку доступу. Це є призначенням шаблону Одинак.

Також неявно був використаний такий патерн, як DI.

Впровадження залежностей — це шаблон проєктування, в якому залежності (або сервіси) впроваджуються, або передаються по посиланню в залежний об'єкт (клієнт) і стають частиною клієнтського стану. Шаблон відокремлює створення залежностей клієнта від власної логіки клієнта, що дозволяє компонентам бути слабко зв'язаними і притримуватися принципів інверсії залежностей і єдиного обов'язку. Це суперечить анти-шаблону service locator, який дозволяє клієнтам знати про систему, що використовується для пошуку залежностей.

3.2.4. Опис використаних технологій та мов програмування

Оскільки логічно додаток можна розділити на плагін, який завантажується в Autodesk Inventor і служить фактичної точкою входу в додаток і на Unity частини дозволяє користувачеві налаштовувати режим доповненої реальності і взаємодіяти з моделлю в режимі доповненої реальності.

Слід окремо розглядатися технології, які були використані для розробки окремих частин програми.

Для розробки плагіна для САПР Autodesk Inventor використовувався мова програмування C #.

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research.

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Особливості мови:

- Портативність: C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR.
- Типи даних: C# підтримує строго типізовані неявні оголошення змінних з ключовим, словом, var і неявно типізовані масиви з ключовим, словом, new [], за яким слідує ініціалізатор колекції.

— Метапрограмування: метапрограмування через атрибути C# є частиною мови. Багато з цих атрибутів дублюють функціональні можливості директив препроцесора, орієнтованих на платформу GCC і VisualC ++.

— Властивості: C# надає властивості як синтаксичного цукру для загального шаблону, в якому пара методів, accessor (getter) і mutator (setter) інкапсулює операції по одному атрибуту класу.

Частина додатка, що залишилася написана за допомогою засобів ігрового движка Unity.

Unity — багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосунки працюють під системами Windows, OS X, Android, Apple, iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.

Є можливість створювати інтернет-додатки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Технічні характеристики

— сценарії на C#, JavaScript та Boo;
— ігровий рушій повністю пов'язаний із середовищем розробки;
— робота з ресурсами можлива через звичайний Drag&Drop;
— система успадкування об'єктів;
— підтримка імпортування великої кількості форматів файлів;
— система успадкування об'єктів;
— підтримка імпортування великої кількості форматів файлів;
— вбудований генератор ландшафтів;
— вбудована підтримка мережі.

Існує рішення для спільної розробки — Asset Server. Також можна використовувати зручний для користувача спосіб контролю версій. Наприклад, SVN або Source Gear.

Частина програми, безпосередньо відповідає за відображення раннє експортованої моделі на екрані використовує фреймоворк доповненої реальності EasyAr. Основне завдання даного фреймоворка полягає в розпізнання об'єктів, які є картинками і їх опис в форматі json.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація). Проєкт в Unity ділиться на сцени (рівні) — окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні ігрові об'єкти — тобто ті, які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у них є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов'язково присутній компонент Transform — він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об'єктів з видимою геометрією також за умовчанням присутній компонент Mesh Renderer, що робить модель видимою.

Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна — буде створено матеріал, з яким буде

призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Пакетний файл є свого роду файлом сценарію(скриптом) в DOS, OS/2 і Windows. Він містить набір команд, які виконуються командним інтерпретатором і зберігається як звичайний текстовий файл. Пакетний файл може містити команди, які інтерпретатор приймає в інтерактивному режимі і надає змогу створення умовних/безумовних переходів та циклів всередині пакетного файлу, наприклад «if», «for», «goto» та мітки.

Подібно до JCL та інших мов для мейнфреймів, пакетні файли були створені для полегшення виконання однотипних дій через створення сценаріїв (скриптів) для їх автоматизації. Коли запускається пакетний файл, shell-програма (зазвичай COMMAND.COM або cmd.exe) зчитує та виконує його команди рядок за рядком.

Розширення файлу .bat використовується в DOS та Windows. Windows NT та OS/2 також можуть мати розширення .cmd. Пакетні файли в інших системах мають і інші розширення, наприклад .btm в 4DOS, 4OS2.

Особливості виконання пакетний файлів змінились. Частка особливостей, що описані у статті справджуються для усіх пакетних файлів, а решта тільки для певних версій.

3.3. Фізична структура програми

Фізично додаток складається з чотирьох проєктів: InventorPlugin.dll, PluginCore.dll, UtilsLib.dll, UnityARCore.exe.

InvnentorPluginUI (рис. 3.2) служить точкою входу в додаток, містить в собі безліч класів та безпосередньо відповідає за взаємодію з API САПР Autodesk

Inventor, а саме за отрисовку компонентів для користувача інтерфейсу відповідає необхідного для даного плагіна.

Також класи з даного модуля пропонують своїм клієнтам можливості валідації і конвертації моделей в формат необхідний для її відображення в режимі доповненої реальності.

Також в даному класі був реалізований механізм лінковки моделі або складання моделей з відповідним їм QR кодом, слід зазначити що лінковка не має на увазі фізичну ін'єкцію файлу QR коду в файл моделі, замість це у властивостях моделі прописується шлях до QR коду.

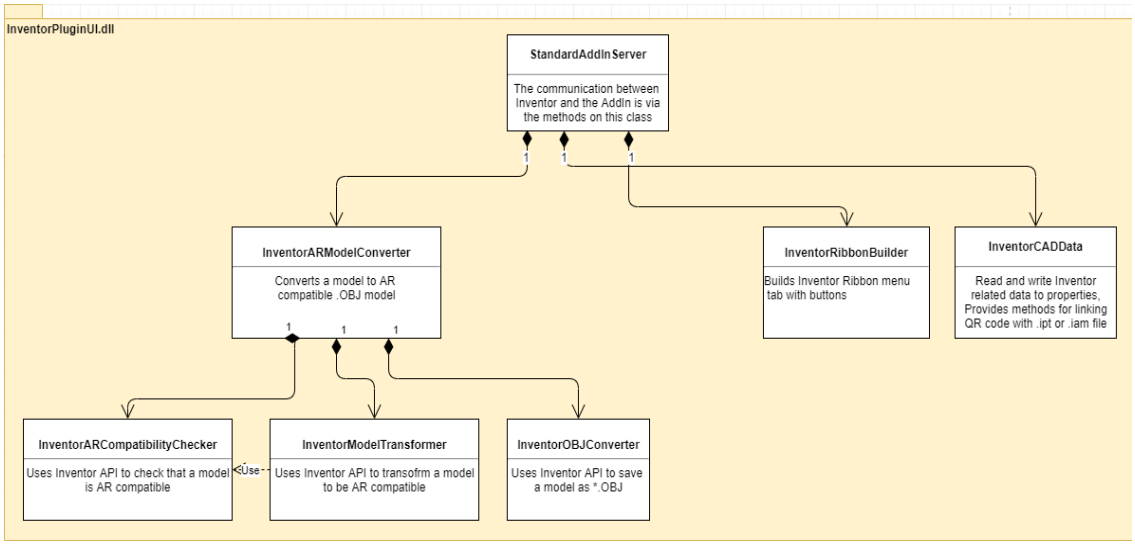

Рис. 3.2. Діаграма класів InventorPlugin.dll

PluginCoreManager (рис. 3.3) являє собою ядро, частина відповідає за взаємодію з Autodesk Inventor, містить безліч інтерфейсів і абстрактних класів, значно спрощують розробку і дозволяють в найкоротші терміни виконати модифікацію всього програми так що б воно працювало з будь-якої іншої САПР, без переписування всього коду програми.

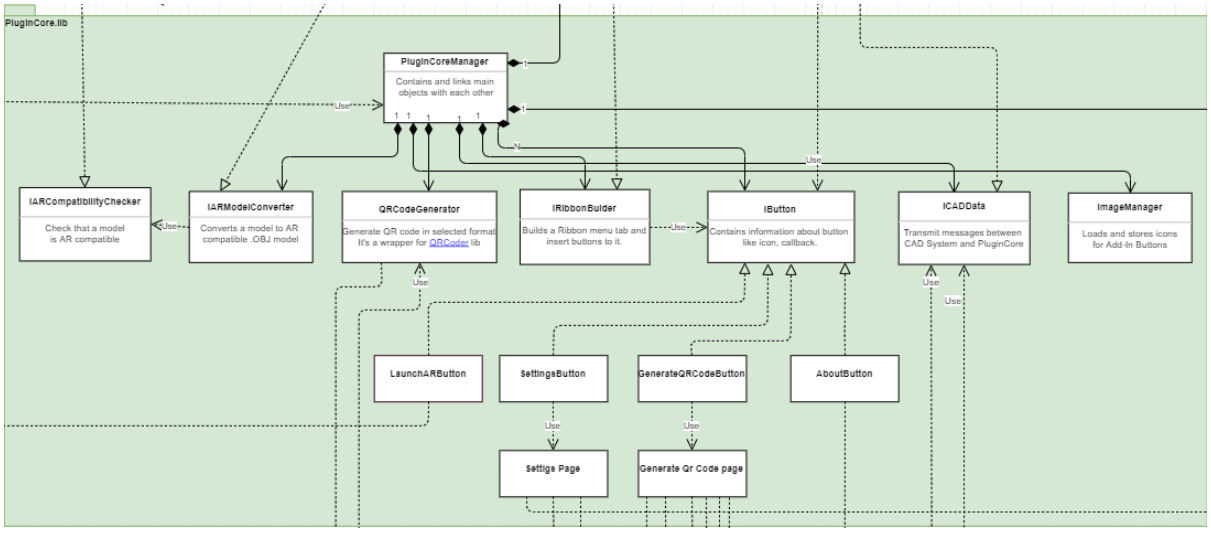

Рис. 3.3. Діаграма класів PluginCore.dll

UtlisLib (рис. 3.4) – є модулем якій включає в себе сервісні класи, використовувані всім додатком, містить отберткі над викликом системних функцій необхідних для Unity частини програми.

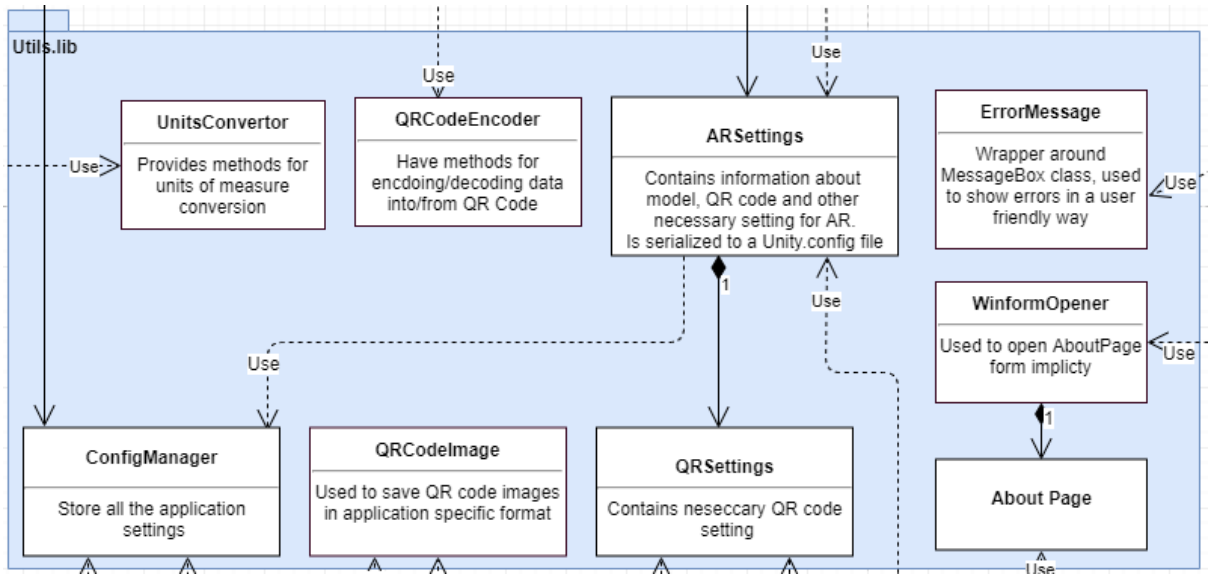

Рис. 3.4. Діаграма класів UtilsLib.dll

UnityARCore (рис. 3.5) – частина програми, що відповідає за роботу майстра налаштувань доповненої реальності і відображення моделі в режимі доповненої реальності, містить класи, що відповідають за асинхронну завантаження раніше експортованої моделі.

Містить безліч класів і сценаріїв движка Unity, що збирає і обробляє для користувача введення, а також подмодуль, який служить точкою входу для роботи з доповненої реальністю, а саме містить у собі клас, який є спадкоємцем від ARCameraDevice.

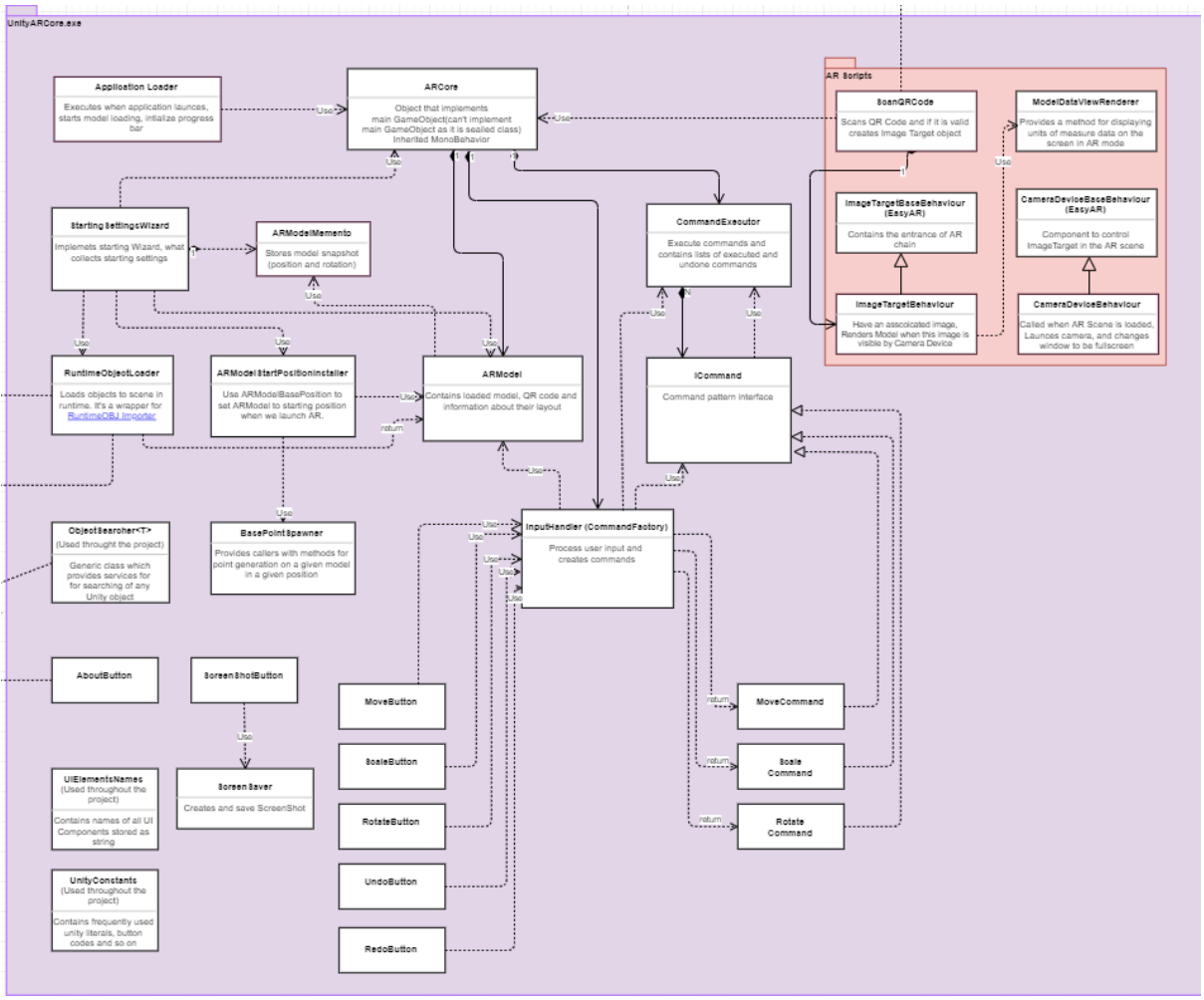

Рис. 3.5. Діаграма класів UnityARCore.exe

3.4. Опис логічної структури програми

Розроблена система дозволяє оператору здійснювати валідацію, конвертацію моделей, створених в САПР з Autodesk Inventor для подальшого їх перегляду в режимі доповненої реальності.

Розроблена система дає наступний функціонал:

— генерація QR коду;

— вибір одиниць виміру, в яких буде експортуватися модель;

— вибір моделі необхідної для відображення в режимі доповненої реальності;

— вибір точки на QR коді, щодо якої потрібно розміщувати модель;

— вибір точки на моделі, яка потрібна для позиціонування щодо точки на QR коду;

— зміни розміру моделі для відображення в режимі доповненої реальності. За замовчування моделі відображається в своєму теперішньому розмірі;

— збільшення моделі;

— зменшення моделі;

— редагування розташування моделі в просторі;

— скасування перерахованих вище операцій.

3.5. Виклик та завантаження програми

Оскільки розроблена система є плагіном, то для її запуску необхідна програма вміє викликати даний плагін, в даному випадку Autodesk Inventor.

Після відкриття Autodesk Inventor необхідно перейти на вкладку Tools (рис. 3.6).

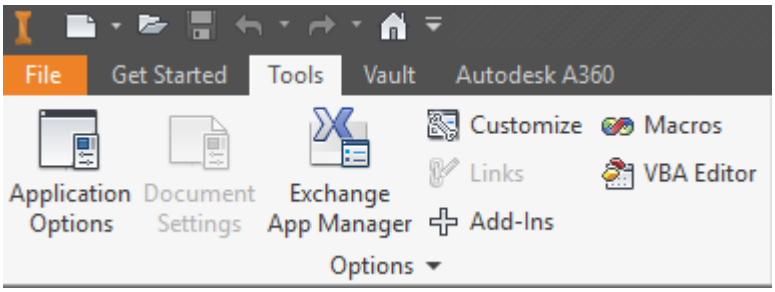

Рис. 3.6. Скріншот інтерфейсу користувача

Далі, в меню Options, необхідно перейти в розділ Add-Ins (рис 3.7) в якому будуть відображені всі виявлені і встановлені розширення для Autodesk Inventor.

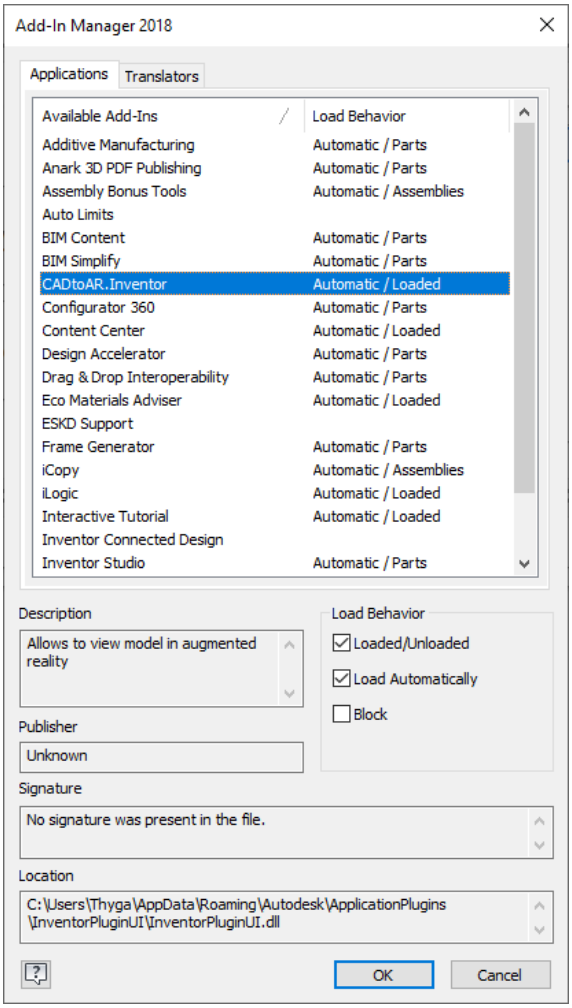

Рис. 3.7. Скріншот менеджера плагінів

В отриманому списку плагінів потрібно знайти плагін, який був розроблений в рамках даної дипломної роботи. Після вибору потрібно плагіна зі списку (CADtoAR-Inventor), не обходимо вказати, що даний плгаін потрібно завантажити. Після чого можна помітити нову вкладку (рис. 3.8).

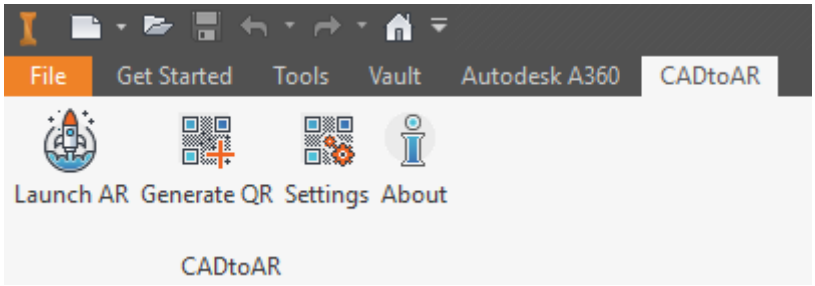

Рис. 3.8. Скріншот вкладки плагіну

3.6. Опис інтерфейсу користувача

Після запуску Autodesk Inventor з попередньо встановленим розробленим плагіном, у верхній частині риббона з'явиться вкладка "CADtoAR" (Рис 3.8), де знаходяться 4 елементи управління. Розглянемо їх права на ліво, починаючи з кнопки About.

При натисканні на кнопку About, відкривається діалогове вікно "About CADtoAR.Inventor" (рис. 3.9)

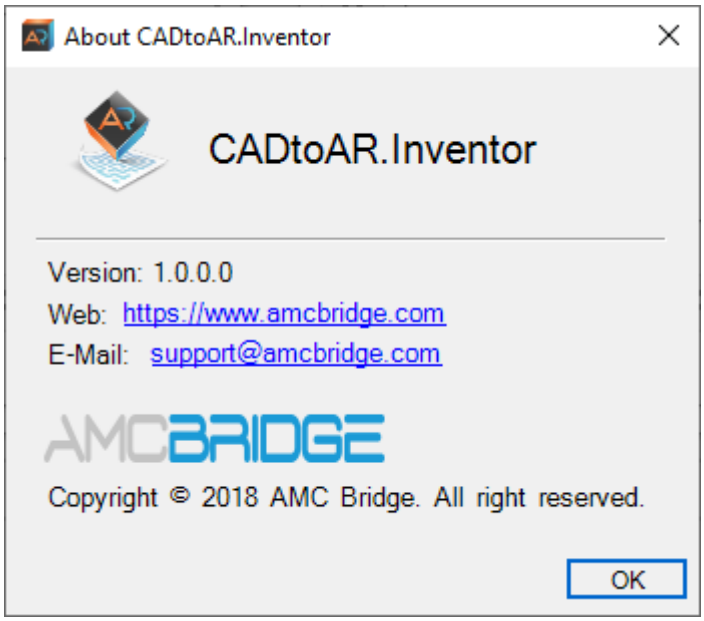

Рис. 3.9. Діалогове вікно “About CADtoAR.Inventor”

В даному вікні відображається інформація про компанію розробила, вказаний плагін посилання на сайт компанії, а також на сервіс технічної підтримки. Також вказується версія розробленого програмного забезпечення. Після натискання на кнопку "OK" діалогове вікно "About CADtoAR.Inventor" - закривається.

Після натискання на кнопку Settings, відкривається модальне діалогове вікно "Settings" (рис. 3.10)

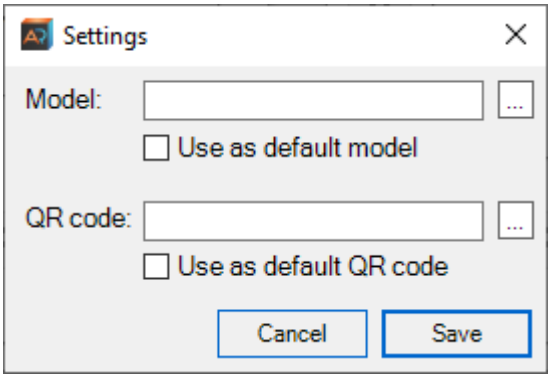

Рис. 3.10. Діалогове вікно “Settings”

В якому присутні два текстових поля, службовців для введення шляху до експортованої моделі і відповідно до QR коду, який необхідно використовувати разом з обраною моделлю.

Праворуч від текстових полів "Model" і "QR Code" присутні кнопки, при натисканні на які відкривається вбудований в ОС Windows додаток "Open File Wizard", що дозволяє вибрати цікаву для користувача моделі, а також QR код, варто відзначити, що в разі, якщо модель раніше використовувалася, то програма автоматично завантажити відповідний QR код.

Також на формі присутні два компоненти типу "CheckBox", якщо знаходиться в стані "Істина", додаток запам'ятовує обрану модель, або QR код, і при наступному відкритті додаток виставляє потрібні шляхи, без втручання оператора.

Кнопка "Cancel" розташована в нижній частині діалогового вікна - закриває форму без збереження результатів. При натисканні на кнопку "Save" дії раніше скоєні користувачем, зберігаються в конфігураційний файл.

Після натискання на кнопку “Generate QR” відкривається модальне діалогове вікно “Generate QR Code” (рис. 3.11)

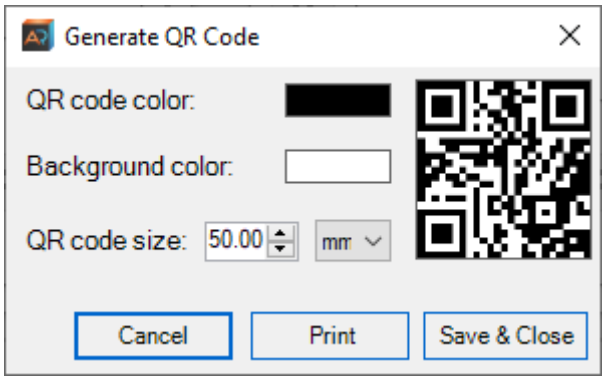

Рис. 3.11. Діалогове вікно “Generate QR Code”

У правій частині форми знаходиться прев’ю генерується QR коду, при зміні значень, представлених в даній формі, зображення QR коду, буде змінюватися в режимі реального часу.

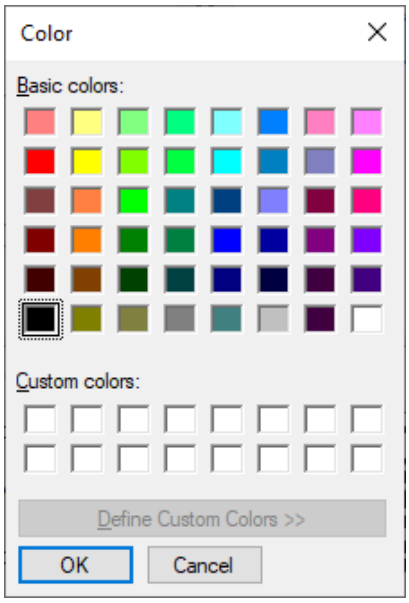

Рис. 3.12. Діалогове вікно вибору кольору

Зліва від прев’ю QR коду, знаходиться кнопки вибору кольору QR коду, при натисканні кнопки відкриється діалогове вікно вибору основного кольору. (рис. 3.12)

Після вибору кольору відмінного від того, що був раніше і натисканні кнопки "OK", основний колір зображення QR коду змінюється на раніше обраний, оновлене вікно "Generate QR Code". (рис. 3.13)

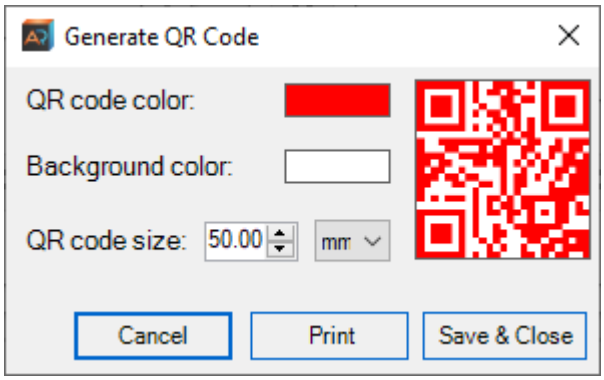

Рис. 3.13. Діалогове вікно “Generate QR Code”

Нижче меню вибору кольору QR коду, знаходяться компоненти, що дозволяють змінювати розмір QR коду, і одиниці вимірювання.

Нижче зображено оновлене вікно "Generate QR Code" (рис. 3.14) зі зміненим розміром QR коду, і одиницями вимірювання.

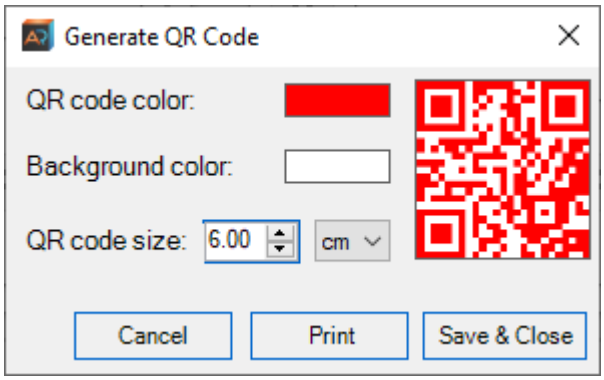

Рис. 3.14. Діалогове вікно “Generate QR Code”

У нижній частині діалогового вікна, розміщені кнопки "Cancel", "Print" і "Save & Close",

При натисканні на кнопку "Cancel" вікно "Generate QR Code" закривається без збереження раніше скоєних дій.

При натисканні на кнопку "Save & Close" буде відкрито вікно майстра збереження (рис), за допомогою, якого користувач зможу зберегти зображення в бажаному місці на жорсткому диску.

Рис. 3.15. Діалогове вікно “Print”

При натисканні на кнопку "Print" відкривається діалогове вікно "Print" (рис. 3.15) дозволяє вказати настройки друку, а також відправити раніше згенерований QR код на друк або ж зберегти его у форматі .pdf.

Після натискання на кнопку "Launch AR" відкривається яке є майстром налаштувань доповненої реальності. Що складається з трьох вкладок і бічній панелі, що дозволяє переміщатися по вкладках, варто відзначити, що перехід на наступну сторінку можливий тільки після того, як дії на поточній – завершені.

Перша вкладка називається "QR Code base point" (рис. 3.15)

На даному етапі налаштувань користувач, повинен поставити крапку на QR коді, який є клікабельним)

У нижній частині форма присутні кнопки "Use defaults", яка виставляє всі налаштування по замовчуванні і завершує процес роботи з з масетером налаштувань. При натисканні на кнопку "Next", відкривається наступна вкладка (рис. 3.16)

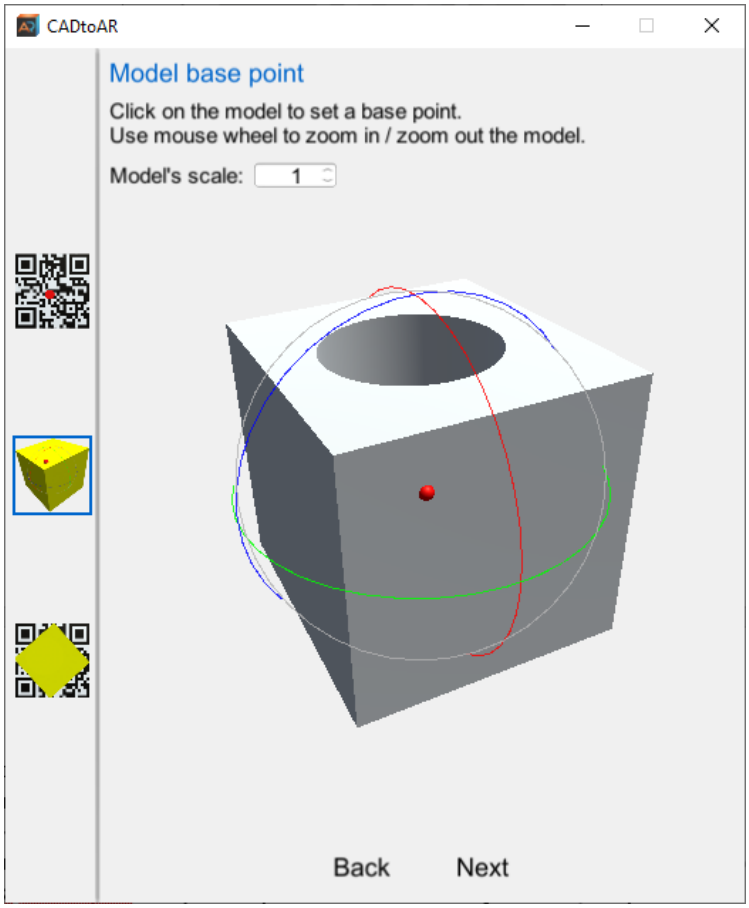

Рис. 3.16. Вкладка вибору базової точки на моделі

На вкладці "Model base point" користувач повинен поставити крапку на моделі, яка є клікабельно.

У верхній частині форми присутній елемент призначеного для користувача інтерфейсу за допомогою якого можна задати коефіцієнт збільшення моделі. Також варто відзначити, що навколо моделі є осі обертання, за допомогою яких можна обертати модель. На даній вкладці також є кнопка "Back", при натисканні на яку майстер налаштувань повертається на попередню вкладку, а також кнопка

"Next", при натисканні на яку відкривається наступна і остання вкладка майстра налаштувань доповненої реальності.

Рис. 3.17. Вкладка попереднього перегляду

На вкладці "Layout Preview" (рис. 3.17), схематично зображується розташування моделі щодо QR коду. У нижній частині вікна присутні дві кнопки "Back" і "Finish".

При натисканні на кнопку "Back", майстер налаштувань повертається на попередню вкладку. При натисканні на кнопку "Finish" відкривається вікно з відеопотоком від камери.

Також варто відзначити, що в разі, коли замість однієї моделі, завантажена збірка моделей, з'являється додатковий пункт меню.

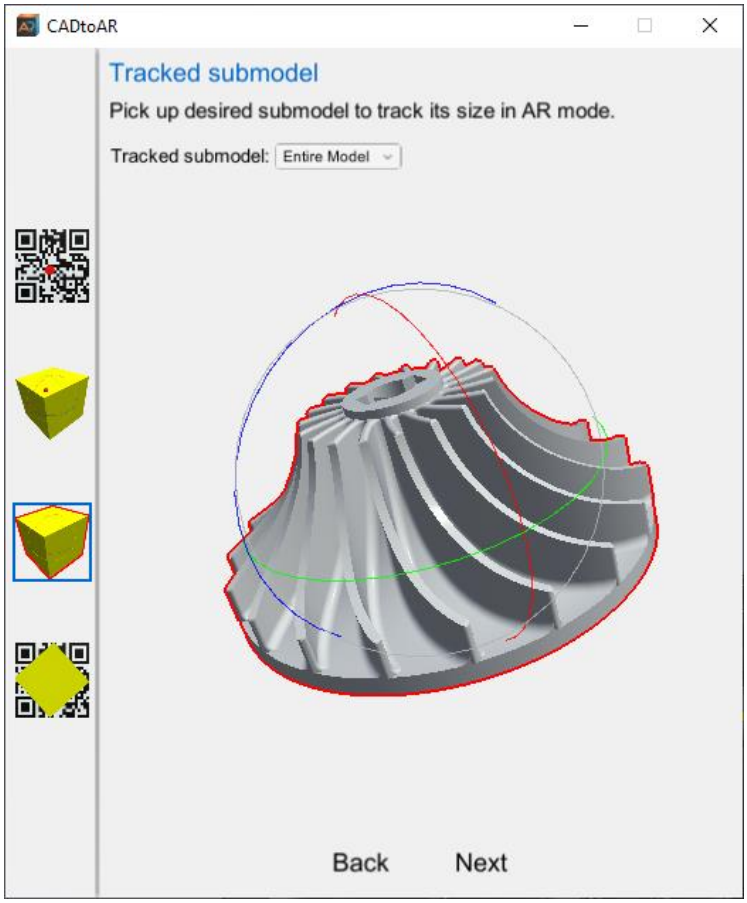

Рис. 3.18. Вкладка “Tracked submodel”

На вкладці "Tracked Submodel" (рис. 3.18) є присутнім компонент, що відображає дочірні елементи, з яких складається вихідна модель (рис. 2.21) Видно, що дана модель складається з двох елементів, які називаються Solid1 і srf8, при виборі будь-якого пункту з цього списку, відповідна йому модель буде виділена червоним.

Після завершення роботи з майстром налаштувань доповненої реальності, в повноекранному режимі відкриється вікно, на яке виводиться відео потік з підключеної камери.

У верхній частині вікна присутній меню з наступними кнопками:

- Кнопка “Move”.
- Кнопка “Rotate”.
- Кнопка “Scale”.
- Кнопка “Undo”.

- Кнопка “Redo”.
- Кнопка “Screen”.
- Кнопка “About”.
- Кнопка “Exit”.

При натисканні на кнопку "Move", близько центра моделі малюються осі трансформації, за допомогою яких можна змінювати координати моделі відносно QR коду.

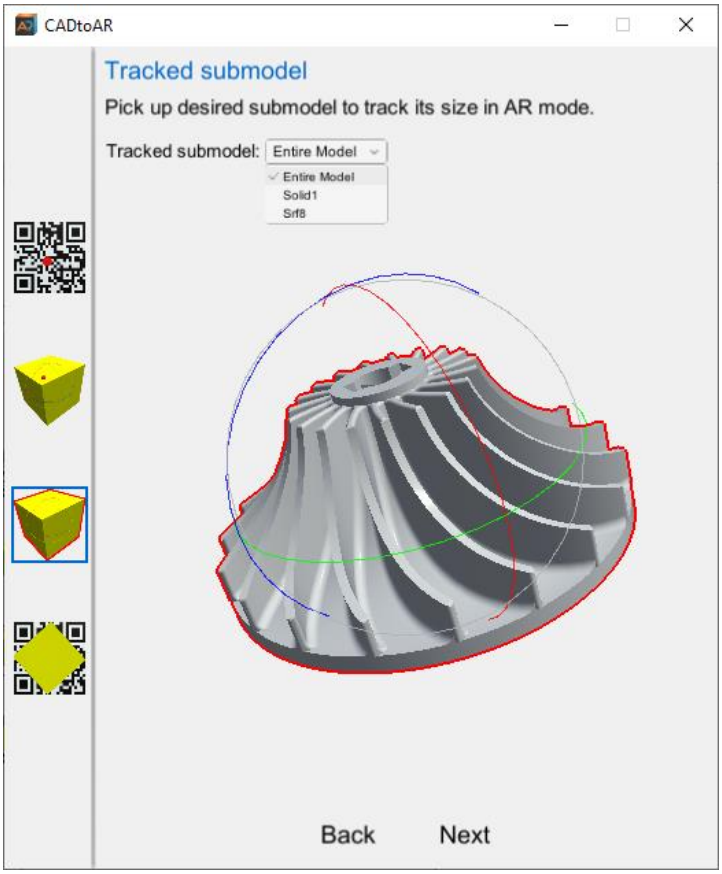

Рис. 3.19. Вкладка “Tracked submodel” з листом моделей

При натисканні на кнопку "Rotate", всередині моделі з'являються осі обертання, за допомогою яких можна обертати модель щодо її фізичного центру.

При натисканні на кнопку "Scale", близько центра моделі з'являються осі збільшення, за допомогою яких можна змінювати фізичний розмір моделі щодо її центру.

При натисканні на кнопку "Undo", будь-яка раніше виконана дія скасовується.

При натисканні на кнопку "Redo", будь-який раннє дію, яку відновлюється.

При натисканні на кнопку "Exit", на екран виводиться діалог з проханням підтвердити операцію. У нижній частині виведеного діалогового вікно знаходяться кнопки "OK" і "Cancel". При натисканні, а кнопку OK додаток припиняється свою роботу.

При натисканні на кнопку "Screenshot", створюється поточний знімок екрана.

При натисканні на кнопку "About", з'являється форма, на якій можна побачити наступне: інформація, про компанію в якій було розроблено програму, вказані гіперпосилання на сайт компанії, а також на сервіс технічної підтримки. Також вказується версія розробленого програмного забезпечення. Після натискання на кнопку "OK" діалогове вікно "About CADtoAR.Inventor" закривається.

У нижній частині вікна відображення доповненої реальності, виводяться дані про розмірності моделі, такі як: висота, довжина і ширина, а також слайдер з допомогою якого можна змінювати прозорість моделі у відсотках.

3.7. Висновки до третього розділу

В рамках даного розділу був розроблений плагін розширює стандартний функціонал САПР Autodesk Inventor. Розроблений плагін надає можливість розглядати створені моделі в режимі доповненої реальності. Можливість розглядати створені моделі в режимі доповненої реальності дозволяє демонструвати розробляється продукт на всіх етапах проєктування без необхідності його виготовлення, тим самим зменшуючи витрати на виготовлення і транспортування вироби. Розроблюється, має простий інтерфейс і інтегрується з Autodesk Inventor, додаючи свої компоненти

користувальницького інтерфейсу на його панель інструментів, тим самим усуваючи необхідність запуску окремого програмного забезпечення. Після здійснити початкову конфігурацію користувач має можливість почати перегляд моделі в режимі AR, який також має свій власний інтерфейс складається з групи кнопок дозволяє виконувати маніпуляцію з демонстрованої моделлю.

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1. Розрахунок трудомісткості та вартості розробки програмного продукту

При розробці програмного забезпечення важливими етапами є визначення трудомісткості розробки ПЗ, розрахунок витрат на створення програмного продукту і аналіз ринку збуту розробленого програмного забезпечення.

Початкові дані:

1) передбачене число операторів – 1125;

2) коефіцієнт складності програми – 1,7;

3) коефіцієнт корекції програми в ході її розробки – 0,6;

4) годинна заробітна плата програміста, грн / год – 65;

5) вартість машино-години ЕОМ, грн / год – 12;

6) коефіцієнт кваліфікації програміста – 0.8;

7) коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання – 1.3.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста, тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_\alpha + t_n + t_{отл} + t_\partial, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі

(приймається 50),

$t_и$ - витрати праці на дослідження алгоритму рішення задачі,

$t_α$ - витрати праці на розробку блок-схеми алгоритму,

$t_п$ - витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ,

$t_д$ - витрати праці на підготовку документації.

Умовне число операторів (підпрограм):

$$Q=q\times C\times(1+p),\ \text{людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми,

p - коефіцієнт кореляції програми в ході її розробки.

Q - передбачуване число операторів (q = 1125).

С коефіцієнт складності програми. Коефіцієнт складності завдання З характеризує відносну складність програми по відношенню до так званої типової задачі, що реалізує стандартні методи рішення, складність якої прийнята рівною одиниці (величина С лежить в межах від 1,25 до 2). Для даного програмного продукту, з урахуванням великої кількості і різноманітності оброблюваної інформації і складності складання звітів, коефіцієнт складності завдання візьмемо 1,7.

Р коефіцієнт корекції програми в ході її розробки. Коефіцієнт корекції програми р - збільшення обсягу робіт за рахунок внесення змін до алгоритму або програму за результатами уточнення постановок. В даному випадку програма вимагала численних доробок. З урахуванням цього візьмемо коефіцієнт рівний 0,6.

Q=1125·1,7·(1+0,6)=3060, людино-годин.

Витрати праці на вивчення опису задачі *tu* визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}, \text{ людино-годин,} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5,

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

В коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання. Коефіцієнт збільшення витрат праці в залежності від складності завдання приймається від 1,25 до 1,5, внаслідок недостатнього опису рішення задачі приймемо В = 1,3.

К коефіцієнт кваліфікації програміста, який визначається від стажу роботи за даною спеціальністю. К = 0,8.

$$t_u = \frac{3060 \cdot 1,3}{80 \cdot 0,8} = 62,15, \text{ людино-годин,}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}, \text{ людино-годин,} \qquad (3.4)$$

$$t_a = \frac{3060}{22 \cdot 0,8} = 173,879, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}, \text{ людино-годин.} \quad (3.5)$$

$$t_n = \frac{3060}{23 \cdot 0{,}8} = 166{,}304, \text{ людино-годин.}$$

Витрати праці на налагодження програми за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}, \text{ людино-годин.} \quad (3.6)$$

$$t_{отл} = \frac{3060}{5 \cdot 0{,}8} = 765, \text{ людино-годин.}$$

Витрати праці на налагодження програми за умови комплексного налагодження завдання:

$$t_{отл}^{\,k} = 0{,}8 \cdot t_{отл}, \text{ людино-годин,} \quad (3.7)$$

$$t_{отл}^{\,k} = 0{,}8 \cdot 765 = 612 \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial р} + t_{\partial о}, \text{ людино-годин} \quad (3.8)$$

де $t_{\partial p}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{\partial p} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \quad (3.9)$$

$$t_{\partial p} = \frac{3060}{18 \cdot 0{,}8} = 212{,}5, \text{ людино-годин.}$$

$t_{\partial o}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{\partial p} = 0{,}75 \cdot t_{\partial p}, \text{ людино-годин.} \quad (3.10)$$

$$t_{\partial p} = 0{,}75 \cdot 212.5 = 159{,}375, \text{ людино-годин.}$$

$$t_{\partial} = 212{,}5 + 159.375 = 371{,}875, \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 62{,}15 + 67{,}879 + 173{,}879 + 173{,}879 + 371{,}875 = 899{,}662, \text{ людино-годин.}$$

4.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП}=t\cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП}=899{,}662\cdot 65=58478{,}03 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ}=t_{отл}\cdot C_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год,

C_M - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ}=765\times 12=9180 \text{ грн.}$$

$$К_{ПО}=58478{,}03+9180=67658{,}03 \text{ грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T=\frac{t}{B_k\cdot F_p}, \text{ міс.} \quad (3.14)$$

де B_k - число виконавців,

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176).

$$T=\frac{899,662}{1\cdot 176}=5,1 \text{ міс.} \quad (3.15)$$

Висновки: визначено трудомісткість розробки плагіна для Autodesk Inventor для конвертації моделі в формат для відображення в режимі доповненої реальності, вона становить 899,662 люд-год, проведений підрахунок вартості роботи по створенню програми 67658,03 грн та розраховано час на його створення 5,1 міс.

4.3. Маркетингові дослідження

Для розрахунку маркетингової частини на даний момент жодна з аналогічних існуючих програм, що використовують технологію доповненої реальності не задовольняє вимогам, тому що їх вихідний код невідомий, а тому складність розробки неможливо порівняти.

4.4. Економічна ефективність

У цьому розділі буде проведено аналіз ефективності впровадження створеної нової системи доповненої реальності на підприємстві.

Впровадження такої системи може дозволити підприємству:

- істотно скоротити витрати на виготовленні готового виробу;
- наочно демонструвати розробляється виріб;

- нівелювати необхідність транспортування готового виробу іншій команді, якщо така є;
- підвищити продуктивність праці;

Тож, можна зробити висновок, що економічний ефект від впровадження ПЗ, яке використовує технологію доповненої реальності, очікується позитивним так як можливість розглядати створені моделі в режимі доповненої реальності дозволяє демонструвати розробляємий продукт на всіх етапах проєктування без необхідності його виготовлення, тим самим зменшуючи витрати.

ВИСНОВКИ

Мета кваліфікаційної роботи є поліпшення методів та алгоритмів, які дозволяють демонструвати тривимірні об'єкти, будь-якої складності, створені в САПР Autodesk Inventor в режимі доповненої реальності

Програма являє собою плагін для системи автоматизованого проєктування Autodesk Inventor, функціонал якого дозволить виконувати валідацію і конвертування файлів моделей Autodesk Inventor в формат який підтримує відображення в режимі Augmented Reality.

Програма може виконувати наступні дії:

- Валідація моделей.
- Конвертація моделей.
- Генерації маркера для відображення моделі.
- Експорт моделі в формат необхідний для відображення в режимі доповненої реальності.

Актуальність даного програмного продукту визначається тим що дане ПЗ допоможе істотно спростити розробку певних деталей, машин або механізмів, над якими працюють кілька команд інженерів які знаходяться в різних місцях, в подібному випадку розробляємо ПО поліпшить комунікації між командами, і нівелює необхідність транспортування частини роботи зробленої однією командою, іншій команді.

Розроблене ПЗ може застосовуватися у багатьох галузях, коли треба уявити, як ще не існуюча річ, буде виглядати у реальному оточенні, але цільовою аудиторією донного ПЗ, є інженери, працюючій у сфері машинобудування.

Структура програми являє собою плагін для САПР Autodesk Inventor, написаний на мові програмування C#, та працюючий за допомогою платформи .Net Framework 4.6.2. Плагін є точкою входу в додаток і після завершення роботи з ним викликається частина додатку яка розроблена за допомогою ігрового

движка Unity, а також фреймворка EasyAR, який спрощує розробку додатків доповненої реальності.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (592 чол-год), підраховані витрати на створення програмного забезпечення (25100 грн.) і гаданий період розробки (3,4 міс.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rahman M. C# Deconstructed / M. Rahman. – Apress, 2014. – 172 p.

2. Ringler R. C# Multithreaded and Parallel Programming / R. Ringler. PACKT Published, 2015. – 344 p..

3. McCool M., Robison A.D., Reinders J. Structured Parallel Programming: Patterns for Efficient Computation. – Morgan Kaufmann, 2012. – 433 p.

4. Harel D., Feldman Y. Algorithmics. The Spirit of Computing / D. Harel, Y.Feldman. – Addison–Wesley, 2004. – 533 pp.

5. Geist G. A., Beguelin A., Dongarra J., Jiang W., Manchek B., Sunderam V. PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Network Parallel Computing. MIT Press, 1994. – 740 p.

6. Gebali F. Algorithms and parallel computing / F. Gebali. Wiley series on parallel and distributed computing. Wiley & Sons, Inc., 2011. – 365 p.

7. Freeman B. NET 4.5 Parallel Extensions Cookbook / B. Freeman. Published by Packt Publishing Ltd., UK. – 320 p.

8. Freeman A. Pro .NET 4 Parallel Programming in C#. / A. Freeman. Apress, New York, 2010. – 329 p.

9. Fox G. C. Solving Problems on Concurrent Processors / G. C. Fox. Prentice Hall, Englewood Cliffs, NJ, 1988. – 416 p.

10. Encyclopedia of Artificial Intelligence. 2nd ed. / [S. C. Shapiro editor]. New York: John Wiley & Sons, 1992. – Volume 1. – p. 434.

11. Duffy J. Concurrent Programming on Windows / J. Duffy. Addison–Wesley, Boston, 2009. – 990 p.

12. Dijkstra E. How do we tell truths that might hurt? / E. Dijkstra // Selected Writings on Computing: A Personal Perspective. – 1982. – № 1(23). – P. 89-131.

13. Cleary S. Concurrency in C# Cookbook / S. Cleary. – O'Reilly Media, 2014.– 208 p.

14. Campbell C., Miller A. Parallel Programming with Microsoft Visual C++ /C. Campbell, A. Miller. – Microsoft Corporation, 2011. – 195 p.

15. Buchanan B. G., Shortliffe E. H. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project / B. G. Buchanan, E. H. Shortliffe. – MA: Addison-Wesley, 1984. – 769 p.

16. Brusilovsky, P. Methods and techniques of adaptive hypermedia / P.Brusilovsky // User Modeling and User-Adapted Interaction. – 1996. – № 6 (2-3). – P. 87-129.

17. Brusilovsky, P. Adaptive and intelligent Web-based educational systems P.Brusilovsky, C. Peylo // International Journal of Artificial Intelligence in Education.Special Issue on Adaptive and Intelligent Web-based Educational Systems – 2003. –№ 13 (2-4). – P. 159-172.

18. Kovacs, James (September 7, 2007). "C#/.NET History Lesson". Retrieved June 18, 2009.

19. Hejlsberg, Anders (October 1, 2008). "The A-Z of Programming Languages: C#". Computerworld.

20. Microsoft C# FAQ". Microsoft. Archived from the originalon February 14, 2006. Retrieved March 25, 2008.

21. Visual C#.net Standard". Microsoft. September 4, 2003. Archived from the original (JPEG) on June 22, 2017. Retrieved June 18, 2009.

22. Piekarski, William; Thomas, Bruce. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer Fifth International Symposium on Wearable Computers (ISWC'01), 2001, pp. 31.

23. Behringer, R.;Improving the Registration Precision by Visual Horizon Silhouette Matching. Rockwell Science Center.

24. Behringer, R.;Tam, C; McGee, J.; Sundareswaran, V.; Vassiliou, Marius. Two Wearable Testbeds for Augmented Reality: itWARNS and WIMMIS. ISWC 2000, Atlanta, 16–17 October 2000.

25. R. Behringer, G. Klinker,. D. Mizell. Augmented Reality – Placing Artificial Objects in Real Scenes. Proceedings of IWAR '98. A.K. Peters, Natick, 1999. ISBN 1-56881-098-9.

26. Felix, Hamza-Lup (30 September 2002). "The ARC Display: An Augmented Reality Visualization Center". CiteSeer. CiteSeerX 10.1.1.89.5595.

27. Wagner, Daniel (29 September 2009). First Steps Towards Handheld Augmented Reality. ACM. ISBN 9780769520346. Retrieved 29 September 2009.

28. Johnson, Joel. "The Master Key": L. Frank Baum envisions augmented reality glasses in 1901 Mote & Beam 10 September 2012.

29. Simoes, Angela (2014-03-27). "New Autodesk 2015 Design Suites Drive Unsurpassed Value for Subscribers" (Press release). San Francisco: Autodesk, Inc. Archived from the original on 2014-03-30. Retrieved 2016-11-11.

30. Simoes, Angela (2015-04-13). "Autodesk 2016 Design & Creation Suites Now Available" (Press release). San Francisco: Autodesk, Inc. Archived from the original on 2015-04-16. Retrieved 2015-04-15.

31. "Autodesk University 2017 Highlights New Tech Cadalyst".

32. StereoLithography Interface Specification, 3D Systems, Inc., July 1988

33. Chua, C. K; Leong, K. F.; Lim, C. S. (2003), Rapid Prototyping: Principles and Applications (2nd ed.), World Scientific Publishing Co, ISBN 981-238-117-1 Chapter 6, Rapid Prototyping Formats. Page 237, "The STL file, as the de facto standard, has been used in many, if not all, rapid prototyping systems." Section 6.2 STL File Problems. Section 6.4 STL File Repair.

34. Burns, Marshall (1993). Automated Fabrication. Prentice Hall. ISBN 978-0-13-119462-5.

35. Hiller, Jonathan D.; Lipson, Hod (2009). "STL 2.0: A Proposal for a Universal Multi-Material Additive Manufacturing File Format" (PDF). Cornell University. Retrieved 5 May 2017.

36. "Export to STL file format - Solid Utopia". solidutopia.com. 11 March 2015.

37. Coppock, Mark (November 30, 2016). "Unity 5.5 can now be used to create HoloLens augmented reality titles". Digital Trends. Archived from the original on February 9, 2019. Retrieved December 3, 2018.

38. Bright, Peter (March 18, 2014). "Unity game engine heading to the browser without plug-ins". Ars Technica. Archived from the original on March 22, 2019. Retrieved October 29, 2018.

39. Orland, Kyle (March 15, 2016). "How new graphics effects can make Unity Engine games look less generic". Ars Technica. Archived from the original on March 20, 2019. Retrieved January 14, 2019.

40. Iman Foroutan; Jack Sklansky (1987). "Feature Selection for Automatic Classification of Non-Gaussian Data". IEEE Transactions on Systems, Man and Cybernetics. 17 (2): 187–198. doi:10.1109/TSMC.1987.4309029.

41. Murray, T. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art / T. Murray. // International Journal of Artificial Intelligence in Education. – 1999.– № 10 – P. 98-129.

42. Padua D. Encyclopedia of Parallel Computing / D. Padua. – Springer, New York, Dordrecht, Heidelberg, London, 2011. – 2196 p.

43. Parhami B. Introduction to Parallel Processing. Algorithms and Architectures / B. Parhami. – Kluwer, 2002. – 557 p.

44. Piekarski, William; Thomas, Bruce. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer Fifth International Symposium on Wearable Computers (ISWC'01), 2001, pp. 31.

45. Pfister G. P. In Search of Clusters / G. P. Pfister. – Prentice Hall PTR, 1998. – 314 p.

46. Rahman M. C# Deconstructed / M. Rahman. – Apress, 2014. – 172 p.

47. Rauber T., Rünger G. Parallel Programming: for Multicore and Cluster Systems / T. Rauber, G. Rünger. – Springer, 2013. – 522 p.

48. Ringler R. C# Multithreaded and Parallel Programming / R. Ringler. – PACKT Published, 2015. – 344 p.

49. Stroustrup B. The C++ Programming Language 4th Edition / B. Stroustrup. – Prentice Hall, 2013. – 1281 p.

50. Wilkinson B., Allen M. Parallel programming / B. Wilkinson, M. Allen. – Prentice Hall, 1999. – 245 p.

51. Албахари Д. C# 5.0. Справочник. Полное описание языка / Д. Албахари. – М.: ООО "И.Д. Вильямс", 2014. – 1008 с.

52. Афанасьев К. Е. Многопроцессорные вычислительные системы и параллельное программирование: Учебное пособие / Афанасьев К.Е., Стуколов С. В., Демидов А. В., Малышенко В. В.; Кемеровский госуниверситет. – Кемерово: Кузбассвузиздат, 2003. – 182 с.

53. Барский А. Б. Параллельные информационные технологии. – Учебное пособие – М.: Интернет–Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. – 503 с.

54. Богачёв К. Ю. Основы параллельного программирования / К. Ю. Богачёв. – Бином. Лаборатория знаний, 2010 г. – 344 с.

55. Бочаров Н. В. Технологии и техника параллельного программирования / Н. В. Бочаров // "Программирование", 2003, № 1. – С. 5–23

56. Буч Г. Объектно–ориентированный анализ и проектирование с примерами приложений / Г. Буч. – 3–е издание, –М.: ООО "И. Д. Вильямс", 2008. – 720 с.

57. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб.: БХВ–Петербург, 2002. – 600 с.

58. Вирт Н. Систематическое программирование / Н. Вирт. – М.: «Мир», 1977. – 184 с.

59. Гергель В. П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем / В. П. Гергель. – Нижний Новгород; Изд–во ННГУ им. Н. И. Лобачевского, 2010. – 421 с.

60. Голдштейн С., Зурбалев Д., Флатов И. и др. Оптимизация приложений на платформе .NET. – М.: ДМК Пресс, 2014. – 522 с.

61. Демьянович Ю. К., Евдокимова Т. О. Теория распараллеливания и синхронизация / Ю. К. Демьянович, Т. О. Евдокимова. – СПб.: Изд–во С. –Пб.ун–та, 2004. – 110 с.

62. Иванников В. П., Ковалевский Н. С., Метельский В. М. О минимальном времени реализации распределенных конкурирующих процессов в синхронных режимах. // Программирование. 2000, № 5. – С. 44–52.

63. Ершов Н. М. Построение графов вычислительных алгоритмов методом автотрассировки / Н. М. Ершов // Программирование. 2000, № 6. – С. 58–64.

64. Керниган Б., Ричи Д. Язык программирования С / Б. Керниган, Д. Ричи. – М: Вильямс, 2006. –304 с.

65. Кнут Д. Искусство программирования для ЭВМ. Сортировка и поиск / Д. Кнут. – М.: Вильямс, 2007. — 824 с.

66. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: Вильямс, 2013. – 1328 с.

67. Крюков В. А., Удовиченко Р. В. Отладка DVM программ / В. А. Крюков, Р. В. Удовиченко // Программирование. – 2001, № 3. – С.19–29.

68. Лаврищева Е. М. Методы программирования: теория, инженерия, практика / Е. М. Лаврищева. – К.: Наукова думка. – 2006. – 451 с.

69. Неупокоев Е.В., Тарнавский Г.А., Вшивков В.А. Распараллеливание алгоритмов прогонки: целевые вычислительные эксперименты. // Автометрия, № 4, т. 38, 2002, С. 74–87.

70. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.0 на языке C#. 3–е изд / Дж. Рихтер. – СПб.: Питер, 2012. – 928 с.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

StandardAddInServer.cs

```
using System;
using System.Runtime.InteropServices;
using InventorAddIn;

namespace InventorPluginUI
{
    /// <summary>
    /// This is the primary AddIn Server class that implements the ApplicationAddInServer interface
    /// that all Inventor AddIns are required to implement. The communication between Inventor and
    /// the AddIn is via the methods on this interface.
    /// </summary>
    [Guid("0f612b87-335c-472a-894c-d08eead1078f")]
    public class StandardAddInServer : Inventor.ApplicationAddInServer
    {
        private PluginCore.PluginCoreManager _coreManager;
        private Inventor.Application _inventorApplication;

        public StandardAddInServer()
        {
        }

        #region ApplicationAddInServer Members

        public void Activate(Inventor.ApplicationAddInSite addInSiteObject, bool firstTime)
        {
            try
            {
                _inventorApplication = addInSiteObject.Application;
                GuidAttribute                                    addInCLSID                                =
(GuidAttribute)Attribute.GetCustomAttribute(typeof(StandardAddInServer), typeof(GuidAttribute));

                InventorCADData inventorData = new InventorCADData(_inventorApplication);
                _coreManager = new PluginCore.PluginCoreManager(
                    new InventorRibbonBuilder(_inventorApplication, addInCLSID.Value),
                    new InventorModelConverter(_inventorApplication, inventorData),
                    inventorData,
                    new
UtilsLib.ConfigManager(UtilsLib.ConfigManager.CallingApplication.InventorAddIn));
                _coreManager.CreateRibbon();
            }
            catch (Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
                Deactivate();
            }
        }

        public void Deactivate()
        {
            Marshal.ReleaseComObject(_inventorApplication);
            _coreManager.Dispose();
            GC.Collect();
            GC.WaitForPendingFinalizers();
        }

        public void ExecuteCommand(int commandID)
        {
            // Note:this method is now obsolete, you should use the
            // ControlDefinition functionality for implementing commands.
        }

        public object Automation
        {
            // This property is provided to allow the AddIn to expose an API
            // of its own to other programs. Typically, this  would be done by
            // implementing the AddIn's API interface in a class and returning
            // that class object through this property.
```

```
            get
            {
                // TODO: Add ApplicationAddInServer.Automation getter implementation
                return null;
            }
        }
        #endregion
    }
}
```

InventorRibbonBuilder.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using System.Drawing;
using Inventor;
using PluginCore;

namespace InventorAddIn
{
    sealed class InventorRibbonBuilder : IRibbonBuilder
    {
        #region Members
        private readonly Inventor.Application _inventorApplication;
        private readonly List<ButtonDefinition> _buttonDefinitions;
        private readonly List<Ribbon> _ribbons;
        private readonly List<RibbonTab> _ribbonTabs;
        private readonly List<RibbonPanel> _ribbonPanels;
        private readonly string _inventorClientID;
        #endregion Members

        #region Constructors
        public InventorRibbonBuilder(Inventor.Application application, string _inventorClientId)
        {
            _inventorApplication = application;

            _ribbons = BuildRibbonArray();
            _ribbonTabs = new List<RibbonTab>();
            _ribbonPanels = new List<RibbonPanel>();

            _inventorClientID = _inventorClientId;
            _buttonDefinitions = new List<ButtonDefinition>();
        }
        #endregion Constructors

        #region Enums
        private enum RibbonNames
        {
            ZeroDoc,
            Part,
            Assembly
        }
        #endregion Enums

        #region Nested classes
        private class AxHostConverter : AxHost
        {
            private AxHostConverter()
                : base("")
            {
            }

            public static stdole.IPictureDisp ImageToPictureDisp(Image image)
            {
                return (stdole.IPictureDisp)GetIPictureDispFromPicture(image);
            }

            public static Image PictureDispToImage(stdole.IPictureDisp pictureDisp)
            {
                return GetPictureFromIPicture(pictureDisp);
            }
        }
        #endregion Nested classes

        #region IRibbonBuilder
        public void AddButtonsToPanel(List<IButton> buttons)
        {
```

```
            foreach (IButton button in buttons)
            {
                _buttonDefinitions.Add(GetButtonDefinition(button));

                foreach (RibbonPanel panel in _ribbonPanels)
                {
                    panel.CommandControls.AddButton(_buttonDefinitions.Last(), true);
                }

                _buttonDefinitions.Last().OnExecute += new ButtonDefinitionSink_OnExecuteEventHandler
                    ((NameValueMap arg) =>
                    {
                        button.OnExecute(new System.IntPtr(_inventorApplication.MainFrameHWND));
                    });
            }
        }

        public void CreateTab(string displayName, string internalName)
        {
            foreach (Ribbon ribbon in _ribbons)
            {
                _ribbonTabs.Add(ribbon.RibbonTabs.Add(displayName,                      internalName,
_inventorClientID));
            }
        }

        public void CreatePanel(string displayName, string internalName)
        {
            foreach (RibbonTab tab in _ribbonTabs)
            {
                _ribbonPanels.Add(tab.RibbonPanels.Add(displayName,                     internalName,
_inventorClientID));
            }
        }
        #endregion IRibbonBuilder

        #region IDisposable
        public void Dispose()
        {
            UnloadRibbon();
        }
        #endregion IDisposable

        #region Methods
        private List<Ribbon> BuildRibbonArray()
        {
            return new List<Ribbon> {
                 _inventorApplication.UserInterfaceManager.Ribbons[RibbonNames.ZeroDoc.ToString()],
                _inventorApplication.UserInterfaceManager.Ribbons[RibbonNames.Part.ToString()],
                _inventorApplication.UserInterfaceManager.Ribbons[RibbonNames.Assembly.ToString()]
            };
        }

        private ButtonDefinition GetButtonDefinition(IButton button)
        {
            stdole.IPictureDisp smallIcon = AxHostConverter.ImageToPictureDisp(button.SmallIcon);
            stdole.IPictureDisp largeIcon = AxHostConverter.ImageToPictureDisp(button.LargeIcon);

            return _inventorApplication.CommandManager.ControlDefinitions.AddButtonDefinition(
                button.DisplayName,
                button.InternalName,
                CommandTypesEnum.kEditMaskCmdType,
                _inventorClientID,
                button.Description,
                button.ToolTipText,
                smallIcon,
                largeIcon,
                ButtonDisplayEnum.kAlwaysDisplayText);
        }

        private void UnloadRibbon()
        {
            RemoveButtonDefinitions();
            RemoveRibbonPanels();
            RemoveRibbonTabs();
        }
```

```
        private void RemoveButtonDefinitions()
        {
            foreach (ButtonDefinition buttonDefinition in _buttonDefinitions)
            {
                buttonDefinition.Delete();
            }
        }

        private void RemoveRibbonPanels()
        {
            foreach (RibbonPanel ribbonPanel in _ribbonPanels)
            {
                ribbonPanel.Delete();
            }
        }

        private void RemoveRibbonTabs()
        {
            foreach (RibbonTab tab in _ribbonTabs)
            {
                tab.Delete();
            }
        }
        #endregion Methods
    }
}
```

InventorCADData.cs

```
using UtilsLib;
using Inventor;
using System.Windows.Forms;
namespace InventorAddIn
{
    class InventorCADData : PluginCore.ICADData
    {
        #region Members
        private readonly Inventor.Application _inventorApplication;
        private readonly string[] _extensions = { ".ipt", ".iam" };
        private const string PropertyName = "QRCodePath";
        private DocumentWrapper _documentWrapper;
        #endregion Members

        #region Constructors
        public InventorCADData(Inventor.Application inventorApplication)
        {
            _inventorApplication = inventorApplication;
        }
        #endregion Constructors

        #region nested classes
        private class DocumentWrapper
        {
            private readonly Inventor.Application _inventorApplication;

            public DocumentWrapper(Inventor.Application inventorApplication, string fullDocumentName)
            {
                _inventorApplication = inventorApplication;

                if (!IsInventorDocumentOpened(inventorApplication, fullDocumentName))
                {
                    const bool shouldOpenDocumentVisible = false;
                    inventorApplication.Documents.Open(fullDocumentName, shouldOpenDocumentVisible);
                }
                InventorDocument = inventorApplication.Documents.ItemByName[fullDocumentName];
            }

            private bool IsInventorDocumentOpened(Inventor.Application inventorApplication, string
fullDocumentName)
            {
                foreach (Document inventorDocument in inventorApplication.Documents)
                {
                    if (inventorDocument.FullDocumentName == fullDocumentName)
                    {
                        return true;
                    }
                }
                return false;
```

```
            }

            public Document InventorDocument { private set; get; }

            public void Close()
            {
                if (!IsCurrentDocumentActive(InventorDocument.FullDocumentName))
                {
                    InventorDocument.Close();
                }
            }

            private bool IsCurrentDocumentActive(string documentName)
            {
                if (_inventorApplication.ActiveDocument == null)
                {
                    return false;
                }

                return _inventorApplication.ActiveDocument.FullDocumentName == documentName;
            }
        }
        #endregion nested classes

        #region Propertires
        public string DocumentFullName { set; get; }
        public string QRCodeImageReference { set; get; }
        public string[] CADSystemExtension { get => _extensions; }

        public Document InventorDocument { private set; get; }
        #endregion Properties

        #region ICADData interface methods

        public bool HasActiveDocument()
        {
            return _inventorApplication.ActiveDocument != null;
        }

        public UtilsLib.MeasurementUnits GetCurrentModelMeasurementUnits()
        {
            return
InventorUnitsConvertor(_inventorApplication.ActiveDocument.UnitsOfMeasure.LengthUnits);
        }

        private MeasurementUnits InventorUnitsConvertor(Inventor.UnitsTypeEnum unitsType)
        {
            switch (unitsType)
            {
                case Inventor.UnitsTypeEnum.kMillimeterLengthUnits:
                    return MeasurementUnits.mm;

                case Inventor.UnitsTypeEnum.kCentimeterLengthUnits:
                    return MeasurementUnits.cm;

                case Inventor.UnitsTypeEnum.kMeterLengthUnits:
                    return MeasurementUnits.m;

                case Inventor.UnitsTypeEnum.kInchLengthUnits:
                    return MeasurementUnits.inch;

                default:
                    return MeasurementUnits.UnsupportedUnits;
            }
        }

        public bool LinkQRCodeWithActiveModel(string qrCodePath)
        {
            Document _inventorDocument = _inventorApplication.ActiveDocument;
            if (_inventorDocument != null)
            {
                LinkQRCode(qrCodePath, _inventorDocument);
                return true;
            }
            return false;
        }
```

```
        public void LinkQRCodeWithModel(string qrCodePath, string documentFullName)
        {
            if (_documentWrapper != null)
            {
                _documentWrapper.Close();
            }
            _documentWrapper = new DocumentWrapper(_inventorApplication, documentFullName);
            InventorDocument = _documentWrapper.InventorDocument;
            DocumentFullName = _documentWrapper.InventorDocument.FullDocumentName;
            LinkQRCode(qrCodePath, _documentWrapper.InventorDocument);
        }

        public string GetQRCodePathFromModel(string documentFullName)
        {
            if (_documentWrapper != null)
            {
                _documentWrapper.Close();
            }
            _documentWrapper = new DocumentWrapper(_inventorApplication, documentFullName);
            DocumentFullName = _documentWrapper.InventorDocument.FullDocumentName;
            InventorDocument = _documentWrapper.InventorDocument;
            return GetQRCodePath(_documentWrapper.InventorDocument);
        }

        public string GetQRCodePathFromModel()
        {
            return GetQRCodePath(_inventorApplication.ActiveDocument);
        }
        #endregion ICADData interface methods

        #region Methods
        private string GetQRCodePath(Document inventorDocument)
        {
            PropertySet propertySetTyped = null;
            if
(inventorDocument.PropertySets.PropertySetExists(ApplicationConstants.ApplicationName,   out   object
propertySet))
            {
                propertySetTyped = (PropertySet)propertySet;
                return propertySetTyped[PropertyName].Value;
            }

            return "";
        }

        private void LinkQRCode(string qrCodePath, Document inventorDocument)
        {
            bool                                isPropertySetExist                                  =
inventorDocument.PropertySets.PropertySetExists(ApplicationConstants.ApplicationName,   out   object
propertySet);
            PropertySet propertySetTyped = (PropertySet)propertySet;
            if (!isPropertySetExist)
            {
                propertySetTyped                                                                    =
inventorDocument.PropertySets.Add(ApplicationConstants.ApplicationName);
                propertySetTyped.Add(qrCodePath, PropertyName, propertySetTyped.Count);
                QRCodeImageReference = qrCodePath;
            }
            else
            {
                DialogResult dialogResult = MessageBox.Show("The current model already has a QR Code
attached.\n" +
                "Are    you    sure    you    want    to    change    the    linked    QR    Code    ?",
ApplicationConstants.ApplicationName, MessageBoxButtons.YesNo);
                if (dialogResult == DialogResult.Yes)
                {
                    Property property = propertySetTyped[PropertyName];
                    property.Value = qrCodePath;
                    QRCodeImageReference = qrCodePath;
                }
            }
            inventorDocument.Save();
        }

        public string GetActiveModelQrLink()
        {
            object propertySet = null;
```

```
            if
(_inventorApplication.ActiveDocument?.PropertySets?.PropertySetExists(ApplicationConstants.Applicati
onName, out propertySet) == true)
            {
                PropertySet propertySetTyped = (PropertySet)propertySet;
                Property property = propertySetTyped[PropertyName];
                return (string)property.Value;
            }

            return string.Empty;
        }

        public string GetActiveDocumentFullName()
        {
            return _inventorApplication.ActiveDocument?.FullDocumentName;
        }

        public void OpenDocument(string documentFullName,bool isVisible)
        {
            InventorDocument = _inventorApplication.Documents.Open(documentFullName, isVisible);
        }
        #endregion Methods
    }
}
```

InventorOBJConveter.cs

```
namespace InventorAddIn
{
    class InventorOBJConverter
    {
        private readonly Inventor.Application _inventorApplication;

        #region Enums
        private enum ModelQuality
        {
            Ultra = 0,
            High = 1,
            Middle = 2,
            Low = 3
        };

        // This is enum is used as allies for Exporter AddIn' Units of measure values
        private enum ExportUnits
        {
            Inch = 2,
            Foot = 3,
            Centimeter = 4,
            Millimeter = 5,
            Meter = 6,
            Micron = 7
        };
        #endregion Enums

        #region Methods
        public InventorOBJConverter(Inventor.Application inventorApplcation)
        {
            _inventorApplication = inventorApplcation;
        }

        public void ExportDocument(Inventor.Document document, string compatibleName)
        {
            SaveDocument(document, compatibleName);
        }

        private void SaveDocument(Inventor.Document document, string compatibleNameWithPath)
        {
            const string ObjTranslatorGuid = "{F539FB09-FC01-4260-A429-1818B14D6BAC}";
            Inventor.TranslatorAddIn objTranslator =

(Inventor.TranslatorAddIn)_inventorApplication.ApplicationAddIns.ItemById[ObjTranslatorGuid];

            if (objTranslator == null)
            {
                throw new System.Exception("Unable to get OBJ translator addin.\nProbably you are
using old version of Inventor.");
            }
```

```
            Inventor.TranslationContext                          context                          =
_inventorApplication.TransientObjects.CreateTranslationContext();
            context.Type = Inventor.IOMechanismEnum.kFileBrowseIOMechanism;
            Inventor.NameValueMap                                options                          =
_inventorApplication.TransientObjects.CreateNameValueMap();
            Inventor.DataMedium data = _inventorApplication.TransientObjects.CreateDataMedium();

            if (objTranslator.HasSaveCopyAsOptions[document, context, options])
            {
                options.Value["Resolution"] = ModelQuality.Ultra;
            }

            data.FileName = compatibleNameWithPath;
            objTranslator.SaveCopyAs(document, context, options, data);
        }
        #endregion Methods
    }
}
```

InventorModelTransformer.cs

```
using System.Text.RegularExpressions;
using System.IO;

namespace InventorAddIn
{
    class InventorModelTransformer
    {
        #region Members
        private readonly Inventor.Application _inventorApplication;
        #endregion  Members

        #region Constructors
        public InventorModelTransformer(Inventor.Application inventorApplication)
        {
            _inventorApplication = inventorApplication;
        }
        #endregion Constructors

        #region Methods
        public          string          TransformModel(Inventor.Document          inventorDocument,
PluginCore.IARCompatibilityChecker compatiblityChecker)
        {
            compatiblityChecker.IsDocumentARCompatible();

            string documentName = inventorDocument.DisplayName;
            if (!compatiblityChecker.IsDocumentNameARCompatible(documentName))
            {
                documentName = ConvertName(documentName);

                if (!compatiblityChecker.IsDocumentNameARCompatible(documentName))
                {
                    throw new System.Exception("Document name is not AR compatible!");
                }
            }

            return AppendOBJExtension(documentName);
        }

        private string ConvertName(string name)
        {
            const string FindSpacesRegex = @"\s+";
            string spacelessName = Regex.Replace(name, FindSpacesRegex, "");
            return spacelessName;
        }

        private string AppendOBJExtension(string name)
        {
            const string Extension = ".obj";
            string TempFolder = Path.GetTempPath();
            string FileName = Path.GetFileNameWithoutExtension(name) + Extension;

            return TempFolder + FileName;
        }
        #endregion Methods
    }
}
```

InventorModelConverter.cs

```
using Inventor;
using PluginCore;

namespace InventorAddIn
{
    class InventorModelConverter : IARModelConverter
    {
        #region Members
        private readonly Application _inventorApplication;
        private readonly InventorCADData _inventorCADData;
        #endregion Members

        #region Constructors
        public InventorModelConverter(Application inventorApplication, InventorCADData _cadData)
        {
            _inventorCADData = _cadData;
            _inventorApplication = inventorApplication;
            ApplicationEvents inventorEvents = _inventorApplication.ApplicationEvents;
            inventorEvents.OnOpenDocument                              +=                              new
ApplicationEventsSink_OnOpenDocumentEventHandler(OnOpenDocumentEventHandler);
        }
        #endregion Constructors

        #region IARModelConverter interface methods
        public string SaveToARCompatibleOBJ(string documentName)
        {
            Document exportedDocument = _inventorApplication.ActiveDocument;
            if (!IsActiveDocumentExported(documentName))
            {
                exportedDocument = _inventorCADData.InventorDocument;
            }

            InventorModelTransformer              modelTransformer              =              new
InventorModelTransformer(_inventorApplication);
            string arCompatibleModelFullName = modelTransformer.TransformModel(exportedDocument, new
InventorARCompatibilityChecker(exportedDocument));
            new          InventorOBJConverter(_inventorApplication).ExportDocument(exportedDocument,
arCompatibleModelFullName);

            return arCompatibleModelFullName;
        }
        #endregion IARModelConverter interface methods

        #region Methods
        private void OnOpenDocumentEventHandler(
           _Document documentObject,
           string fullDocumentName,
           EventTimingEnum processAfterModelOpening,
           NameValueMap context,
           out HandlingCodeEnum handledCode)
        {
            _inventorCADData.DocumentFullName                                                          =
_inventorApplication.ActiveDocument.FullDocumentName;
            _inventorCADData.QRCodeImageReference = _inventorCADData.GetQRCodePathFromModel();

            processAfterModelOpening = EventTimingEnum.kAfter;
            handledCode = HandlingCodeEnum.kEventHandled;
        }

        private bool IsActiveDocumentExported(string documentName)
        {
            if (_inventorApplication.ActiveDocument == null)
            {
                return false;
            }

            return _inventorApplication.ActiveDocument.FullDocumentName == documentName;
        }
        #endregion Methods
    }
}
```

IARCompatibilityChecker.cs

```
namespace PluginCore
{
    public interface IARCompatibilityChecker
    {
```

```
        #region Mehtods
        void IsDocumentARCompatible();
        bool IsDocumentNameARCompatible(string name);
        #endregion Mehtods
    }
}
```

IButton.cs

```
using System.Drawing;

namespace PluginCore
{
    public interface IButton
    {
        #region Properties
        string InternalName { get; }
        string DisplayName { get; }
        string Description { get; }
        string ToolTipText { get; }
        Bitmap SmallIcon { get; }
        Bitmap LargeIcon { get; }
        #endregion Properties

        #region Methods
        void OnExecute(System.IntPtr winHandler);
        #endregion Methods
    }
}
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу магістра

на тему: «Методи, алгоритми і програмне забезпечення для розширення функцій САПР моделей

реальних об’єктів в режимі доповненої реальності.»

студента групи 121м-19-1 Циганок Євгена Дмитровича

Керівник економічного розділу

доцент каф. ПЕП та ПУ, к.е.н. Л. В. Касьяненко

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ДИСКУ

Ім’я файла	Опис
Пояснювальні документи	
Master-thesis-ytsyhanok.docx	Пояснювальна записка до дипломного проєкту. Документ Word.
Master-thesis-ytsyhanok.pdf	Пояснювальна записка до дипломного проєкту в форматі PDF
Програма	
InventorPlugin.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Master-thesis-tsyhanok.pptx	Презентація дипломного проєкту