МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розроблення математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских об'єктів____

Рівень вищої освіти другий (магістерський)_____

Спеціальність 122 Комп'ютерні науки_____

Освітня програма Комп'ютерні науки_____

Виконав: студент групи МгІТ-2-22

_____ **Максиміліан МІНЕНКО**

Науковий керівник: д.т.н., проф. **Віктор ЧУПРИНКА**

Рецензент д.ф.-м.н.,проф. **Сергій КРАСНИТСЬКИЙ**

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____Володимир ЩЕРБАНЬ

«___»_____2023___року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Міненку Максиміліану Станіславовичу

1.Тема роботи: Розроблення математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских об'єктів.

Науковий керівник роботи д.т.н., професор Чупринка Віктор Іванович, затверджені наказом закладу вищої освіти від 12 . 09.2023 року , № 210-уч.

2. Вихідні дані до кваліфікаційної роботи: Розробка кафедри комп'ютерних наук

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

Вступ, розділ 1(САПР технологічних процесів); розділ 2 (Математичне забезпечення для інтерактивної побудови щільних укладок плоских об'єктів); розділ 3(Програмне забезпечення побудови укладок плоских об'єктів) Додатки – програмні коди модулів системи.

4. Дата видачі завдання 08.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	30.08.2023	
2	Розділ 1 САПР технологічних процесів	06.09.2023	
3	Розділ 2. Математичне забезпечення для інтерактивної побудови щільних укладок плоских об’єктів	28.09.2023	
4	Розділ 3. Програмне забезпечення побудови укладок плоских об’єктів	21.10.2023	
5	Висновки	29.10.2023	
6	Оформленнякваліфікаційної роботи (чистовий варіант)	06.11.2023	
7	Подача кваліфікаційної роботи (проєкту) науковому керівнику для відгуку (за 14 днів дозахисту)		
8	Подача кваліфікаційної роботи (проєкту) для рецензування		
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату		
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри		

З завданням ознайомлений:

Студент _____ Максиміліан МІНЕНКО
(підпис) (ім’я та прізвище)

Науковий керівник роботи _____ Віктор ЧУПРИНКА
(підпис) (ім’я та прізвище)

АНОТАЦІЯ

Міненко М.С. Розроблення математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских об’єктів – Рукопис.

Дипломна магістерська робота за спеціальністю 122- «Комп’ютерні науки» – Київський національний університет технологій та дизайну, Київ, 2023 рік.

В роботі запропонований методи та алгоритми для інтерактивної побудови щільних укладок плоских об’єктів. Запропоновані алгоритми реалізовані в програмний продукт для інтерактивної побудови щільних укладок плоских об’єктів.

Розроблений програмний продукт має дружній інтерфейс та не потребує спеціальних знань з комп’ютерної техніки для роботи з ним.

Ключові слова: математичне та програмне забезпечення, щільні укладки, плоскі об’єкти

ABSTRACT

Minenko M.S. Development of mathematical and software for interactive construction of dense stacks of flat objects - Manuscript.

Master's thesis in specialty 122- "Computer Science" - Kyiv National University of Technology and Design, Kyiv, 2023.

The paper proposes methods and algorithms for interactive construction of dense stacks of flat objects. The proposed algorithms are implemented in a software product for interactive construction of dense stacks of flat objects.

The developed software product has a friendly interface and does not require special knowledge of computer technology to work with it.

Keywords: mathematical and software, dense stacking, flat objects

ЗМІСТ

ВСТУП

На ціну взуттєвих виробів впливає їх значна матеріалоємність та висока вартість сировини. Тому задачу зменшення витрат матеріалів ставлять на одне із перших місць.

Автоматизоване проектування схем розкрою у взуттєвій галузі дозволить раціонально використовувати рулонні матеріали при розкрої на деталі, зменшити кількість відходів, які забруднюють навколишнє середовище, знизити собівартість виробів.

Крім того, використання у виробничому процесі обчислювальної техніки, автоматизованого обладнання, створює можливість застосування автоматизованих комплексів в розкрійному виробництві.

А це підвищить якість виробів, зменшить часові витрати, оптимізувати працю робітників та покращить раціональність використання матеріалу при розкрої. Враховуючи те, що матеріали у собівартості готового взуття складають значну частину, то необхідно використовувати всі можливості підвищення проценту використання рулонних матеріалів при розкрої на деталі взуття.

Інтерактивна побудова щільних укладок деталей взуття дозволяє визначити економічність моделі, так як вона дозволяє оцінити міжлекальні випади при розкрої матеріалу на деталі взуття. Звідси очевидна актуальність даної роботи.

Методи дослідження. Теоретичні дослідження роботи грунтуються на комплексі основних положень виробництва взуття та застосування комп'ютерних наук при вирішенні поставленої задачі.

Експерементальні дослідження заключалися в тестуванні розробленого продукту для проектування декоративних елементів на деталях взуття.

Наукова новизна полягає у розробці математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских геометричних обєктів.

. Апробація результатів магістерської роботи. Основні положення і результати магістерської роботи протягом 2023 року були представлені та одержали позитивну оцінку на міжнародній науковій конференції:

Публікації. За темою магістерської роботи «Розробка математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских об’єктів» опубліковано одна наукових робота.

1. САПР ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

1.1. CAE/CAD/CAM-системи

У наш час діяльність проєктних організацій не можна уявити без комп'ютеризації, що забезпечує проєктним роботам якісно новий рівень, обґрунтоване розв'язання багатьох складних інженерних завдань, а також підвищує темпи проєктування. Така реалізація стала можливою завдяки використанню ефективних спеціалізованих програм. Діяльність зі створення програмних продуктів і технічних засобів для автоматизації проєктних робіт має загальну назву – САПР.

До систем автоматизованого проєктування (САПР) зараховують зазвичай CAE/CAD/CAM-системи, які призначені для забезпечення спільного функціонування компонентів САПР різного призначення, тобто координації роботи CAE/CAD/CAM-систем.

Для управління проєктними даними та проєктуванням розробляють системи, що отримали назву PDM – Product Data Management (системи управління проєктними даними).

Сучасні системи автоматизованого проєктування зазвичай використовують спільно із системами автоматизації інженерних розрахунків й аналізу CAE.

CAE-системи (computer-aided engineering – підтримка інженерних розрахунків) – це великий клас систем, кожна з яких дає змогу розв'язувати певну розрахункову задачу, починаючи від розрахунків на міцність, аналізу та моделювання теплових процесів до розрахунків гідравлічних систем та ін.

У CAE-системах також використовується тривимірна модель виробу, що створюється в CAD-системі. CAE-системи ще називають системами інженерного аналізу. Термін «САПР» містить як CAD, так і елементи CAM (Computer-aided manufacturing), а іноді й елементи CAE (Computer-aided engineering). Більшість термінів, що використовується в галузі автоматизації проєктування, має сталі

англійські абревіатури, які останнім часом широко застосовуються в українськомовних технічних текстах, присвячених САПР.

За галузями застосування САПР традиційно розділяються на (рис. 1):

– архітектурно-будівельні (AEC CAD);

– механічні (MCAD);

– електронні (ECAD);

– технологічні (CAPP). CAD CAM CAE PDM

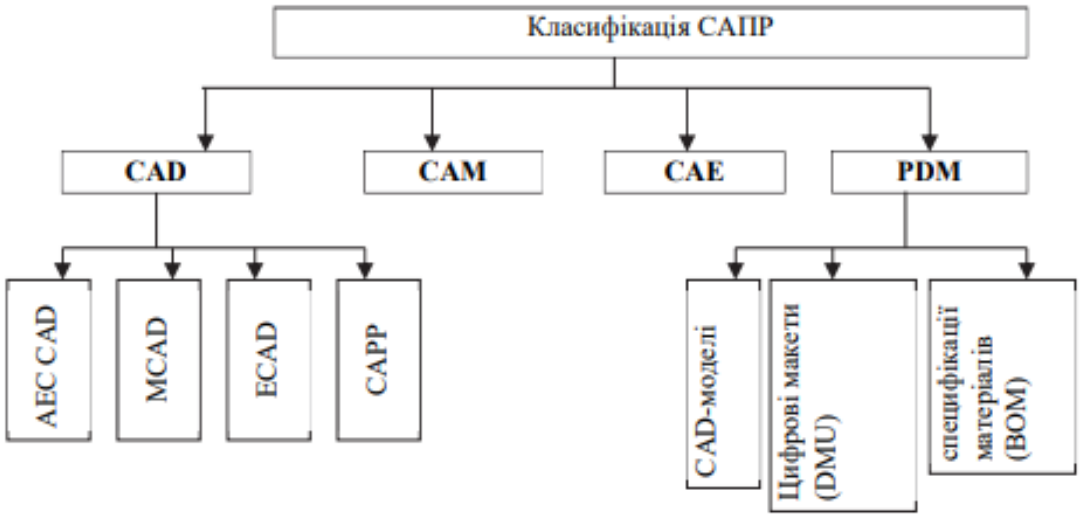

Рис. 1. 1. Класифікація САПР

Система автоматизації проєктних робіт (САПР), або CAD (англ. Computer-Aided Design), – організаційно-технічна система, що призначена для підготовки виробництва та конструювання, управління інженерними даними, виконання проєктної діяльності із застосуванням обчислювальної техніки, а також яка дає змогу створювати конструкторську та/або технологічну документацію (CAD/CAM/CAE/PDM).

CAD-системи (computer-aided design – комп’ютерна підтримка проєктування) призначені для розв’язання конструкторських задач й оформлення конструкторської документації.

Сучасні CAD-системи містять модулі:

– створення тривимірної об’ємної конструкції (деталі);

– тривимірного параметричного моделювання поверхонь й об’ємних тіл;

– створення геометричних моделей виробу (твердотільних, тривимірних, складених);

– генерацію креслень виробу;

– оформлення креслень;

– розробку документації.

Сучасні тривимірні CAD-системи дають змогу реалізувати майже повний життєвий цикл виробів. Серед систем, притаманних САПР, можна відзначити такі: конструкторські, технологічні, архітектурно-будівельні, проєктування теплопостачання, електротехнічні системи автоматизації, а також модулі для створення специфікацій.

Робота із САПР полягає у створенні геометричної моделі виробу (двовимірної чи тривимірної, твердотілої), генерації на основі цієї моделі конструкторської документації (креслень виробу, специфікацій тощо) і його супровід.

CAM-системи (computer-aided manufacturing – комп’ютерна підтримка виготовлення), призначені для проєктування обробки виробів на верстатах (створення програм для управління верстатами – фрезерних, свердлильних, ерозійних, пробивних, токарних, шліфувальних й ін.) з числовим програмним керуванням (ЧПК), забезпечують автоматизацію програмування та керування обладнанням з ЧПК.

Вхідними даними CAM-системи є геометрична модель виробу, розроблена в системі автоматизованого проєктування CAD. У процесі інтерактивної роботи з тривимірною моделлю в CAM-системі інженер визначає траєкторії руху різального інструмента по заготівці виробу (так звані CL-дані, від cutter location – положення різця), які потім автоматично верифікуються, візуалізуються (для візуальної перевірки коректності) й обробляються постпроцесором для отримання програми управління (G-коду) конкретним верстатом. CAM-системи

ще називають системами технологічної підготовки виробництва. З моменту появи перших верстатів з ЧПК до впровадження нових обробних центрів з'явилися різні мови програмування обробки. Сьогодні програмування в G- і M-кодах є найбільш популярним, яке прийнято за стандарт. Мова G- та M-кодів ґрунтується на положеннях Міжнародної організації зі стандартизації (ISO) та Асоціації електронної промисловості (EIA) (Найкраще програмне забезпечення, 2022).

PDM (product data management) – управління даними про виріб. Призначена безпосередньо для розв'язання завдань, пов'язаних з розробкою конкретного проєкту, вузла, агрегату. Категорія програмного забезпечення, що дає змогу зберігати дані про виріб у базах даних. До даних про виріб зараховують інженерні дані, такі як:

– CAD-моделі та креслення;

– цифрові макети (DMU);

– специфікації матеріалів (BOM);

– метадані, що містять інформацію про розробника файлу й поточний статус відповідної компоненти.

Система PDM дає змогу:

– організувати сумісний доступ до цих даних, забезпечуючи їх постійну цілісність, керування зберіганням даних і документів;

– організувати керування процесами;

– організувати керування структурою виробу;

– організувати класифікацію;

– організувати календарне планування;

– забезпечувати внесення необхідних змін до всіх версій виробу;

– модифікувати специфікацію матеріалів;

– допомагати конфігурувати варіанти виробу, допоміжні функції.

Проте найважливішою перевагою системи PDM є її використання впродовж всього життєвого циклу виробу в межах концепції управління цим циклом. Більшість PDM-систем дає змогу одночасно працювати з інженерними даними, отриманими від різних CAD-систем.

1.2. Роль і місце САПР ТП в технологічній підготовці виробництва

Основна мета розробки САПР ТП - це отримання в найкоротші терміни максимального прибутку з мінімальними витратами виготовлення виробу, і мінімальною кількістю можливих помилок. Для досягнення цих цілей необхідно мати у своєму розпорядженні засоби автоматизації оформлення технологічної документації, засоби інформаційної підтримки проектування та автоматизації прийняття рішень.

З розвитком САПР ТП арсенал таких засобів поступово розширювався. На першому етапі ці системи часто були спеціалізованими текстовими редакторами, деякі з яких були документованими.

З появою баз даних з'явилася можливість пошуку БД необхідних засобів технологічного оснащення. Однак переважна більшість САПР ТП, у тому числі й нині існуючих, не здатна підтримувати автоматизацію прийняття рішень у процесі проектування на основі технологічних знань. Відмінна особливість САПР ТП полягає у необхідності їх налаштування при впровадженні у різних виробничих умовах. Змінам підлягають насамперед склад і структура баз даних, процедури прийняття технологічних рішень, форми вихідної документації.

Вирішальними факторами життєздатності САПР ТП є гнучкість та переналаштовуваність у процесі впровадження та експлуатації. Центральне місце у САПР ТП займає модель технологічного процесу. Решта баз даних системи є джерелом інформації для цієї моделі.

Кінцевою метою САПР ТП є розробка комплекту технологічної документації. До складу САПР ТП, як правило, закладається принцип групової технології, згідно з яким усі деталі класифікуються та об'єднуються у групи залежно від способу їх виготовлення. Для кожної групи деталей визначається свій алгоритм розробки процесу, причому методика проектування може бути реалізована двома способами.

Перший – це безмашинне проектування, що базується на автоматизації розрахунків. Такий підхід має механістичний характер, і ефективний у разі проектування добре формалізованих процесів.

В основі другого способу закладено системний підхід до технології, що розробляється, приймаються до уваги взаємозв'язок і закономірності всіх явищ, що супроводжують процес, що вивчається.

При використанні автоматизованих технологічних процесів вони розміщуються в комп'ютерній базі даних, а необхідна документація стає відображенням внутрішнього подання технологічного процесу у зовнішню сферу. Технологічні процеси, що зберігаються в базі даних, є основним джерелом інформації для вирішення завдань автоматизованого управління технологічною підготовкою виробництва.

1.3. Огляд та аналіз автоматизованих САПР ТП у різних галузях промисловості

За даними авторитетних організацій щорічно у світі продається близько 500 тис. ліцензій на САПР для різних галузей економіки. Це цілком зрозуміло, оскільки США є світовим лідером з випуску САПР, що пов'язано, перш за все, з історичним фактом виникнення та розвитку в Новому світлі так званої силіконової долини.

У процесі аналітичної роботи з'ясувалося, що «левова частка» з усієї якості САПР припадає на системи CAD/CAM для конструкторської підготовки

виробництва з виходом на технологічне обладнання, майже третину від них складають системи CAD/CAЄ для виконання та аналізу розрахунків, і лише сьома частина всіх впроваджених у вітчизняне виробництво систем належить системам автоматизованої технологічної підготовки виробництва.

Найбільшу питому вагу мають багатогалузеві або універсальні програми, такі, як «LSM OPTIMUS» (Бельгія), «Lotsia ERP», «ТехноПро», «T-FLEX», програми фірм «СПРУТ-Технологія» та «Сударушка» (Росія), а також програми для дизайну та конструювання виробів різного призначення фірм Австралії, Англії, Німеччини, Ізраїлю, Росії, США та інші.

Найбільш поширеними САПР ТП є: "Вертикаль", "Techcard", "ТехноПро", "TechnologiCS", "T-FLEX Технологія". Дані системи є універсальними системами автоматизованого проектування технологічної підготовки, придатними для проектування техпроцесів практично будь-якого виробництва, зокрема САПР «КомпасАВТОПРОЕКТ» та «T-FLEX/ТехноПро», є невід'ємними частинами комплексних САПР, призначених для всього циклу проектування виробів.

САПР ТП «NATTA» є підсистемою інтегрованої системи технічної підготовки та управління виробництвом, до якої, поряд з NATTA, входить Конструкторська САПР, побудована на базі CATIA, а також PDM система SMARTEAM та ERP система SAP.

САПР ТП "NATTA" - власна розробка компанії "ГЕТНЕТ-Консалтинг". Головною відмінністю та перевагою NATTA є модульне проектування, що базується на асоційованих конструктивних та технологічних модулях, та синтез техпроцесів із технологічних модулів. Це обумовлює універсальність методології автоматизації технологічного проектування NATTA та високу гнучкість системи. Однак, з просуванням цієї розробки та впровадженням на підприємствах поки що великі труднощі, і немає відгуків про практичні результати впровадженої у виробництво системи «NATTA».

Система "T-FLEX/ТехноПро" є результатом співпраці фірми "Топ Системи" та фірми "Вектор-Альянс" (розробник популярної системи 32 СІАП ТП). Розробниками вдалося досягти повної інтеграції системи параметричного проектування «T-FLEX CAD» та системи автоматизованого проектування технологічних процесів «T-FLEX/ТехноПро». Пропонований комплекс програмних засобів дозволяє здійснити спільну автоматизацію конструкторських та технологічних підрозділів підприємства.

Управління технологічною підготовкою підприємств будується на зберіганні та використання інформації про продукцію на певних стадіях його життєвого циклу. Як один з базових інструментів реалізації CALS-технологій виступають системи класу PDM (Product Data Management). PDM є розрахованою на багато користувачів системою, яка працює в комп'ютерній мережі. Вона організує єдиний інформаційний простір підприємства, забезпечуючи створення, зберігання та обробку інформації у єдиній базі даних з допомогою системи управління базами даних (СУБД).

На сучасному етапі автоматизації проектування актуальна розробка комплексних систем автоматизованого проектування та виготовлення, що включають конструювання виробів, технологічне проектування та виготовлення виробів. Спроектований технологічний процес має оперативно реагувати зміну виробничих ситуацій процесу виготовлення виробів.

Аналіз використання діючих на підприємствах у технологічній підготовці виробництва пакетів прикладних програм показав, що більшість з них мають певну спрямованість на вирішення конкретних, локалізованих технологічних завдань з використанням великої різноманітності систем математичного забезпечення і, при цьому, не завжди вони методологічно взаємопов'язані, для їх реалізації потрібні специфічні вхідні потоки, які, зазвичай, немає взаємозв'язку з раніше створеними інформаційними базами.

Враховуючи принцип стандартизації при створенні АСТПП, необхідно прагнути пошуку вже «готових» систем, які, з одного боку, відповідають необхідним функціональним вимогам, а з іншого - вже довели свою надійність і якість при їх використанні на інших підприємствах. Такі універсальні «готові» системи, зазвичай, є розробками відомих фірм, що спеціалізуються у цій галузі, і можуть забезпечити рішення кола завдань.

На конкретному підприємстві можуть бути прийняті за основу, але за умови налаштування (адаптації) до умов конкретного виробництва. Налаштування полягає у заповненні баз даних відомостями про наявне на підприємстві обладнання, розробку алгоритмів (програм) проектування конкретних техпроцесів, опис форм конкретних документів, і т. д.

Одним із найбільш універсальних програмних продуктів, який може стати основою створення комплексу інтегрованого проектування, та інструментарієм, що дозволяє автоматизувати процес технологічного проектування, є система «ТехноПро». Система «ТехноПро» має основні критерії, що пред'являються користувачами до САПР.

Такими критеріями є: простота освоєння, можливість роботи з доступної техніці, простота створення документації різних форм та її повне відповідність існуючим вимогам і стандартам, можливість роботи з великими базами даних. Система "ТехноПро" є технологічним ядром для формування інтегрованих комплексів на основі CALS.

При цьому в комплексі можуть застосовуватись різні набори CAD/CAM, PDM та АСУП/ERP систем, реалізовані універсальні методи сполучення «ТехноПро» з PDM та ERP. Система «ТехноПро» забезпечує вирішення спеціалізованих завдань технологічного проектування за відомостями, одержуваними з CAD/CAM і PDM систем, і навіть передачу зведених технологічних даних АСУП/ERP і PDM. Комплекс системи «ТехноПро» суттєво розширює коло підсистем, які забезпечують можливості автономної роботи

програми. Для використання системи «ТехноПро» в інших галузях промисловості користувачам потрібно самостійно помістити в неї свої технологічні знання.

Потрібно витратити багато часу на адаптацію, щоб повною мірою розгорнути САПР ТП на шкіргалантерейному підприємстві.

2. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕАКТИВНОЇ ПОБУТОВИ ЩІЛЬНИХ УКЛАДОК

2.1. Представлення інформації про зовнішні контури плоских геометричних об'єктів

Для побудови щільних укладок необхідно мати інформацію про зовнішні контури плоского геометричного об’єкту у текстовому файлі. Нехай *S* – плоский геометричний об’єкт із заданою орієнтацією. Зв’яжемо з деталлю S координатну систему *XOY* , де *O* – полюс деталі, обраний у будь-якій її внутрішній точці.

Контур плоского геометричного об’єкту S апроксимуємою ламаною лінією з вершинами в послідовно вибраних на контурі деталі точках Кількість цих точок повинна забезпечувати потрібну точність апроксімації.

Тоді плоский геометричний об’єкт *S* можна представити координатами точок вершин апроксимуючого многокутника, тобто масивом координат вершин {*Xk ,Yk* }, *i = 1…n,* де *Xi,Yi* – координати *i*-ї вершини та *n* – кількість вершин апроксимуючого многокутника. Таке представлення дозволяє надати аналітичне описання зовнішнього контуру плоского геометричного об’єкту у вигляді систем рівнянь відрізків, з яких складаються ці контури. У параметричній формі запису ця система має вигляд:

$$\begin{cases} X(t_k) = X_k + (X_{k+1} - X_k)t_k \\ Y(t_k) = Y_k + (Y_{k+1} - Y_k)t_k \end{cases} \quad k = \overline{1,\ n}$$

де $\{X_k, Y_k\}$, *k=1..n,* - точки на контурі деталі, вибрані при апроксимації, або вершин многокутника, t_k - параметр, $t_k \in [0; 1)$.

Досвід свідчить, що для практичних задач при виготовленні взуття достатньо забезпечити точність апроксимації контурів 0,5 мм. Іншими словами, максимальне відхилення контуру деталі взуття від ланки апроксимуючої лінії не повинно перевищувати $H \le 0{,}5$ *мм*.

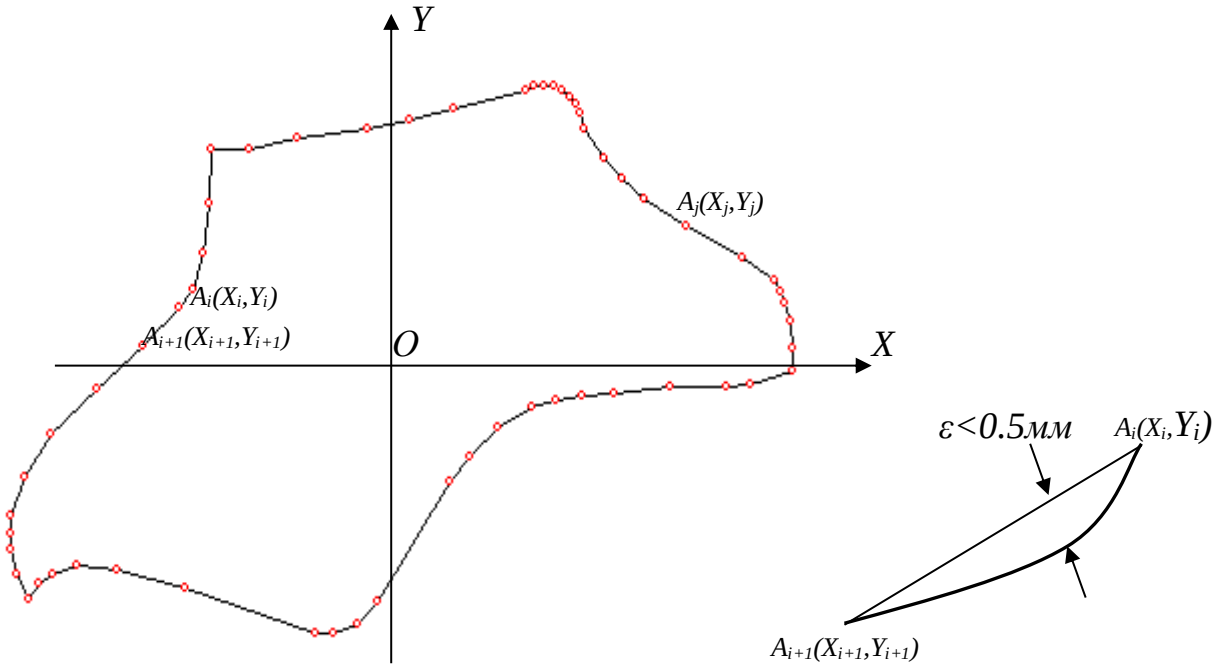

Рис. 2.1. Кусково-лінійна апроксимація

2.2. Решітчасті укладки

Розглянемо на площині об'єкти S_1 та S_2. Нехай $\text{int } S = S - S^{\wedge}$, де $S^{\wedge}$ – границя об'єкта S. Об'єкти S_1 та S_2 не перетинаються, якщо

$$\text{int } S_1 \cap \text{int} S_2 = 0 \qquad (2.1).$$

Якщо одночасно виконується умова

$$S_1 \cap S_2 \neq 0 \qquad (2.2),$$

то об'єкти S_1 та S_2 називаються щільно розміщеними.

Щільно розміщені об'єкти не мають спільних внутрішніх точок, але обов'язково мають спільні граничні точки.

Система об'єктів S_i , $i=1..p$, утворюють на площині укладку, якщо для кожної пари об'єктів із цієї системи виконуються умови їх взаємного неперетину:

$$\text{int } S_n \cap \text{int } S_m = 0,\ n \neq m,\ n,m=1..p \qquad (2.3)$$

та для будь-якого об'єкта S_i , $i=1..p$ знайдеться хоч один об'єкт S_q, де $q\epsilon[1..p]$, $p\neq i$, який дотикається до об'єкта S_i.

Позначимо через S+$\boldsymbol{a}$ об'єкт, який можна отримати переміщенням кожної точки об'єкта S на вектор $\boldsymbol{a}$ та назвемо його трансляцією об'єкта S.

Множину векторів виду

$$\boldsymbol{r}=n\boldsymbol{a_1}+m\boldsymbol{a_2}, \text{ де } n,m=0, \pm1, \pm2,\pm3,\dots\pm k\dots \qquad (2.4),$$

де $\boldsymbol{a_1}(a_{1x},a_{1y})$, $\boldsymbol{a_2}(a_{2x},a_{2y})$ - лінійно-незалежні вектори, назвемо решіткою з базисом $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ та позначимо через $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$.

Абсолютна величина визначника, який складений із векторів решітки, називається визначником решітки $\boldsymbol{\Lambda}$ та позначається $\det\boldsymbol{\Lambda}$, де [261]:

$$\det\boldsymbol{\Lambda}=|[\ \boldsymbol{a_1} \text{ x } \boldsymbol{a_2}\ ]|=\left\|\begin{bmatrix} a_{1x} & a_{1y} \\ a_{2x} & a_{2y} \end{bmatrix}\right\| = \left|a_{1x}a_{2y}-a_{2x}a_{1y}\right|. \qquad (2.5)$$

Розглянемо систему об'єктів $\bigcup_{n.m} S^{nm}$, де $n,m=0, \pm1, \pm2, \pm3,\dots\pm k\dots$, які складаються із трансляції $S^{nm}=S+n\boldsymbol{a_1}+m\boldsymbol{a_2}$ об'єкта S на вектори решітки $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$. Якщо ця система є укладкою, то така укладка називається укладкою об'єкта S, виконаної по решітці $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$. Решітка $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$ в цьому випадку є допустимою для укладки об'єкта S.

Щільність $\delta_s(\Lambda)$ решітчастої укладки можна характеризувати за допомогою співвідношення:

$$\delta_s(\Lambda)=|S|\ /\det\boldsymbol{\Lambda}\ , \qquad (2.6),$$

де |S| - площа плоского геометричного об'єкта S, $\det\boldsymbol{\Lambda}$ - визначник решітки $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$, за якою виконана укладка. Із наведеного співвідношення видно, що щільність $\delta_s(\boldsymbol{\Lambda})$ решітчастої укладки тим вища чим менша площа паралелограма, сторонами якого є базові вектори решітки $\boldsymbol{a_1}$ та $\boldsymbol{a_2}$.

Множину векторів виду:

$$\boldsymbol{r_1}=n\boldsymbol{a_1}+m\boldsymbol{a_2} \text{ та } \boldsymbol{r_2}=n\boldsymbol{a_1}+m\boldsymbol{a_2}+\boldsymbol{g}, \text{ де } n,m=0, \pm1, \pm2, \pm3,\dots\pm k\dots\ ,$$

де $\boldsymbol{a_1}$, $\boldsymbol{a_2}$- лінійно-незалежні вектори, назвемо подвійною решіткою з базисом $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ і вектором зсуву решітки $\boldsymbol{g}$ та позначимо через $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$. Абсолютна величина визначника, який складений із базових векторів подвійної решітки, називається визначником решітки та позначається det $\boldsymbol{W}$.

Розглянемо систему об'єктів $\bigcup_{n.m} S_1^{nm}$ та $\bigcup_{n.m} S_2^{nm}$, де $n,m=0, \pm1, \pm2, \pm3, \ldots \pm k \ldots$, які складаються із трансляції $S_1^{nm}=S_1+n\boldsymbol{a_1}+m\boldsymbol{a_2}$ та $S_2^{nm}=S_2+n\boldsymbol{a_1}+m\boldsymbol{a_2}+\boldsymbol{g}$ об'єктів S_1 та S_2 на вектори подвійної решітки $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$ [118]. Якщо ця система є укладкою, то така укладка називається укладкою об'єктів S_1 та S_2, виконаної по подвійній решітці $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$. Подвійна решітка $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$ в цьому випадку є допустимою для укладки об'єктів S_1 та S_2. Фрагмент подвійної решітчастої укладки представлений на рис. 2.2. В цьому випадку під об'єктом S_1 мають на увазі деталь S(0) у вихідному положенні, а під об'єктом S_2 мають на увазі деталь $S(\pi)$, повернуту на 180^0 відносно вихідного положення. Подвійна решітка представляє собою дві однакові одинарні решітки $\boldsymbol{\Lambda_1}=\boldsymbol{\Lambda}(\boldsymbol{a_1},\boldsymbol{a_2})$ та $\boldsymbol{\Lambda_2}=\boldsymbol{\Lambda}(\boldsymbol{a_1}+\boldsymbol{g}, \boldsymbol{a_2}+\boldsymbol{g})$, які зміщені одна відносно іншої на вектор зсуву решітки $\boldsymbol{g}$. У вузлах решітки $\boldsymbol{\Lambda_1}$ розміщаються об'єкти S_1, а у вузлах решітки $\boldsymbol{\Lambda_2}$ розміщаються об'єкти S_2.

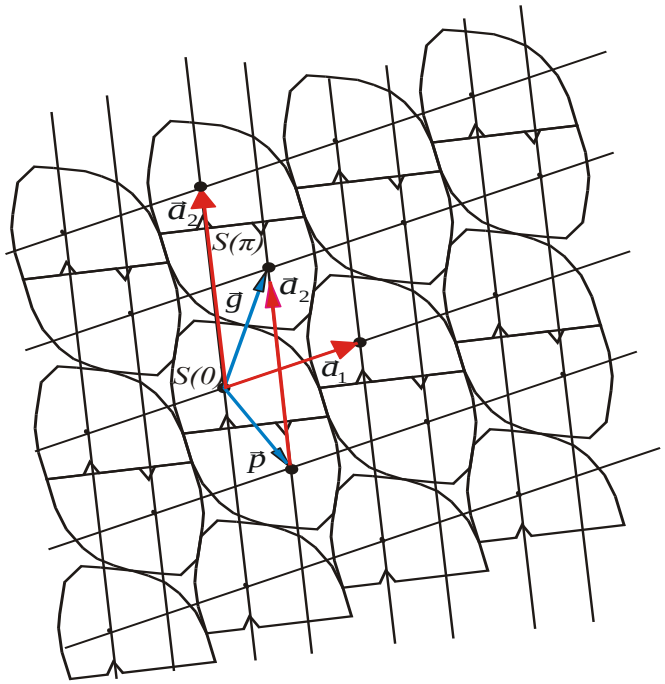

Рис. 2.2. Фрагмент подвійної решітчастої укладки

Абсолютна величина визначника, який складений із векторів решітки, називається визначником решітки $\boldsymbol{W}$ та позначається det $\boldsymbol{W}$, де:

$$\det \boldsymbol{W} = |[\, \boldsymbol{a_1} \times \boldsymbol{a_2} \,]| = \left| \begin{bmatrix} a_{1x} & a_{1y} \\ a_{2x} & a_{2y} \end{bmatrix} \right| = \left| a_{1x}a_{2y} - a_{2x}a_{1y} \right|. \qquad (2.7)$$

Щільність $\delta_s(\boldsymbol{W})$ решітчастої укладки можна характеризувати за допомогою співвідношення:

$$\delta_s(\boldsymbol{W}) = (|S_1| + |S_2|) / \det \boldsymbol{W}\,, \qquad (2.8)$$

де $|S_1|$ та $|S_2|$ - відповідно площі плоского геометричного об'єкта S_1 та S_2, det $\boldsymbol{W}$ - визначник решітки $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{g})$, за якою виконана укладка. Із наведеного співвідношення видно, що щільність $\delta_s(\boldsymbol{W})$ решітчастої укладки тим вища, чим менша площа паралелограма, сторонами якого є базові вектори решітки $\boldsymbol{a_1}$ та $\boldsymbol{a_2}$.

На основі технологічної постановки задач проектування щільних укладок в паралелограмі сформулюємо математичні постановки цих задач.

Математична постановка задачі „Укладка А". Серед множини допустимих решіток $\boldsymbol{\Lambda^i}=\boldsymbol{\Lambda}(\boldsymbol{a^i_1}, \boldsymbol{a^i_2})$, де $i=1,2..q,$ для побудови щільних укладок для однакових та однаково орієнтованих плоских геометричних об'єктів S знайти таку решітку $\boldsymbol{\Lambda^*}=\boldsymbol{\Lambda}(\boldsymbol{a^*_1}, \boldsymbol{a^*_2})$, для якої $\delta_s(\boldsymbol{\Lambda^*})=|S| / \det\boldsymbol{\Lambda^*}=max(\delta_s(\boldsymbol{\Lambda^i}))$, де $|S|$ - площа плоского геометричного об'єкту S . Так як площа плоского геометричного об'єкту S є постійною величиною, то серед допустимих решіток $\boldsymbol{\Lambda^i}=\boldsymbol{\Lambda}(\boldsymbol{a^i_1},\boldsymbol{a^i_2})$ необхідно знайти таку, для якої $\det\boldsymbol{\Lambda^*}=min(\det\boldsymbol{\Lambda^i})$.

Математична постановка задачі „Укладка Б". Серед множини допустимих подвійних решіток $\boldsymbol{W^i}=\boldsymbol{W}(\boldsymbol{a^i_1},\boldsymbol{a^i_2},\boldsymbol{g^i})$, де $i=1,2..q,$ для побудови щільних укладок для однакових плоских геометричних об'єктів S з поворотом в рядах на 0^0 та 180^0, знайти таку решітку $\boldsymbol{W^*}=\boldsymbol{W}(\boldsymbol{a^*_1}, \boldsymbol{a^*_2},\boldsymbol{g^*})$, для якої $\delta_s(\boldsymbol{W^*})=|S| / \det\boldsymbol{W^*}=max(\delta_s(\boldsymbol{W^i}))$, або $\det\boldsymbol{W^*}=min(\det\boldsymbol{W^i})$.

2.3. Визначення площі деталей взуття як площі апроксимуючого випукло-вгнутого многокутника

Аналіз існуючих методів апроксимації плоских геометричних об'єктів показав, що найбільш підходить кусково-лінійний метод апроксимації для опису геометрії взуттєвих деталей та деталей шкіргалантерейних виробів. Тоді інформацію про геометрію плоских геометричних об'єктів будемо подавати у вигляді масивів координат точок апроксимації зовнішньої границі деталей взуття $\{X_k, Y_k\}$, $k=1..n$, де точки $\{X_k, Y_k\}$ - є вершинами опукло-вгнутого многокутника, апроксимуючого зовнішній контур плоского геометричного об'єкту. Отже площа апроксимуючого многокутника буде дорівнювати:

$$S_n = \left| \iint_{\Omega} dXdY \right| = \frac{1}{2} \left| \oint \left(XdY - YdX \right) \right| = \frac{1}{2} \left| \sum_{k=1}^{n-1} \left(\int_{Y_k}^{Y_{k+1}} XdY - \int_{X_k}^{X_{k+1}} YdX \right) \right|, \qquad (2.7)$$

де $X_n = X_1; Y_n = Y_1$

Так як рівняння прямої, яка проходить через точки (X_k, Y_k) и (X_{k+1}, Y_{k+1}) має вигляд:

$\frac{X - X_k}{X_{k+1} - X_k} = \frac{Y - Y_k}{Y_{k+1} - Y_k}$, то $Y = \frac{Y_{k+1} - Y_k}{X_{k+1} - X_k} \cdot (X - X_k) + Y_k$ і $dY = \frac{Y_{k+1} - Y_k}{X_{k+1} - X_k} \cdot dX$.

Підставивши вирази для Y і dY у вираз (2.1) і розрахувавши інтеграли

$$\int_{Y_k}^{Y_{k+1}} XdY = \int_{X_k}^{X_{k+1}} \frac{Y_{k+1} - Y_k}{X_{k+1} - X_k} \cdot dX = \frac{(X_{k+1} + X_k) \cdot (Y_{k+1} - Y_k)}{2}$$

$$\int_{X_k}^{X_{k+1}} YdX = \int_{X_k}^{X_{k+1}} \left[\frac{Y_{k+1} - Y_k}{X_{k+1} - X_k} \cdot (X - X_k) + Y_k \right] \cdot dX = \frac{(X_{k+1} - X_k) \cdot (Y_{k+1} + Y_k)}{2}$$

отримаємо

$$S_n = \frac{1}{2} \left| \sum_{k=1}^{n-1} \left(X_k Y_{k+1} - X_{k+1} Y_k \right) \right| \qquad (2.8)$$

Очевидно, що площа плоского геометричного об'єкту може бути наближено визначена як площа апроксимуючого опукло-вгнутого многокутника. При чому

точність визначення площі плоского геометричного об’єкту залежить від кількості точок апроксимації. Чім більше точок апроксимації зовнішнього контуру плоского геометричного об’єкту, тим вища точність визначення площі плоского геометричного об’єкту і дійсна площа плоского геометричного об’єкту буде визначатися наступним чином:

$$S = \lim S_n = \lim \frac{1}{2}\left|\sum_{k=1}^{n-1}\left(X_k Y_{k+1} - X_{k+1} Y_k\right)\right| \qquad (2.9)$$

2.4. Ітерактивна побудова щільних укладок

Для інтерактивної побудови щільних укладок необхідно вирішити задачу контролю не перетину активної деталі із вже розміщенимив укладкі.

Розглянемо цю задачу більш детально. Умову взаємного не перетину активної деталі із вже розміщеними деталями в укладкі можна представити наступним чином: якщо жодна вершина на зовнішньому контурі активної деталі не лежить в середині жодної із вже розміщених деталей в укладці та жодна вершина із уже розміщених не лежить всередині активної деталі, то активна деталь не перетинається із вже розміщеними деталямив укладці. Цю задачу можна звести до задачі визначення взаємного розміщення многокутника та точки, а саме точка знаходиться всередині многокутника чи зовні. Для цього застосуємо метод кутів.

Розглянемо метод кутів для вирішення проблеми належності точки. При цому підході необхідно визначити поняття кута зі знаком. Нехай маємо направлений відрізок прямої лінії $\overline{bc}$ і деяка точка *a* і нехай кут між векторами $\overline{ab}$ і $\overline{bc}$ дорівнюватиме θ. Тоді кут зі знаком для точки *a* відносно відрізка $\overline{bc}$ дорівнюватиме θ, якщо точка *c* лежить ліворуч від вектора $\overline{ab}$ або коленіарна з ним, або – θ, якщо точка *c* лежить праворуч від вектора $\overline{ab}$. Примітимо, що кут зі знаком і орієнтація трикутника ΔABC мають однаковий знак.

Розширемо визначення кута зі знаком для низки вершин. Кут зі знаком для точки *a* відносно низки вершин дорівнюватиме суммі кутів зі знаком для точки *a* відносно ребер, що входять в цей ланцюг. Розглянемо тепер, як кути зі знаком можуть бути використані для визначення розташування точки *a* відносно заданого багатокутника *P.* Позначимо буквою α величину кута зі знаком в точці *a* відносно границі багатокутника *P,* при чому обхід багатокутника виконується в напрямку годинникової стрілки. Значення кута α дозволяє визначити приналежність точки: якщо α = 360 градусів, то точка *A* лежить в середині багатокутника , і α = 0, якщо точка *A* розташована поза багатокутником.

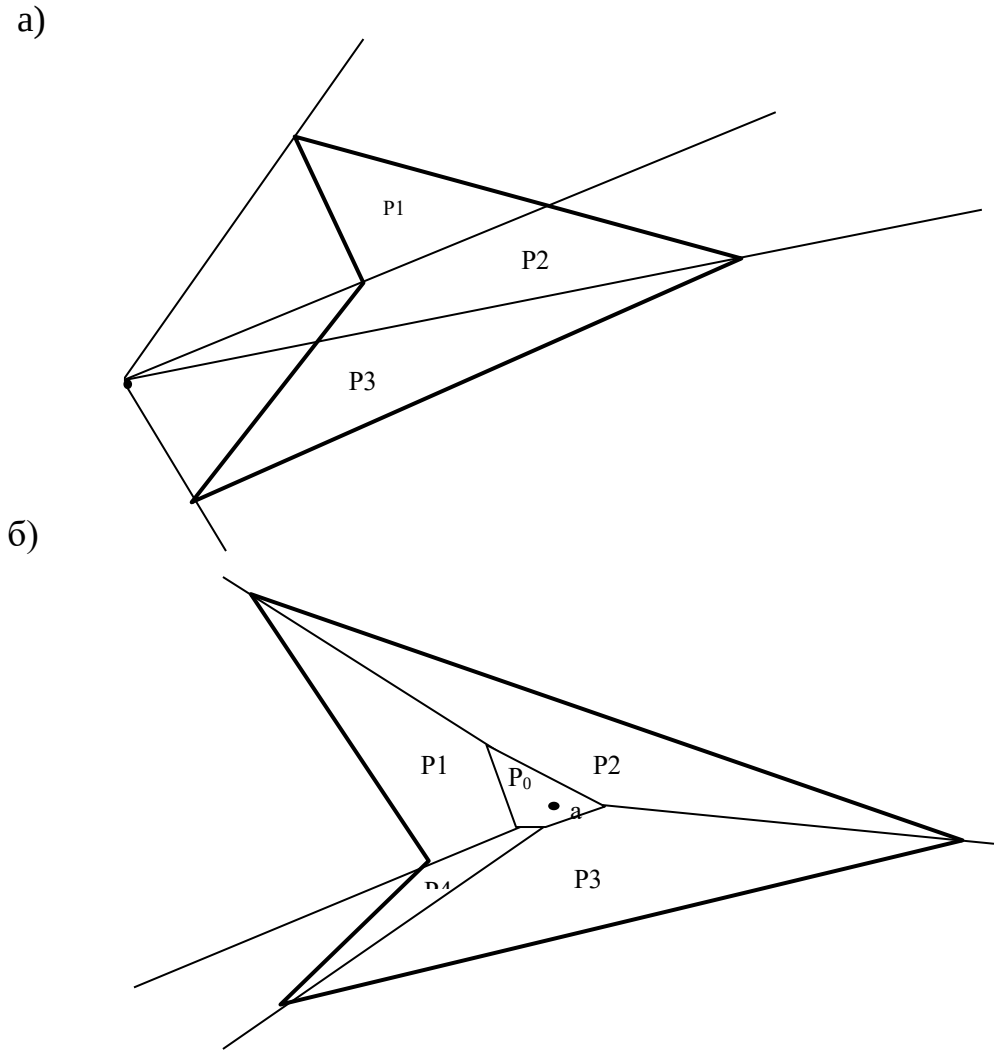

Рис. 2.3. Розташування точки
а) поза багатокутником б) в багатокутнікі

Покажемо справедливість цього ствердження для випуклих багатокутників. Якщо точка *A* знаходиться в середині випуклого багатокутника *P,* то при обході границі багатокутника *P* виконується оберт на повні 360 градусів. В протилежному випадку, якщо точка *A* розташована поза багатокутником *P,* то границя багатокутника може бути розщеплена на два ланцюги, ближній і дальній до точки *a*. Якщо через α_N позначити кут зі знаком відносно ближнього ланцюга, а через α_f - кут зі знаком відносно дальнього ланцюга, то будемо мати $\alpha_N = -\alpha_f$, звідки маємо: $\alpha = \alpha_N - \alpha_f = 0$.

Менш очевидно, що ця умова зберігається і для невипуклих багатокутників. Спочатку припустимо, що точка *a* знаходиться поза багатокутником. Уявимо, що з точки *a* через кожну вершину багатокутника проведено промінь, розбиваючи таким чином багатокутник *P* на декілька трикутників та випуклих чотикутників $p_1, p_2, ..., p_k$ (рис.2.3.а).

Оскільки з рис. 2.3.б зрозуміло, чому $\alpha = 360$ град., якщо точка знаходиться всередині полігона *p*. Як і раніше, припустимо розподіл полігона променем, що виходить з точки *A,* але на цей раз передбачимо невеликий випуклий полігон в околі точки *A* (p_0). Оскількии p_0 – це випуклий полігон і в середині нього знаходиться точка *A*, то $\alpha = 360$ град.

Для визначення сумарного кута необхідно знайти елементарні кути. Елементарні кути будуть мати знак. Для визначення знаку елементарного кута α (рис.2.4) скористаємося модулем векторного добутку :

$$| [\boldsymbol{OA_{j+1}} \times \boldsymbol{OA_j}] | = \begin{vmatrix} X_{ij+1} - X_0 & Y_{ij+1} - Y_0 \\ X_{ij} - X_0 & Y_{ij} - Y_0 \end{vmatrix} = (X_{ij+1} - X_0)(Y_{ij} - Y_0) - (X_{ij} - X_0)(Y_{ij+1} - Y_0),$$

де (X_{ij+1}, Y_{ij+1}) – координати точки A_{j+1};

(X_{ij}, Y_{ij}) – координати точки A_j; (X_o, Y_o) – координати точки A_o;

Якщо $| [\boldsymbol{OA_{j+1}} \times \boldsymbol{OA_j}] | > 0$, то кут буде додатнім, якщо $| [\boldsymbol{OA_{j+1}} \times \boldsymbol{OA_{j+1}}] | < 0$, то кут буде від'ємним, якщо $| [\boldsymbol{OA_{j+1}} \times \boldsymbol{OA_j}] | = 0$, то кут дорівнює нулю.

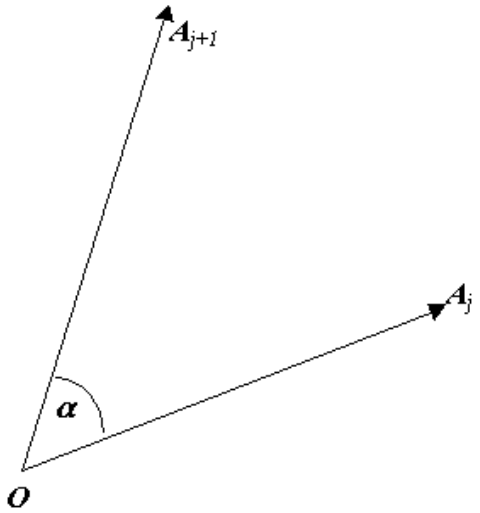

Рис. 2.4.

Визначимо cos φ із скалярного добутку векторів $\boldsymbol{a=OA_{j+1}}$ та $\boldsymbol{b=OA_j}$. Будемо мати:

$$\cos\varphi = \frac{(ab)}{|a||b|} = \frac{(X_{ij+1} - Xo)(X_{ij} - Xo) + (Y_{ij} - Yo)(Y_{ij+1} - Yo)}{\sqrt{(X_{ij+1} - Xo)^2 + (Y_{ij+1} - Yo)^2}\sqrt{(X_j - Xo)^2 + (Y_j - Yo)^2}} \qquad (2.9)$$

Визначимо sin φ із модуля векторного добутку векторів $\boldsymbol{b}$ та $\boldsymbol{a}$. Будемо мати:

$$\sin\varphi = \frac{[ab]}{|a||b|} = \frac{(X_{ij+1} - Xo)(Y_{ij} - Yo) - (X_{ij} - Xo)(Y_{ij+1} - Yo)}{\sqrt{(X_{ij+1} - Xo)^2 + (Y_{ij+1} - Yo)^2}\sqrt{(X_j - Xo)^2 + (Y_j - Yo)^2}} \qquad (2.10)$$

Для прискорення роботи алгоритму визначення взаємного розміщення багатокутників P_1 та P_2 розглянемо задачу взаємного розміщення прямокутників A та B, які відповідно описані навколо багатокутників P_1 та P_2 (рис.2.5 та рис.2.6). Нехай прямокутник B за аналогією з прямокутником A визначається системою нерівностей:

$$\begin{cases} X^b_{\min} \le x \le X^b_{\max} \\ Y^b_{\min} \le y \le Y^b_{\max} \end{cases} \qquad (2.11)$$

Тоді нехай $X_{\min} = \max(X^a_{\min}, X^b_{\min})$; $Y_{\min} = \max(Y^a_{\min}, Y^b_{\min})$;

$X_{\max} = \min(X^a_{\max}, X^b_{\max})$; $Y_{\max} = \min(Y^a_{\max}, Y^b_{\max})$. (2.12)

Якщо $X_{nin} > X_{\max}$ та $Y_{\min} > Y_{\min}$, то прямокутники перетинаються, тобто можливо перетинаються багатокутники P_1 та P_2.

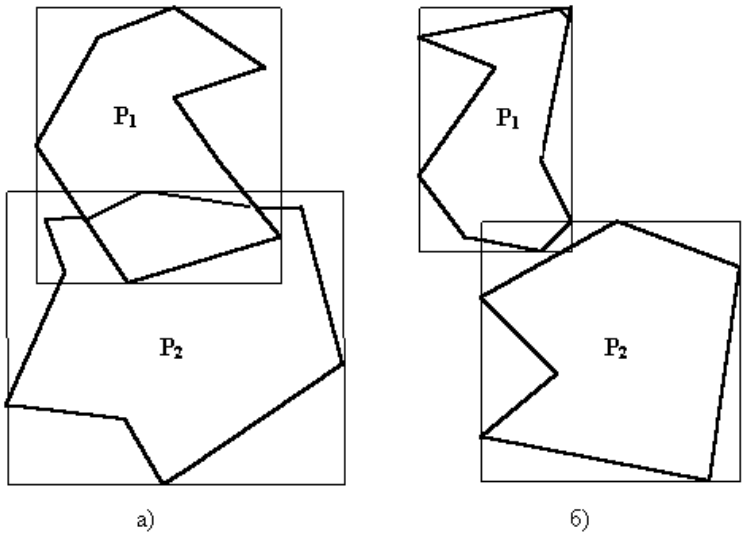

Рис. 2.5

Для вияснення факту перетину двох деталей, необхідно вияснити факт перетину багатокутників P_1 та P_2, які апроксимують ці деталі. Для цього потрібно значення сумарних кутів для кожної із вершин апрокимуючого багатокутника P_1 відносно апроксимуючого багатокутника P_2 та значення сумарних кутів для кожної із вершин апроксимуючого багатокутника P_2 відносно апроксимуючого багатокутника P_1. І якщо знайдеться хоч один сумарний кут, значення якого дорівнює 360 градусів, то багатокутники P_1 та P_2 перетинаються, інакше багатокутники P_1 та P_2 не перетинаються.

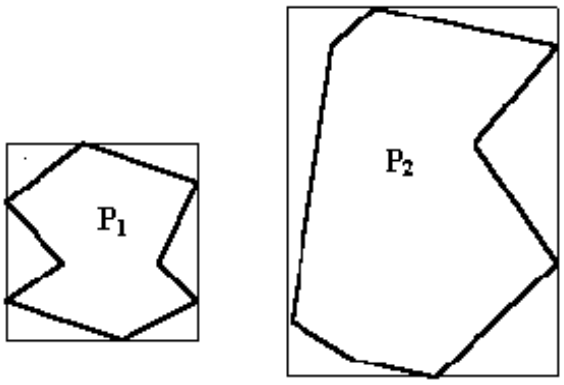

Рис. 2.6

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕРАТИВНОЇ ПОБУТОВИ ЩІЛЬНИХ УКЛАДОК

3.1. Вимоги до програмного продукту

Програмний продукт повинен забезпечувати:

- надійну роботу як в локальному, так і в мережевому варіантах;
- високу точність. Будь-які обмеження на кількість внутрішніх контурів і число точок лекальних кривих в конструюванні ведуть, в кінцевому підсумку, до втрати точності при відтворенні складних деталей;
- гнучкість роботи. Як мінімум, повинні бути можливість скасування операцій на будь-яку кількість кроків, можливості введення та редагування будь-якої кількості додаткових точок і інших елементів креслення на будь-якому етапі конструювання. Дуже корисний механізм автоматичних прив'язок до характерних точок лекальні кривих;
- швидкість. Швидка змінюваність моделей, розширення асортиментної бази неможливі без потужного графічного редактора та конструкторського модуля (не плутати з «креслярськими засобами для конструювання»). Сучасний конструкторський модуль повинен забезпечувати виготовлення комплекту лекал для найскладнішої моделі протягом 2-3 годин;
- багато документальний інтерфейс. Сучасні системи дозволяють відкривати відразу кілька моделей при роботі. Вільно і наочно виділяти і переносити з моделі в модель будь-які елементи креслення - будь то лекала або окремі модельні лінії. Без обмеження комбінувати нові моделі на основі наявних;
- роботу з будь-яким серійним обладнанням. Вільний обмін даними з іншими програмами. Крім усього іншого, це полегшить створення єдиної мережі підприємства;
- вивід на друк в будь-якому масштабі на будь-якому етапі роботи.

3.2. Вибір системи програмування для практичної реалізації запропонованих методів та алгоритмів інтерактивної побудови щільних укладок

Система програмування Delphi 7 фірми Enterprise (Borland) надає найбільш широкі можливості для програмування додатків ОС Windows.

Високопродуктивний інструмент візуального побудови додатків Delphi версії 7 включає в себе справжній компілятор коду і надає засоби візуального програмування, кілька схожі на ті, що можна виявити в Microsoft Visual Basic або в інших інструментах візуального проектування. В основі Delphi лежить мова Object Pascal, який є розширенням об'єктно-орієнтованої мови Pascal. У Delphi також входять локальний SQL-сервер, генератори звітів, бібліотеки візуальних компонентів, і інше, необхідне для того, щоб відчувати себе абсолютно впевненим при професійній розробці інформаційних систем або просто програм для Windows-середовища.

Насамперед Delphi призначений для професійних розробників, бажаючих дуже швидко розробляти додатки. Delphi виробляє невеликі за розмірами високоефективні виконувані модулі (.exe і .dll), тому в Delphi мають бути, перш за все, зацікавлені ті, хто розробляє продукти на продаж. З іншого боку невеликі за розмірами і швидко виконувані модулі означають, що вимоги до клієнтських робочих місцях істотно знижуються - це має важливе значення і для кінцевих користувачів.

Переваги Delphi в порівнянні з аналогічними програмними продуктами :

- Швидкість розробки додатку;
- Висока продуктивність розробленого додатка;
- Низькі вимоги розробленого додатка до ресурсів комп'ютера;
- Можливість розширення за рахунок вбудовування нових компонент та інструментів в середовище Delphi;

- Можливість розробки нових компонентів та інструментів використовуючи розробки Delphi (існуючі компоненти та інструменти доступні у вихідних кодах);
- Вдале опрацювання ієрархії об'єктів.

Система програмування Delphi розрахована на програмування різних додатків і надає велику кількість компонентів для цього. До того ж роботодавців цікавить, насамперед, швидкість і якість створення програм, а ці характеристики може забезпечити тільки середовище візуального проектування, здатне взяти на себе значні обсяги рутинної роботи з підготовки додатків, а також узгодити діяльність групи постановників, кодувальників, тестерів. Можливості Delphi повністю відповідають подібним вимогам і підходять для створення систем будь-якої складності.

Для того щоб обгрунтувати, чому наш вибір зупинився на Borland Delphi 7, досить просто перерахувати деякі недоліки мови C ++ порівняно з ObjectPascal :

1. Треба робити багато ініціалізації (реєструвати клас вікна, організовувати цикл обробки повідомлень, створювати віконну функцію, піктограму та інше 2. Частково бути системним програмістом. На Delphi-ж системне програмування вже вбудовано і ініціалізація працює за замовчуванням, тому програміст головний акцент робить на своїх алгоритмах, а не на організації допоміжних робіт.

3. Значно більша, порівняно з Object Pascal, складність мови, навіть, незважаючи на компактність коду, виникають складнощі в його сприйнятті.

4. Одна особливість, на мій погляд, мови C ++ дуже псує цю мову - він чутливий до регістру символів, тобто змінна A і змінна a - це різні змінні.

5. У Delphi класи (об'єкти) можуть розташовуватися тільки в динамічної пам'яті, а в C ++ в будь-якій пам'яті (статична, стек, динамічна). Це додає безпеки програмування в Delphi.

3.3. Опис основних процедур розробленого програмного продукту, які забезпечують інтеративну побудову щільних укладок для плоских об’єктів

Процедура *TForm1.Open1Click* забезпечує введення інформації про зовнішні контури деталей моделі взуття для яких необхідно інтеративно побудовати щільні укладки:

```
procedure TForm1.Open1Click(Sender: TObject);
  Var x1,x2,y1,y2,m:integer;
     St:string[10];
  Procedure Vvod;
  //читает ввод файла *.dgt
  Var F,Fr:System.Text;
{    FileName,}Str,NameF:string;
    xi,yi,xe,ye,Xc,Yc:Int64;
    i,j,l,m,lr,x1,y1,x2,y2:word;
    nxi,nye,nyi:integer;
    pr:byte;
    Xrab,Yrab,A:real;
begin
{   Form1.Left:=0; Form1.Top:=0;}
   if OpenDialog1.Execute then
  begin
   FileName:=OpenDialog1.FileName;
   System.Assign(F,FileName);
   L:=Length(FileName);
   NameF:='';
   For i:=1 to L-3 do NameF:=NameF+FileName[i];
   NameF:=NameF+'Ukl';
   System.Assign(fr,NameF);
   ReWrite(Fr);
```

```
System.Close(fr);
System.Erase(Fr);
Reset(F);
Readln(F,Model);
Caption:='Побудова щільних укладок для моделі  '+Model;
Readln(F,A);
Readln(F,KolDet);
For i:=1 to KolDet do
  Readln(F,NameDet[i]);
For i:=1 to KolDet do
 begin
  Readln(F,KolPointDet[i],Rost[i]);
  nn[i]:=KolPointDet[i];
 end;
For i:=1 to KolDet do
 begin
  For j:=0 to nn[i]-1 do
   begin
    readln(f,Xrab,Yrab);
    If KolPointDet[i]<>nn[i] then continue;
    if (Xrab=999)and(Yrab=999) then
     begin
      KolPointDet[i]:=j;
      continue
     end;
    x[i,j]:=Round(Xrab*100);
    y[i,j]:=Round(Yrab*100);
{    BitBtn1.Caption:=IntToStr(i)+' '+IntToStr(j);}
```

```
    end;
      nn[i]:=KolPointDet[i];
      MaxMin(nxi,KolPointDet[i],x[i],xi,-1);
      MaxMin(nxi,KolPointDet[i],y[i],yi,-1);
      MaxMin(nxi,KolPointDet[i],x[i],xe,1);
      MaxMin(nxi,KolPointDet[i],y[i],ye,1);
      Xc:=Round((xe+xi)/2);
      Yc:=Round((ye+yi)/2);
      Delta_X[i]:=Round((Xe-Xi)/2);
      Delta_Y[i]:=Round((Ye-Yi)/2);
     for j:=0 to KolPointDet[i]-1 do
      begin
       x[i,j]:=x[i,j]-Xc;
       y[i,j]:=y[i,j]-Yc;
       xd[i,j]:=x[i,j];
       yd[i,j]:=y[i,j];
      end;
     Skv(nn[i],x[i],y[i],Sdet[i]);
   end;
  end;
  MaxMin(nxi,KolDet,Delta_X,MaxDlDet,1);
  MaxMin(nxi,KolDet,Delta_Y,MaxShDet,1);

   {PrF:=True;}
 System.Close(F);
 end;
Begin
Form1.Image3.Visible:=False;
```

```
 Vvod;
//        Sh:=150000;
  //       mxy:=Image2.Height/Sh;
  //  ScrollBox1.Visible:=true;
//    GroupBox1.Visible:=true;
    RadioGroup2.Visible:=true;
    RadioButton3.Visible:=true;
    RadioButton4.Visible:=true;
    //  Label8.Visible:=true;
   { Edit4.Visible:=true;}
  X1:=5;
  X2:=50;
  B_Kor:=False;
  //Image1.Height:=55*(koldet+10)+5;
     Scala(40,50,Dxy);
  With ScrollBox1,Image1.Canvas do
    For i:=1 to koldet do
     begin
      Y1:=5+55*(i-1);
      Y2:=55*i;

       Font.Name:='Ariel';
       Font.Size:=7;
{       Font.Style:=[fsBold];,fsItalic];}
        Font.Color:=clBlack;
        St:=Copy(NameDet[i],1,8);
       TextOut(X1,Y1,st);
     Grd(KolPointDet[i],Xd[i],Yd[i],27,55*i-25,Dxy,i);
```

```
    //   Rectgl(x1,y1,x2,y2,i);
   MoveTo(X1,Y1);
   LineTo(X1,Y2);
   LineTo(X2,Y2);
   LineTo(X2,Y1);
   LineTo(X1,Y1);
  end;
procedure scale_im2(var mx:real);
var   my:real ;
begin
mx:=form1.Image2.Width/dl_dt ;
my:=form1.Image2.Height /sh_dt ;
if mx>my then mx:=my;
end;
```

Процедура *Grd* забезпечує вивід зображення многокутника на область *Image1*.

Параметри процедури:

- *p*-номер кольору:

```
If p=1 then Pen.Color:=ClBlue
 else if p=2 then Pen.Color:=ClRed
  else if p=3 then Pen.Color:=ClMaroon
   else if p=4 then Pen.Color:=ClNavy
    else if p=5 then Pen.Color:=ClPurple
     else if p=6 then Pen.Color:=ClTeal
      else if p=7 then Pen.Color:=ClGray
       else if p=8 then Pen.Color:=ClLime
        else if p=9 then Pen.Color:=TColor(RGB(255,128,64)){ClYellow}
```

```
else if p=10 then Pen.Color:=ClFuchsia
else if p=11 then Pen.Color:=ClAqua
else if p=12 then Pen.Color:=ClSilver
else if p=13 then Pen.Color:=ClOlive
else if p=14 then Pen.Color:=ClGreen
else Pen.Color:=ClBlack;
```

mxy - масштаб, в якому виводиться зображення многокутника вобласть Image1;

n - кількість вершин многокутника*;*

xm,ym - масиви з координатами вершин многокутника*;*

dxc,dyc – координати центру прямокутника, описаного навколо многокутника, зображення якого виводиться в область *Image1;*

```
Procedure Grd(n:integer; Var xr,yr:array of Int64;
 //рисует детали на image1
  dxc,dyc:integer; mxy:real;p:integer);


  Var i:integer;
    xp,yp:array[0..200] of integer;
   Begin
    If p>15 then
     If P mod 15=0 then P:=15
     else P:=P mod 15;
    for i:=0 to n-1 do
     begin
      xp[i]:=round(xr[i]*mxy)+dxc;
      yp[i]:=round(yr[i]*mxy)+dyc;
     end;
    With Form1.Image1,Canvas Do
```

```
begin
  If p=1 then Pen.Color:=ClBlue
   else if p=2 then Pen.Color:=ClRed
    else if p=3 then Pen.Color:=ClMaroon
     else if p=4 then Pen.Color:=ClNavy
      else if p=5 then Pen.Color:=ClPurple
       else if p=6 then Pen.Color:=ClTeal
        else if p=7 then Pen.Color:=ClGray
         else if p=8 then Pen.Color:=ClLime
          else if p=9 then Pen.Color:=TColor(RGB(255,128,64)){ClYellow}
           else if p=10 then Pen.Color:=ClFuchsia
            else if p=11 then Pen.Color:=ClAqua
             else if p=12 then Pen.Color:=ClSilver
              else if p=13 then Pen.Color:=ClOlive
               else if p=14 then Pen.Color:=ClGreen
                else Pen.Color:=ClBlack;

   MoveTo(xp[0],yp[0]);
   for i:=1 to n-1 do
    LineTo(xp[i],yp[i]);
  end;
End;
```

*П*роцедура *Skv* визначає площу *s* многокутника, який визначається наступними параметрами:

n - *кількість* вершин многокутника*;*

x,y - масиви з координатами вершин многокутника.

```
Procedure Skv(n:integer; Var x,y:array of Int64; Var s:Int64);
```

```
Var i:integer;
Begin
s:=0;
For i:=0 to n-2 do
s:=s+x[i]*y[i+1]-x[i+1]*y[i];
s:=Round(abs(s)/2)
End;
```

*П*роцедура *MaxMin* визначає визначає порядковий номер *ne* елементу масиву *xy* та його максимальне(мінімальне) значення *xye* при значенні параметру *np*=1(при значенні параметру *np*=-1).

```
Procedure MaxMin(var ne:integer; n:integer;
var xy:array of Int64; var xye:Int64; np:integer);
Var j:integer;
begin
ne:=1;
xye:=xy[1];
for j:=0 to n-1 do
if xye*np<xy[j]*np then
begin
xye:=xy[j];
ne:=j
end
end;
```

*П*роцедура *PolDet* визначає максимальне значення *xe (ye)* масиву *x(y)* та мінімальне значення *xi (yi)* масиву *x(y)* наступними параметрами:

n - *кількість* вершин многокутника*;*

x,*y* - масиви з координатами вершин многокутника.

```
Procedure PolDet(n:integer; var x,y:array of Int64;
var xe,ye,xi,yi:Int64);
var n2,k:integer;
```

```
Begin
 k:=1;
MaxMin(n2,n,y,ye,k);
MaxMin(n2,n,x,xe,k);
 k:=-1;
MaxMin(n2,n,y,yi,k);
MaxMin(n2,n,x,xi,k);
End;
```

Процедура *Scala* забезпечує визначення масштабу *mxy*, в якому буде будуваться щільна укладка для активної деталі за наступними параметрами:

dln,shn – відповідно довжина та ширина многокутника, описаного навколо активної деталі.

```
Procedure Scala(dln,shn:word; Var mxy:real);
 Var xxi,yyi,xi,yi,xxe,yye,xe,ye,dl,ds:Int64;
  mx:real;
  j:integer;
 Begin
  PolDet(KolPointDet[1],x[1],y[1],xxe,yye,xxi,yyi);
 For j:=2 to KolDet do
  begin
   PolDet(KolPointDet[j],x[j],y[j],xe,ye,xi,yi);
     if xxi>xi then xxi:=xi;
     if yyi>yi then yyi:=yi;
     if xxe<xe then xxe:=xe;
     if yye<ye then yye:=ye;
  end;
 dl:=xxe-xxi;
```

```
  ds:=yye-yyi;
  mx:=(dln-4)/dl;
  mxy:=(shn-4)/ds;
  if mxy>mx then mxy:=mx;
End;
```

*П*роцедура *PerSqr* визначає чи перетинаються два прямокутники за їх максимальним значенням *xe1, xe2* координати *X (ye1, ye1,* координати *У* масиву *x(y)* та мінімальним значенням *xi1, xi2* координати *X (yi1, yi2* координати У*)* масиву *x(y).* Параметр *b* приймає значення *true* коли прямокутники перетинаються, інакше приймає значення *false.*

n - *кількість* вершин многокутника*;*

x,y - масиви з координатами вершин многокутника.

```
Procedure PerSqr(xe1,ye1,xi1,yi1,xe2,ye2,xi2,yi2:Int64;
        Var b:boolean);
  Begin
   b:=false;
   if xi1<xi2 then xi1:=xi2;
   if yi1<yi2 then yi1:=yi2;
   if xe1>xe2 then xe1:=xe2;
   if ye1>ye2 then ye1:=ye2;
   if (xi1<xe1)and(yi1<ye1) then b:=true
 End;
```

3.4. Інструкції по роботі з програмним продуктом

Початок роботи з програмою розпочинається з запуску файлу *PrMinenko.exe*. При цьому на екрані з‘являється головне вікно програми прийме наступний вигляд(рис. 3.1).

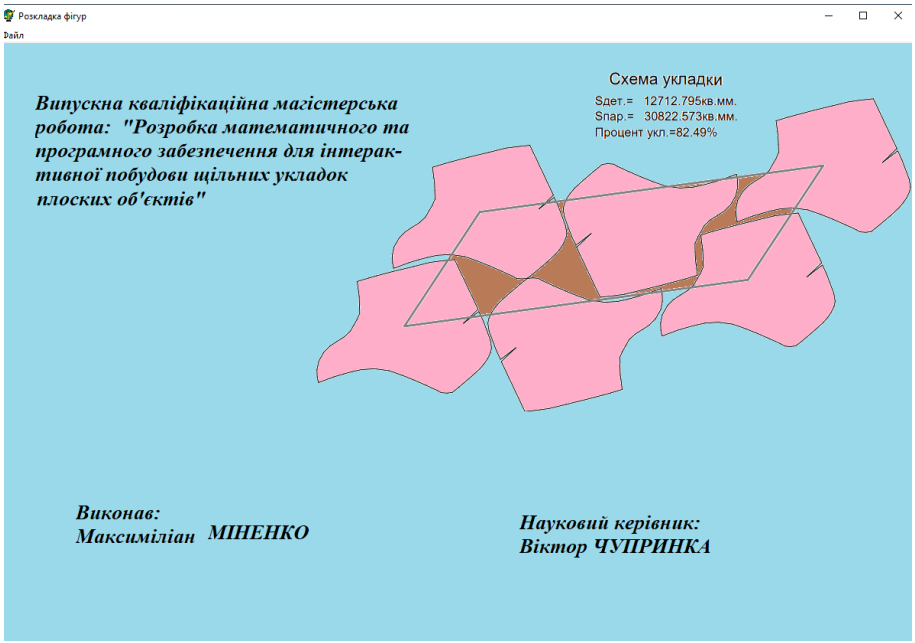

Рис. 3.1

Тепер вводимо інформацію про зовнішні деталей моделі. Після цього головне вікно програми настуний вигляд (рис. 3.2).

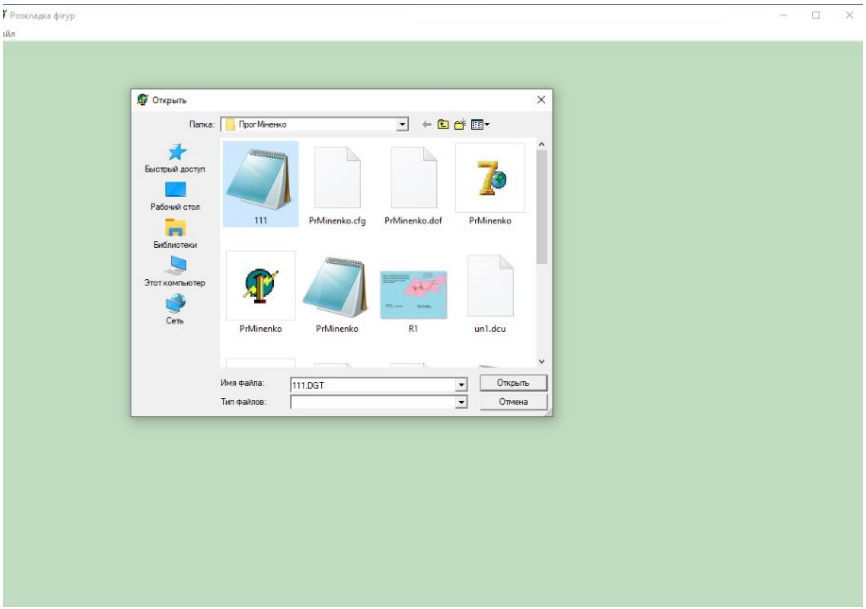

Рис. 3.2

Після вводимо інформацію про зовнішні деталей моделі головне вікно програми настуний вигляд (рис. 3.3).

.

Рис. 3.3

Після вибору схеми укладки без повороту чи з пороротом на 180^0 (нехай це буде з пороротом на 180^0) та натиску на кнопку «Добавити» головне вікно програми настуний вигляд (будується опукла оболонка деталі)(рис. 3.4).

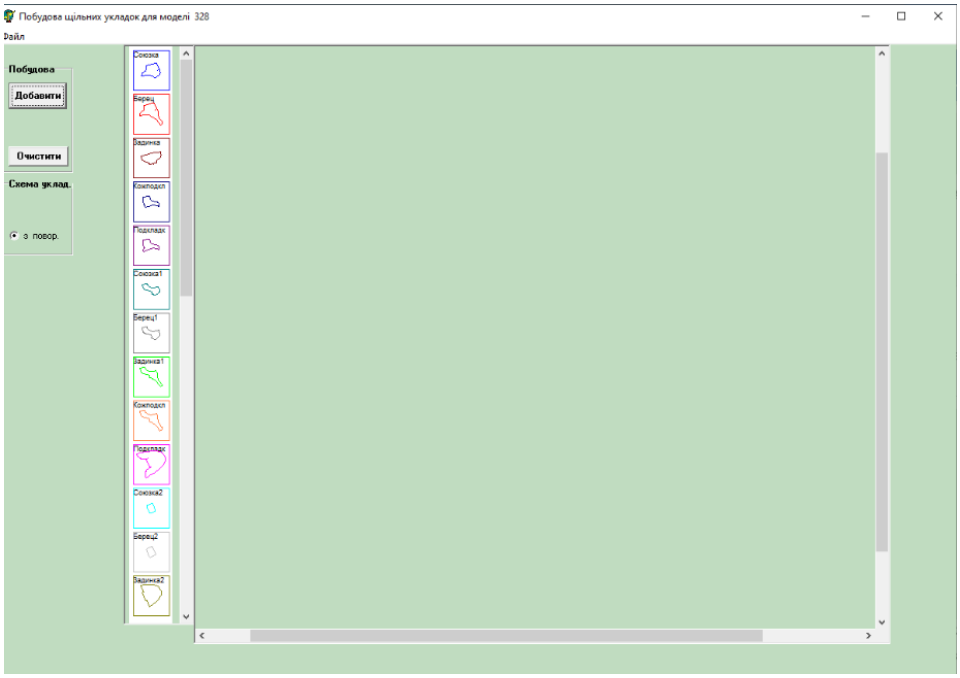

Рис. 3.4

Після вибору деталі із меню, що знаходиться ліворуч, головне вікно програми настуний вигляд (рис. 3.5).

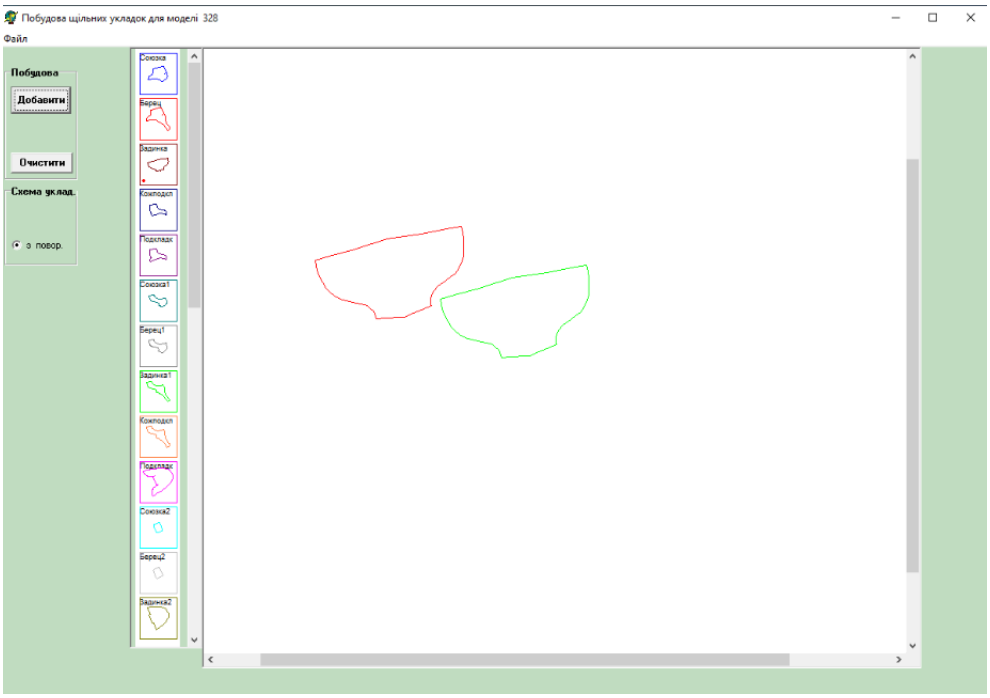

Рис. 3.5

Після щільного суміщення двох однаково орієнтованих деталей головне вікно програми настуний вигляд (рис. 3.6).

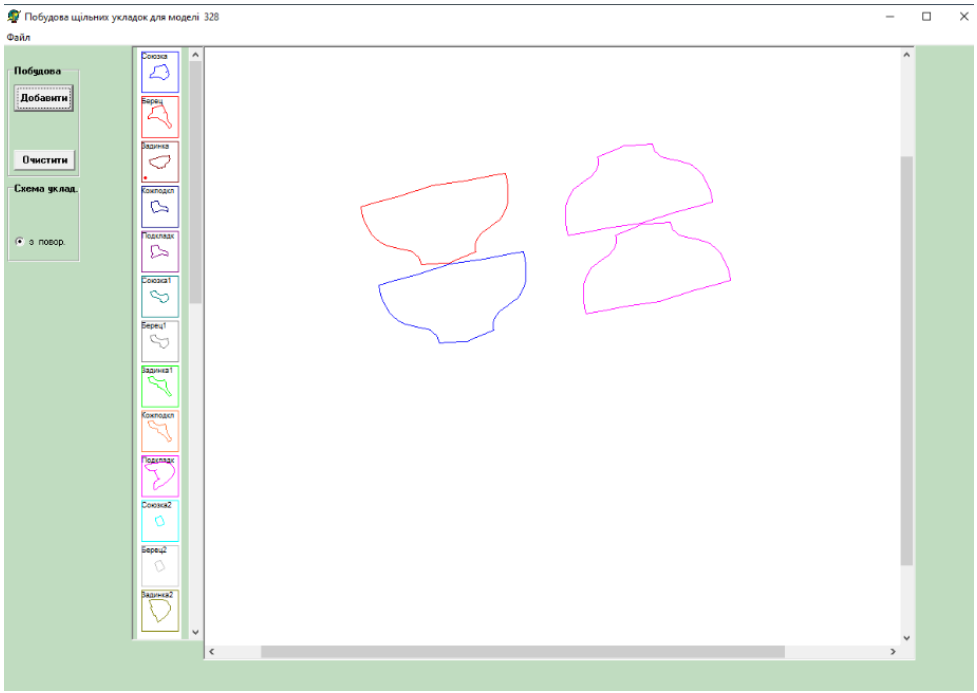

Рис. 3.6

Після щільного суміщення двох однаково орієнтованих деталей з двома деталями, що повернуті на 180^0 головне вікно програми настуний вигляд (рис. 3.7).

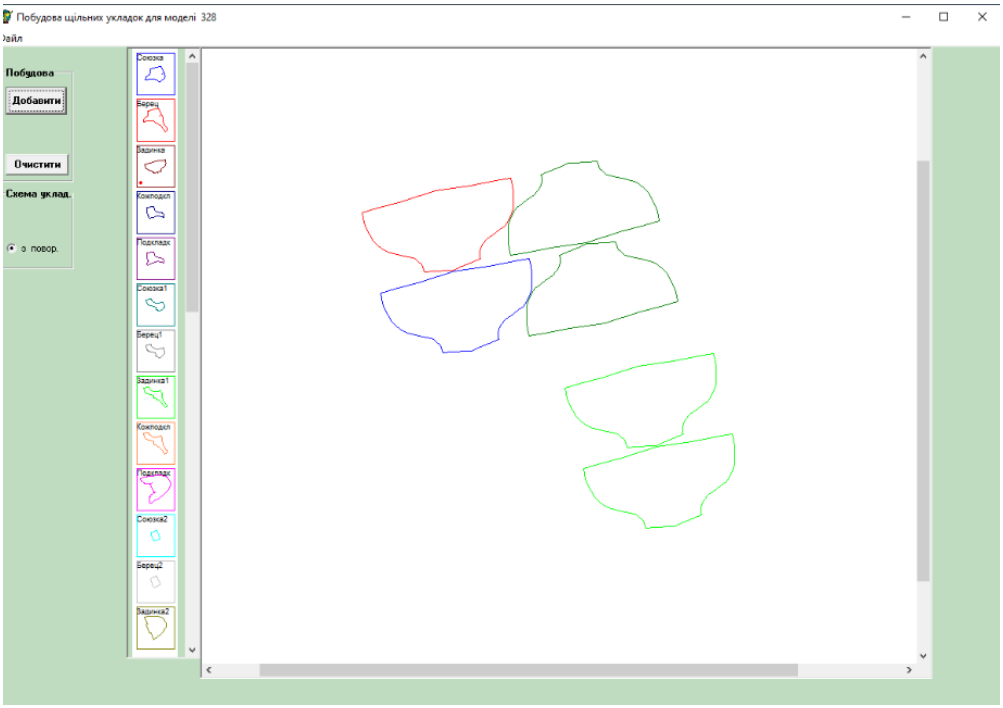

Рис. 3.7

Після щільного суміщення чотирьох деталей з двома деталями головне вікно програми настуний вигляд (рис. 3.8).

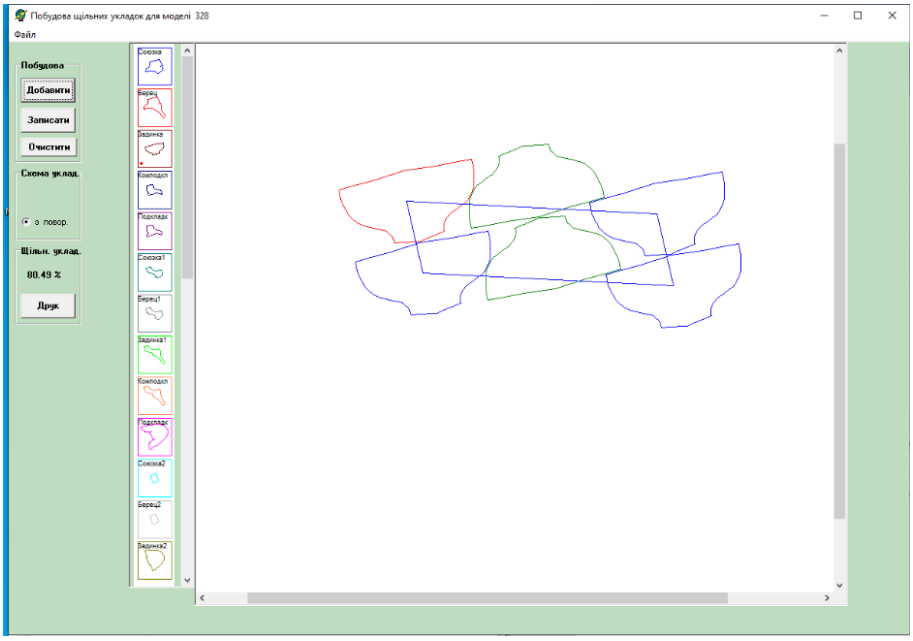

Рис. 3.8

Побудовану схему укладки можна вивести на друк. Для цього треба натиснути на кнопку «Друк». Після цього головне вікно програми настуний вигляд (рис. 3.9).

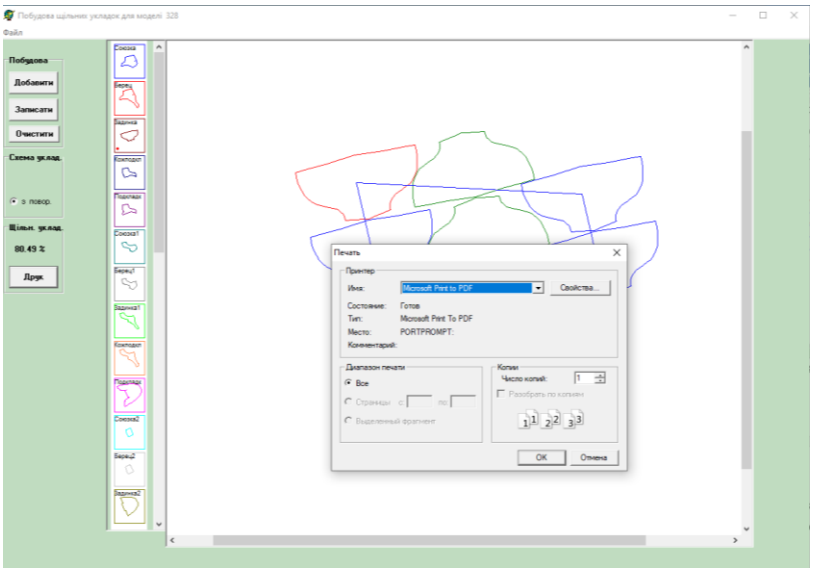

Рис. 3.9

На друк щільна укладка виводиться в наступному вигляді(рис. 3.10)

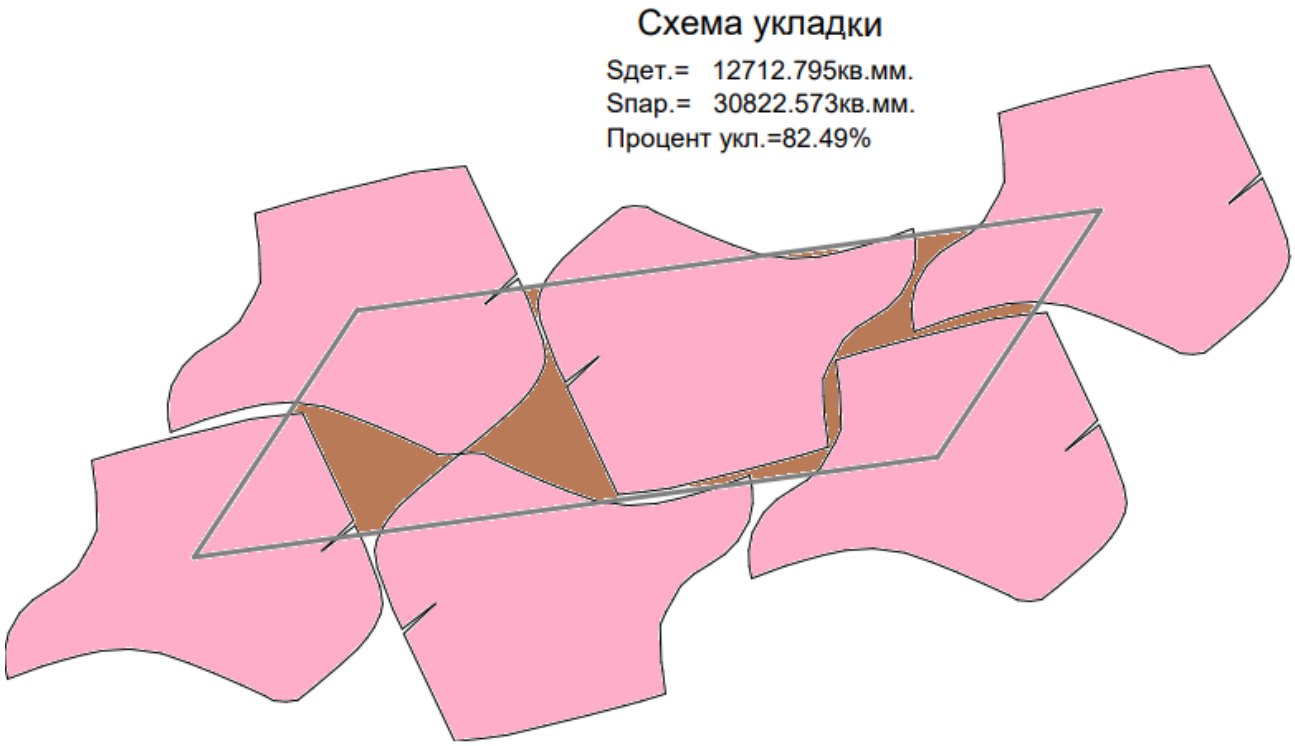

Рис. 3.10

Побудовану схему укладки можна зберегти у файлі. Для цього треба натиснути на кнопку «Зберегти». Після цього головне вікно програми настуний вигляд (рис. 3.11).

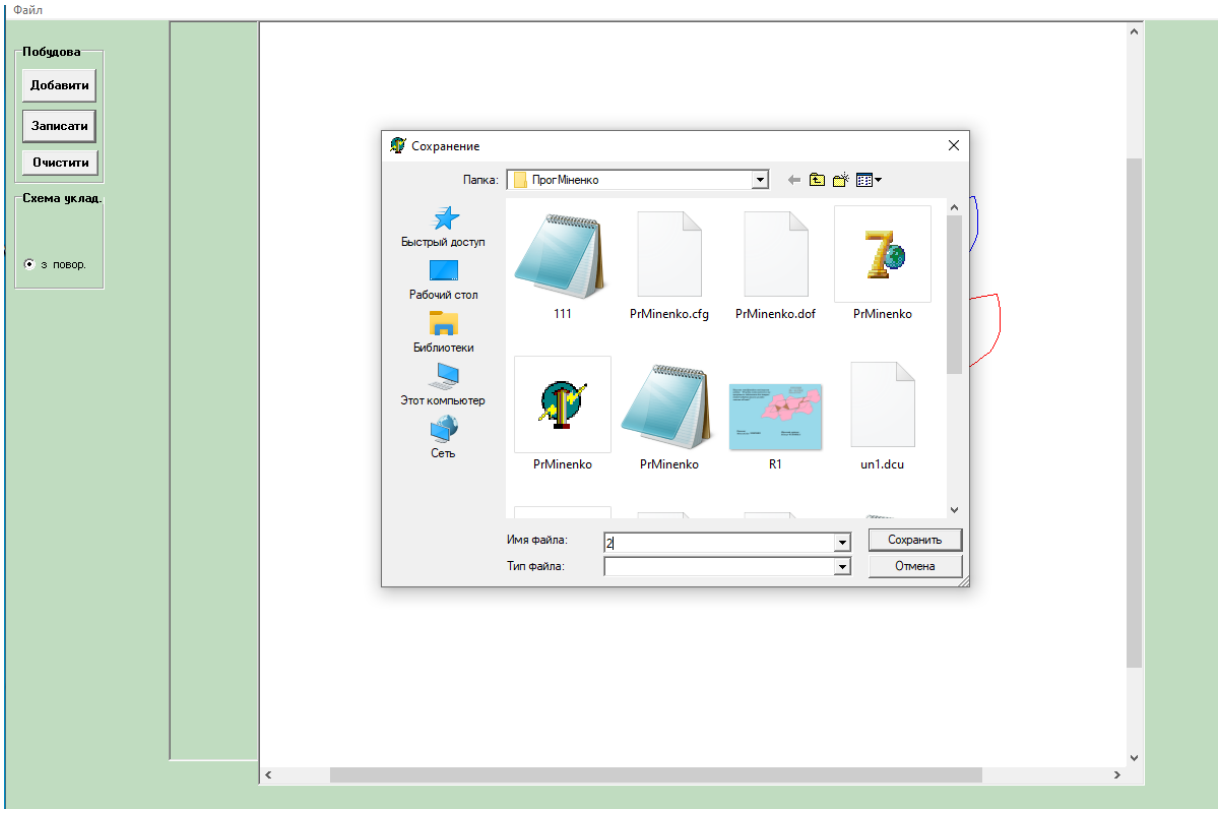

Рис. 3.11

ІНФОРМАЦІЯ У ФАЙЛІ ПРОЗБЕЖЕНУ ІНФОМАЦІЮ МАЄ НАСТУПНИЙ ВИГЛЯД:

N	X	Y	КУТ
3	338.17	376.82	0
3	359.42	469.57	0
3	527.54	450.25	180
3	506.29	357.50	180
3	683.10	486.00	0
3	661.85	393.25	0

Приклади побудованих в інтерактивному режимі щільних укладок за допомогою розробленого програмного продукту представлені на рис. 3.12

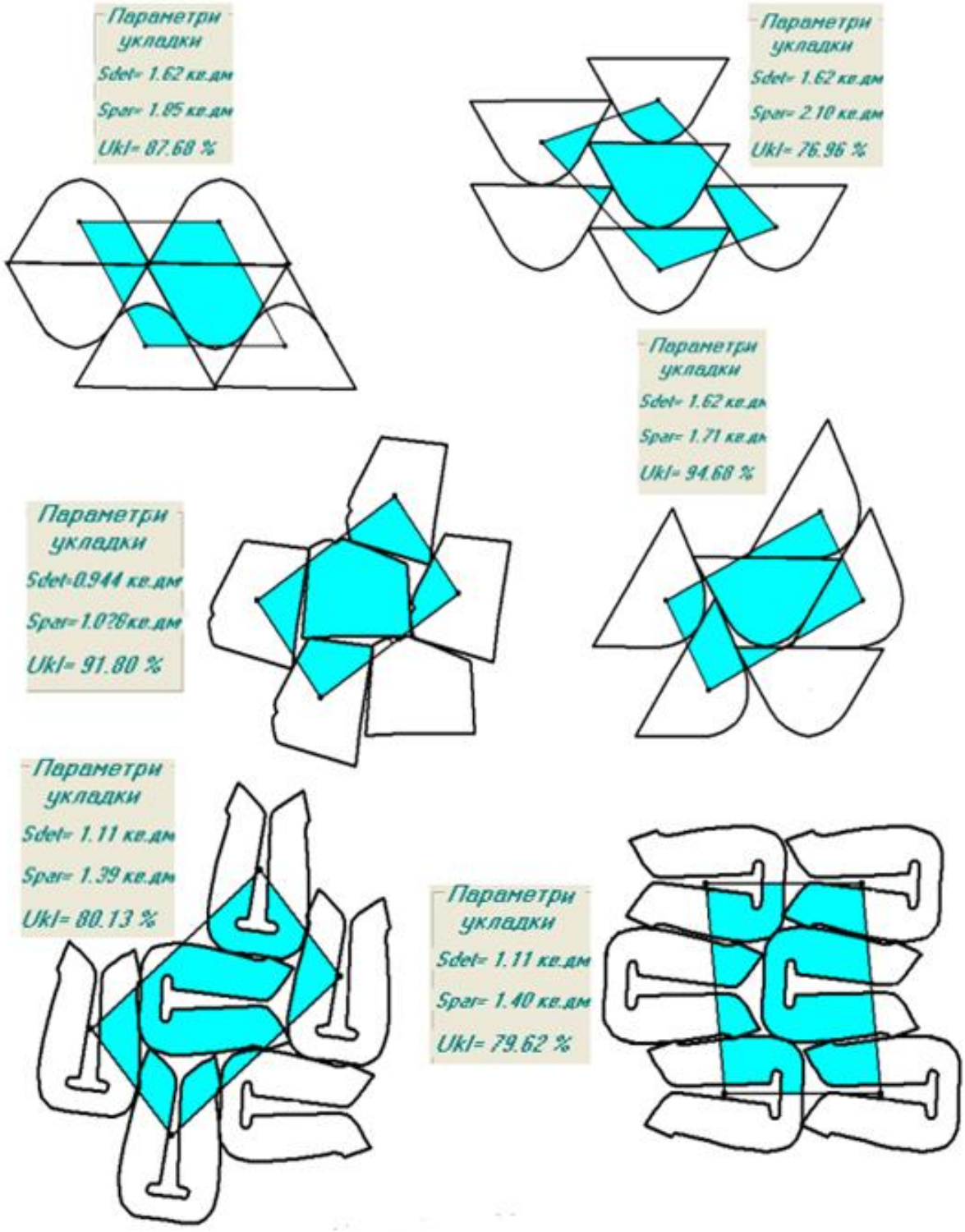

Рис. 3.12

ВИСНОВКИ

Проведено дослідження та розроблені такі алгоритми для інтерактивного проектування щільних укладок для плоских геометричних обєктів:

- визначення взаємого положення геометричних обєктів(перетинаються чи ні) ;
- інтеративного побудови щільних укладок деталей в основном положенні;
- інтеративного побудови щільних укладок деталей з поворотом на 180^0;
- Запропоноване математичне забезпечення для інтеративного побудови щільних укладок деталей в основном положенні та для деталей з поворотом на 180^0.

Запропоноване математичне забезпечення для інтеративного проєектування щільних укладок реалізоване в прогрограмний продукт. Цей програмний продукт легкий в користуванні, має дружній та привабливий інтерфейс. Може бути застосована на підприємствах в взуттєвої галузі легкої промисловості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дякон В. М. Моделі і методи теорії прийняття рішень : підручник / В. М. Дякон, Л. Є. Ковальов. – К.: АНФ ГРУП, 2013. – 604 с.
2. Зайченко Ю.П. Дослідження операцій: Підручник. – К.: ВІПОЛ, 2010.
3. K. Weiler. An incremental angle point in polygon test, in: P. Heckbert (Ed.), Graphic Gems IV, Academic Press, Boston, MA, 1994, pp. 16—23.
4. Hormann K., Agathos A. The point in polygon problem for arbitrary polygons (англ.) // Comput. Geom. Theory Appl.. — 2001. — Vol. 20. — P. 131—144.
5. J. Melissen. Packing 16, 17 or 18 circles in an equilateral triangle // Discrete Mathematics. — 1995. — Т. 145. — С. 333–342. — doi:10.1016/0012-365X(95)90139-C.
6. Y. G. Stoyan, G. N. Yaskov. Packing identical spheres into a cylinder // International Transactions in Operational Research. — 2010. — Т. 17. — С. 51–70. — doi:10.1111/j.1475-3995.2009.00733.x.
7. P. Erdős, R. L. Graham. On packing squares with equal squares // Journal of Combinatorial Theory, Series A. — 1975. — Т. 19. — С. 119-123. — doi:10.1016/0097-3165(75)90099-0.
8. Lodi, S. Martello, M. Monaci. Two-dimensional packing problems: A survey // European Journal of Operational Research. — Elsevier, 2002. — Т. 141. — С. 241–252. — doi:10.1016/s0377-2217(02)00123-6.
9. A. Haji-Akbari, M. Engel, A. S. Keys, X. Zheng, R. G. Petschek, P. Palffy-Muhoray, S. C. Glotzer. Disordered, quasicrystalline and crystalline phases of densely packed tetrahedra // Nature. — 2009. — Т. 462, вип. 7274. — С. 773–777 — doi:10.1038/nature08641. — Bibcode: 2009Natur.462..773H. — arXiv:1012.5138. — PMID 20010683.

10. E. R. Chen, M. Engel, S. C. Glotzer. Dense Crystalline Dimer Packings of Regular Tetrahedra // Discrete & Computational Geometry. — 2010. — Т. 44, вып. 2. — С. 253–280. — doi:10.1007/s00454-010-9273-0.
11. E. G. Birgin, R. D. Lobato, R. Morabito. An effective recursive partitioning approach for the packing of identical rectangles in a rectangle // Journal of the Operational Research Society. — 2010. — Т. 61. — С. 306–320. — doi:10.1057/jors.2008.141.
12. D.A. Klarner, M.L.J Hautus. Uniformly coloured stained glass windows // Proceedings of the London Mathematical Society. — 1971. — Т. 23. — doi:10.1112/plms/s3-23.4.613.
13. Eckard Specht. The best known packings of equal circles in an isosceles right triangle (11 березня 2011).
14. U. Betke, M. Henk. Densest lattice packings of 3-polytopes // Comput. Geom.. — 2000. — Т. 16. — С. 157–186.
15. H. Minkowski. Dichteste gitterförmige Lagerung kongruenter Körper // Nachr. Akad. Wiss. Göttingen Math. Phys. KI. II. — 1904. — С. 311–355.
16. Packing unit squares in squares: a survey and new results // The Electronic Journal of Combinatorics. — 2005. — Т. DS7.
17. Fudos I. Geometric Constraint Solving. PhD thesis, Purdue University, Dept of Computer Science, 1995.
18. Fudos I. and Hoffmann C.M. A graph-constructive approach to solving systems of geometric constraints // ACM Transactions on Graphics, 16:179-216, 199
19. Hoffmann C.M., Sitharam M., Yuan B. Making constraint solvers more usable: overconstraint problem // Computer-Aided Design 36(4): 377-399 (2004)
20. Hoffmann C.M., Yuan B. // On Spatial Constraint Solving Approaches. Automated Deduction in Geometry 2000: 1-15
21. Hoffmann C.M., Lomonosov A., Sitharam M. Planning Geometric Constraint Decomposition via Optimal Graph Transformations // AG-TIVE 1999, Applications

of Graph Transformations with Industrial Relevance, International Workshop, The Netherlands: 309-324

22. Jermann C., Trombettoni G., Neveu B., Rueher M. A Constraint Programming Approach for Solving Rigid Geometric Systems // Principles and Practice of Constraint Programming CP 2000, 6th International Conference, Singapore: 233-248

23. Sitharam M., Arbree A., Zhou Y., Kohareswaran N. Solution space navigation for wellconstrained geometric constraint systems // University of Florida, Submitted in ACM Transactions on Graphics, http://www.cise.ufl.edu/~sitharam/pub.htm

24. Hoffman C.M., Lomonosov A., Sitharam M. Decomposition Plans for Geometric Constraint Systems, Part I: Performance Measures for CAD. Journal of Symbolic Computation, Volume 31, №4, 2001, pages 367-408

25. Hoffman C.M., Lomonosov A., Sitharam M. Decomposition Plans for Geometric Constraint Systems, Part II: New Algorithms. Journal of Symbolic Computation, Volume 31, №4, 2001, pages 409-427

26. Sitharam M. Combinatorial approaches to geometric constraint solving: problems, progress and directions // DIMACS Workshop on computer aided design and manufacturing. 2004 http://www.cise.ufl.edu/~sitharam/pub.html

27. Sitharam M., Oung J.-J., Zhou Y., Arbee A. Geometric constraints within feature hierarchies. 2004

28. Durand C. Symbolic and Numerical Techniques for Constraint Solving. PhD thesis, Purdue University, Dept of Computer Science, 1998.

29. Wielinga B.J., Akkermans J.M., Schreiber A.Th. A formal analysis of parametric design problem solving // Proceedings of the 9th Banff Knowledge Acquisition Workshop, pages 37-1 37-15

30. Durand C., Hoffman C.M. A systematic framework for solving geometric constraints analytically // Journal of Symbolic Computation, Volume 30, №5, 2000, pages 493-519

35 Hoffman C.M., Joan-Arinyo R. Symbolic constraints in constructive geometric constraint solving // Journal of Symbolic Computation, Volume 23, №2-3, 1997, pages 287 299

36. Chen J., Kanj I.A. Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms // Journal of Computer and System Sciences № 67, 2003, pages 833-847

37. Hoffman C.M., Peters J. Geometric constraints for CAGD // Mathematical Methods for Curves and Surfaces, M.Daehken, T.Lyche & L.L.Schumaker ed., Vanderbilt University Press, pages 237-253, 1995.

38. Joan-Arinyo R., Soto-Riera A. Combining constructive and equa-tional geometric constraint solving techniques // ACM Transactions on Graphics, Volume 18, №1, 1999, pages 35-55

39. Joan-Arinyo R., Soto-Riera A., Vila-Marta S. Tools to Deal with Under-constrained Geometric Constraint Graphs // The Asian Symposium on Computer Mathematics ASCM 2003, Beijing, October 23-25, 2003

40. Yong-Sang Pae, Geometric constraint satisfaction by generalized degrees of freedom analysis // PhD thesis, University of Texas, Austin, Faculty of the Graduate School, 1997.

41. Michelucci D., Foufouy S. Using Cayley-Menger Determinants for Geometric Constraint Solving // ACM Symposium on Solid Modeling and Applications (2004)

42. Cleve Ashcraft, Joseph W.H. Liu, Applications of the Dulmage-Mendelsohn Decomposition and Network Flow to Graph Bisection Improvement // SIAM J. Matrix Anal., 19 (1998), pages 325-354.

43. Lovasz, L., M.D. Plummer, M.D. Matching Theory, North-Holland, 1986

44. Adamovicn A., Albano A. Nesting two-dimensional shapes in rectangular Modules // Comput. Aeded Design. 1976. 8(1). P. 27-33.

45. Aurts E., Lenstra J., edit. Local Search in Combinatorial Optimization // John Wiley&Sons. 1996.

46. Belov G., Scheitchauer G., Mukhacheva E.A. On dimensional heuristics adapted for two-dimensional rectangular strip packing. Technical report. Dresden University. 2006. URL http://www.math.tu-dresden.de/capad/. Preprint MATH-NM-02-2006.

47. Belov G., Scheithauer G. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths // European Journal of Operational Research. 2002. 141. P. 274-294.

48. Berman P. Approximating rectilinear polygon cover problems / Berman P., Dasgupta B. // Algorithmica. #17(4). 1997. P. 331-356.

49. Blazewicz J., Hawryluk P., Walkowiak R. Using a tabu search approach for solving the two-dimensional irregular cutting problem // Annals of OR. 1993. 41(4). P. 313-325.

50. Bortfeldt A., Gehring H. Applying tabu search to container loading problems // Operations Research Proceedings. 1997, Springer, Berlin, 1998, P. 533538.

51 Boschetti M.A. The Two-Dimensional Finite Bin Packing Problem // Qua-terly Journal of the Belgian, French and Italian Operations Research Societies, 2002.,

52. Bronnimann H. almost optimal set covers in finite VC-dimension. / Bron-nimann H., Goodrich M. // Discrete comput. Geom., #14. 1995. P.263-279.

53. Burke E., Kendall G. Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem // Accepted for the 1999 International Conference on Artificial Intelligence, Monte Carlo resort. Las Vegas. Nevada. USA. 1999.

54. Liu, X., Liu, J. M., Cao, A. X., and Yao, Z. L. (2015). HAPE3D-a new constructive algorithm for the 3D irregular packing problem. *Front. Inf. Technol. Electron. Eng.* 16 (5), 380–390. doi:10.1631/FITEE.1400421. https://link.springer.com/article/10.1631/FITEE.1400421

55. Cherri, L. H., Cherri, A. C., and Soler, E. M. (2018). Mixed integer quadratically-constrained programming model to solve the irregular strip packing problem with continuous rotations. *J. Glob. Optim.* 72 (1), 89–107. doi:10.1007/s10898-018-0638-x. https://link.springer.com/article/10.1007/s10898-018-0638-x

56. Peralta, J., Andretta, M., and Oliveira, J. F. (2018). Solving irregular strip packing problems with free rotations using separation lines. *Pesqui. Oper.* 38 (2), 195–214. doi:10.1590/0101-7438.2018.038.02.0195. https://www.scielo.br/j/pope/a/RcXzqWKwBnL7QhcgkgNyZPv/?lang=en

57. Stoyan, Y., Pankratov, A., and Romanova, T. (2016). Cutting and packing problems for irregular objects with continuous rotations: Mathematical modelling and non-linear optimization. *J. Operational Res. Soc.* 67 (5), 786–800. doi:10.1057/jors.2015.94. https://www.tandfonline.com/doi/abs/10.1057/jors.2015.94?journalCode=tjor20

58.Wang, A., Hanselman, C. L., and Gounaris, C. E. (2018). A customized branch-and-bound approach for irregular shape nesting. *J. Glob. Optim.* 71 (4), 935–955. doi:10.1007/s10898-018-0637-y. https://link.springer.com/article/10.1007/s10898-018-0637-y

59.Guo, B., Ji, Y., Hu, J., Wu, F., and Peng, Q. (2019). Efficient free-form contour packing based on code matching strategy. *IEEE Access* 7, 57917–57926. doi:10.1109/ACCESS.2019.2914248. https://ieeexplore.ieee.org/abstract/document/8704207

60. Havrylov T.M. Model′ avtomatychnoho proektuvannya skhem rozkroyu lystovykh materialiv na detali vzuttya /T.M. Havrylov, V.I. Chuprynka //Visnyk KNUTD.- 2011, №6. – S. 83-88. https://knutd.edu.ua/files/Visnyk/Visnuk_6_2011.pdf

61.Chuprynka V.I. Metod avtomatyzovanoho proektuvannya shchil′nykh ukladok pry pryamokutno-hnizdoviy skhemi rozkroyu / V.I. Chuprynka, V.S. Murzhenko//Visnyk KNUTD.- 2011, №6. – S. 72-77. https://knutd.edu.ua/files/Visnyk/Visnuk_6_2011.pdf

62.Chuprynka V.I., Naumenko B.V., Vasylenko O.L. Heneruvannya ratsional′nykh skhem rozkroyu rulonnykh materialiv na detali shkirhalantereyi. Mekhatronni systemy: innovatsiyi ta inzhynirynh tezy dopovidey VI mizhnar. Nauk.-prakt. konf., 24 lystopada 2022 r. Kyiv: KNUTD, S. 157-158. https://er.knutd.edu.ua/bitstream/123456789/20956/1/MSIE_2022_P157-158.pdf

64. Kolysko O.Z. Modyfikatsiya henetychnoho alhorytmu dlya heneratsiyi sektsiy rozkriynykh skhem/ O.Z. Kolysko // Visnyk KNUTD. –2009.-№1. – S. 54-56. https://er.knutd.edu.ua/bitstream/123456789/6983/1/V45_P014-017.pdf

65. Кондаков А.І. САПР технологічних процесів і виробництв. ACADEMA, 2007

66. Березовський В.С., Потієнко, В.О. та Завадський, І.О., 2009. Основи комп'ютерної графіки. Київ: BHV.

67. Ванін В.В., Перевертун, В.В. та Надкернична, Т.М., 2006. Комп'ютерна інженерна графіка в середовищі AutoCAD. Київ: Каравела.

68. Веселовська Г.В. та Ходакова, В.Є., 2015. Комп'ютерна графіка. Київ: Кондор.

69. Горобець С.М., 2006. Основи комп'ютерної графіки. Київ: Центр навчальної літератури.

70. Романюк О.Н., 2001. Комп'ютерна графіка. Вінниця: Вінницький державний технічний університет.

ДОДАТОК

Лістинг програмного продукту

```
unit UnMinenko;
interface uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls,Printers,ExtCtrls,un1;

type
  TForm1 = class(TForm)
    ScrollBox1: TScrollBox;
    Image1: TImage;
    GroupBox1: TGroupBox;
    Button2: TButton;
    Button4: TButton;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    N1: TMenuItem;
    Close1: TMenuItem;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    RadioGroup2: TRadioGroup;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    ScrollBox2: TScrollBox;
    image2: TImage;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Button3: TButton;
    PrintDialog1: TPrintDialog;
    Button5: TButton;
    Image3: TImage;
    procedure Open1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Close1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image2MouseMove(Sender: TObject; Shift: TShiftState; X,
```

```
    Y: Integer);
  procedure RadioButton4Click(Sender: TObject);
  procedure RadioButton3Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}


  type
       Mas=array[1..40]of Int64;
       Mas1=array[0..200]of Int64;
       Mas2=array[0..400]of Int64;
       Uk=record
       ykk:array[0..8]of Int64;
       end;

       Rec=record
        NdS:word;
        PrS:integer;
        XpS,YpS:Int64;
        b:boolean
       end;

Const KolD=40; kf:byte=0;
    Pr_Oxr:boolean=True;
    B_Ukl:boolean=True;
 {   Pr_Btn2:boolean=False;
    Pr_Btn3:boolean=False;
    Pr_Btn4:boolean=False;}
Var    Uko:uk;
      Pr_Btn2,Pr_Btn3,Pr_Btn4:boolean;
```

```
    Fu,Ft,Fp,f:System.text{file of uk};
    FileName,FNameO,FNameT,Fname,NFile:string;
    nn,NumbSxm,KolPointL,KolPointR:array[1..KolD]of word;
    x,y,xd,yd,XDl,YDl,XDr,YDr:array[1..KolD]of Mas1;
    KolDet,KolSxm,m,h,Dl,Sh,Sh1,Kz,Most,t,DlinaN,S_Kmp,
    MaxDlDet,MaxShDet,Dx,Dy,Spar:Int64;
    NameDet:array[1..40] of string[15];
    KolPointDet,NomDet{,nn}:array[1..Kold]of word;
    KolDetSxm,KolDetSxmR:array[1..2*Kold,1..Kold]of word;
    {PrVipl:array[1..KolD]of boolean;
    Pr_Rask:array[1..2*KolD]of boolean;
    FactPl:array[1..2*KolD]of real;}
    Model:string[40];
    Wd,Hg,KolRjad,KolSt,KolRd,code,pr,l,nd,NumbDet,NumbRask,Nsxm:word;
    Xc,Yc:mas;
    Delta_X,Delta_Y,Sdet:array[1..KolD]of Int64;
    Rost,Lmax,LmaxK:array[1..KolD]of word;
    Dlina,DlinaR,Snet,Sbr,SnetR:array[1..2*KolD]of Int64;
    UklDet,UklDetM,Ykld,Pmax,Prab,K_Shish,K_ShishRab:array[1..2*KolD] of
real;
    mxy,Dxy,yk,ProcUkl:real;
    i,q,Qmax,X_k,Y_k,pv:integer;
    pq,pp:-1..1;
    B_pr,B_Zap,B_ud,B_dop,B_kor,B_Rab,PrUklOsn,PrUklPov:boolean;
    RSxm:array[1..800]of Rec;
    RsxmRab:Rec;

  Procedure MaxMin(var ne:integer; n:integer;
    var xy:array of Int64; var xye:Int64; np:integer);
      Var j:integer;
       begin
        ne:=1;
        xye:=xy[1];
        for j:=0 to n-1 do
         if xye*np<xy[j]*np then
           begin
            xye:=xy[j];
            ne:=j
           end
      end;
```

```
Procedure PolDet(n:integer; var x,y:array of Int64;
                         var xe,ye,xi,yi:Int64);
//определяет максимальное xe,ye и минимельное значение xi и yi по оси x и y

 var n2,k:integer;

Begin
  k:=1;
  MaxMin(n2,n,y,ye,k);
  MaxMin(n2,n,x,xe,k);
  k:=-1;
  MaxMin(n2,n,y,yi,k);
  MaxMin(n2,n,x,xi,k);
End;


  Procedure Scala(dln,shn:word; Var mxy:real);
  //определяет машитаб для Image1
     Var xxi,yyi,xi,yi,xxe,yye,xe,ye,dl,ds:Int64;
       mx:real;
       j:integer;
      Begin
       PolDet(KolPointDet[1],x[1],y[1],xxe,yye,xxi,yyi);
      For j:=2 to KolDet do
        begin
         PolDet(KolPointDet[j],x[j],y[j],xe,ye,xi,yi);
            if xxi>xi then xxi:=xi;
            if yyi>yi then yyi:=yi;
            if xxe<xe then xxe:=xe;
            if yye<ye then yye:=ye;
        end;
       dl:=xxe-xxi;
       ds:=yye-yyi;
       mx:=(dln-4)/dl;
       mxy:=(shn-4)/ds;
       if mxy>mx then mxy:=mx;
     End;
```

```
 Procedure Skv(n:integer; Var x,y:array of Int64; Var s:Int64);
 //определяет площадь многоугольника
  Var i:integer;
   Begin
    s:=0;
    For i:=0 to n-2 do
    s:=s+x[i]*y[i+1]-x[i+1]*y[i];
    s:=Round(abs(s)/2)
   End;

Procedure Grd(n:integer; Var xr,yr:array of Int64;
 dxc,dyc:integer; mxy:real;p:integer);
  Var i:integer;
      xp,yp:array[0..200] of integer;
     Begin
      If p>15 then
       If P mod 15=0 then P:=15
       else P:=P mod 15;
      for i:=0 to n-1 do
       begin
        xp[i]:=round(xr[i]*mxy)+dxc;
        yp[i]:=round(yr[i]*mxy)+dyc;
       end;
      With Form1.Image1,Canvas Do
      begin
         If p=1 then Pen.Color:=ClBlue
          else if p=2 then Pen.Color:=ClRed
           else if p=3 then Pen.Color:=ClMaroon
            else if p=4 then Pen.Color:=ClNavy
             else if p=5 then Pen.Color:=ClPurple
              else if p=6 then Pen.Color:=ClTeal
               else if p=7 then Pen.Color:=ClGray
                else if p=8 then Pen.Color:=ClLime
                 else if p=9 then Pen.Color:=TColor(RGB(255,128,64)){ClYellow}
                  else if p=10 then Pen.Color:=ClFuchsia
                   else if p=11 then Pen.Color:=ClAqua
                    else if p=12 then Pen.Color:=ClSilver
                     else if p=13 then Pen.Color:=ClOlive
                      else if p=14 then Pen.Color:=ClGreen
                       else Pen.Color:=ClBlack;
        MoveTo(xp[0],yp[0]);
```

```
  for i:=1 to n-1 do
          LineTo(xp[i],yp[i]);
        end;
      End;


procedure TForm1.FormCreate(Sender: TObject);
begin
 B_Ukl:=True;
 Pr_Btn2:=False;
 Pr_Btn3:=False;
 Pr_Btn4:=False;
 Width:=1220;
 Height:=850;
 Left:=0;
 Top:=0;
 Qmax:=0;
end;

procedure TForm1.Open1Click(Sender: TObject);
  Var x1,x2,y1,y2,m:integer;
     St:string[10];


  Procedure Vvod;
  Var F,Fr:System.Text;
{    FileName,}Str,NameF:string;
    xi,yi,xe,ye,Xc,Yc:Int64;
    i,j,l,m,lr,x1,y1,x2,y2:word;
    nxi,nye,nyi:integer;
    pr:byte;
    Xrab,Yrab,A:real;

begin

{   Form1.Left:=0; Form1.Top:=0;}
   if OpenDialog1.Execute then
  begin
   FileName:=OpenDialog1.FileName;
   System.Assign(F,FileName);
   L:=Length(FileName);
   NameF:='';
```

```
For i:=1 to L-3 do NameF:=NameF+FileName[i];
NameF:=NameF+'Ukl';
System.Assign(fr,NameF);
ReWrite(Fr);
System.Close(fr);
System.Erase(Fr);
Reset(F);
Readln(F,Model);
Caption:='Побудова щільних укладок для моделі  '+Model;
Readln(F,A);
Readln(F,KolDet);
For i:=1 to KolDet do
  Readln(F,NameDet[i]);
For i:=1 to KolDet do
 begin
  Readln(F,KolPointDet[i],Rost[i]);
  nn[i]:=KolPointDet[i];
 end;
For i:=1 to KolDet do
 begin
  For j:=0 to nn[i]-1 do
   begin
    readln(f,Xrab,Yrab);
    If KolPointDet[i]<>nn[i] then continue;
    if (Xrab=999)and(Yrab=999) then
     begin
      KolPointDet[i]:=j;
      continue
     end;
    x[i,j]:=Round(Xrab*100);
    y[i,j]:=Round(Yrab*100);
{    BitBtn1.Caption:=IntToStr(i)+' '+IntToStr(j);}
   end;
      nn[i]:=KolPointDet[i];
      MaxMin(nxi,KolPointDet[i],x[i],xi,-1);
      MaxMin(nxi,KolPointDet[i],y[i],yi,-1);
      MaxMin(nxi,KolPointDet[i],x[i],xe,1);
      MaxMin(nxi,KolPointDet[i],y[i],ye,1);
      Xc:=Round((xe+xi)/2);
      Yc:=Round((ye+yi)/2);
      Delta_X[i]:=Round((Xe-Xi)/2);
      Delta_Y[i]:=Round((Ye-Yi)/2);
```

```
        for j:=0 to KolPointDet[i]-1 do
         begin
          x[i,j]:=x[i,j]-Xc;
          y[i,j]:=y[i,j]-Yc;
          xd[i,j]:=x[i,j];
          yd[i,j]:=y[i,j];
         end;
        Skv(nn[i],x[i],y[i],Sdet[i]);
       end;
      end;
      MaxMin(nxi,KolDet,Delta_X,MaxDlDet,1);
      MaxMin(nxi,KolDet,Delta_Y,MaxShDet,1);

      {PrF:=True;}
 System.Close(F);
 end;
Begin
Form1.Image3.Visible:=False;
 Vvod;
//        Sh:=150000;
  //       mxy:=Image2.Height/Sh;
  //  ScrollBox1.Visible:=true;
//    GroupBox1.Visible:=true;
     RadioGroup2.Visible:=true;
     RadioButton3.Visible:=true;
     RadioButton4.Visible:=true;
     //  Label8.Visible:=true;
   { Edit4.Visible:=true;}
  X1:=5;
  X2:=50;
  B_Kor:=False;
  //Image1.Height:=55*(koldet+10)+5;
      Scala(40,50,Dxy);
  With ScrollBox1,Image1.Canvas do
    For i:=1 to koldet do
     begin
      Y1:=5+55*(i-1);
      Y2:=55*i;

       Font.Name:='Ariel';
       Font.Size:=7;
{       Font.Style:=[fsBold];,fsItalic];}
```

```
       Font.Color:=clBlack;
       St:=Copy(NameDet[i],1,8);
      TextOut(X1,Y1,st);
    Grd(KolPointDet[i],Xd[i],Yd[i],27,55*i-25,Dxy,i);
    //   Rectgl(x1,y1,x2,y2,i);
  MoveTo(X1,Y1);
  LineTo(X1,Y2);
  LineTo(X2,Y2);
  LineTo(X2,Y1);
  LineTo(X1,Y1);
 end;
end;



Procedure GrdM(n:integer; Var xr,yr:array of Int64;
 dxc,dyc,color:integer; dx,dy:Int64;pr:integer);
  Var i:integer;
     xp,yp:array[0..200] of integer;
    Begin
     for i:=0 to n-1 do
      begin
       xp[i]:=round((xr[i]*pr+dx)*mxy)+dxc;
       yp[i]:=round((dy+yr[i]*pr)*mxy)+dyc;
      end;
    With Form1.Image2,Canvas Do
     begin
        If pr=1 then Pen.Color:=ClFuchsia{ClGreen}
       else Pen.Color:=ClLime;
        Pen.Mode:=pmXOR;
        MoveTo(xp[0],yp[0]);
        for i:=1 to n-1 do
         LineTo(xp[i],yp[i]);
//          ellipse(round(dx*mxy)-5,round(dy*mxy)-
5,round(dx*mxy)+5,round(dy*mxy)+5);
       end;
    End;

procedure TForm1.Button2Click(Sender: TObject);
Var X1,Y1:Int64;
begin
ScrollBox1.Visible:=True;
```

```
ScrollBox2.Visible:=True;
 Image1.Visible:=True;
 Pr_Btn2:=True;
 Pr_Btn3:=False;
 Pr_Btn4:=False;
 If B_Dop Then
     begin
      X1:=Round(X_k/mxy);
      Y1:=Round(Y_k/mxy);
      GrdM(KolPointDet[Pv],Xd[pv],Yd[pv],0,0,pv,X1,Y1,pp);
     end;
      B_Dop:=False;
      B_Kor:=False;
end;


procedure TForm1.Button4Click(Sender: TObject);
Var X1,Y1:Int64;
    Name:string[100];
begin
 Image1.ViSible:=False;
 GroupBox2.Visible:=False;
// PrUklOsn:=True;
 Pr_Btn4:=True;
 Pr_Btn2:=False;
 Pr_Btn2:=False;
 Image2.Cursor:=crDefault;
 If B_Dop Then
     begin
      X1:=Round(X_k/mxy);
      Y1:=Round(Y_k/mxy);
      GrdM(KolPointDet[Pv],Xd[pv],Yd[pv],0,0,pv,X1,Y1,pp);
     end;
    B_Dop:=False;
    B_Kor:=False;
    SaveDialog1.execute;
name:=SaveDialog1.Filename;
 System.Assign(f,name);
 Rewrite(F);
 //writeln(f,kol);
 For i:=1 to Qmax do
  With RSxm[i] do
```

```
  If b then
    writeln(f,nds:3,'  ',xps:10,'  ',yps:10,'  ',PrS);
CloseFile(f);
    B_Ukl:=True;
      With Form1,Image2.Canvas Do
    begin
     Pen.Mode:=PmCopy;
     FloodFill(10,10,clBlack,fsBorder);
     Brush.Color:=clWhite{BtnFace};
     Brush.Style:=bsSolid;
     Rectangle(0,0,Image2.Width,Image2.Height);
    end;
    Qmax:=0;
    RadioButton3.Visible:=True;
    RadioButton4.Visible:=True;
    RadioButton3.Checked:=False;
    RadioButton4.Checked:=False;
    GroupBox1.Visible:=False;
    Button4.Visible:=False;
  end;


procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
  Var m,NomD,xm,ym,p,x1,y1,x2,y2,L1,L2,k,pv1,Xleft:integer;
   st:string[10];
   mx:real;
With Image1.Canvas do
        begin
         Pen.Color:=ClWhite;
         X1:=10;
         Y1:=55*Pv-5;
         ellipse(x1-2,y1-2,x1+2,y1+2);
         Brush.Style:=bsSolid;
         Brush.Color:=ClWhite;
         FloodFill(X1,Y1,ClWhite,fsBorder);
        end;
   If Pr_Btn2  then
    begin
     If B_Kor then
      begin
       X1:=Round(X_k/mxy);
```

```
    Y1:=Round(Y_k/mxy);
    GrdM(KolPointDet[Pv],Xd[pv],Yd[pv],0,0,pv,X1,Y1,pp);
   end;
   B_Kor:=True;
   B_Dop:=True;
   NomD:=0;
   for m:=1 to Koldet Do
    begin
     if (Y>=5+55*(m-1))and(Y<=55*m)and
      (X>=5)and(X<=50) then
      begin
       NomD:=m;
       break;
      end;
   end;
   If  B_Ukl then pv:=NomD;
   Image2.Cursor:=crNone;
With Image2 do
 begin
  mxy:=Image2.Height/Delta_Y[pv]/12;
  mx:=Image2.Width/Delta_X[pv]/12;
  IF Mxy>mx then mxy:=mx;
  x_k:=450{round(Delta_x[pv]*mxy)+3};
  y_k:=450;
  X1:=Round(X_k/mxy);
  Y1:=Round(Y_k/mxy);
  pp:=1;
 { If RadioButton3.Checked then pp:=1
  else pp:=-1; }
  GrdM(KolPointDet[Pv],Xd[pv],Yd[pv],0,0,pv,X1,Y1,pp);
 end
end;
   B_Ukl:=False;
     With Image1.Canvas do
      begin
       Pen.Color:=ClRed;
       X1:=10;
       Y1:=55*Pv-5;
       ellipse(x1-2,y1-2,x1+2,y1+2);
       Brush.Style:=bsSolid;
       Brush.Color:=ClRed;
       FloodFill(X1,Y1,ClRed,fsBorder);
```

```
        // Pr_Left:=False;
        // Pr_L:=False
        end;
End;
```