Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство
Ниво студија: Основне академске студије
Модул: Информатика у инжењерству
Предмет: Софтверски инжењеринг
Број индекса: 206/2014

Катарина С. Николић

Оптимизација сечења правоугаоне плоче помоћу троуглова

завршни рад

Комисија за преглед и одбрану:
1. **др Ненад Филиповић, ред. проф. - ментор**
2. _____
3. _____

Датум одбране:_____
Оцена:_____

У оквиру овог завршног рада кандидат треба да:

опише и имплементира неки од алгоритама који врше израчунавање оптималног броја троуглова који се могу исећи са дрвене правоугаоне плоче. Плоча је фиксних димензија док су димензије (дужина основе и висина) троугла промењљиве. Цео поступак одабира вредности димензија троугла и израчунавање треба да се обавља у *Windows* апликацији. Резултат треба да буде приказан бројчано и графички. Графички приказ треба да обухвата и интеракцију са корисником, могућност промене одговора корисника итд. Такође, кандидат треба да предвиди могуће изузетке. У оквиру дипломског рада, кандидат треба и да нацрта одговарајуће *UML* дијаграме.

Препоручена литература:

[1] Mao Chen, Wenqi Huang, „*Heuristic Algorithm for Packing Triangles into a Square Container*“, International Journal of Information and Management Sciences, 20 (2009), 255-268.
[2] Amy Chou, „*NP-Hard Triangle Packing Problems*“, January 20, 2016
[3] Hamis T. Dean, „Minimizing Waste in the 2-Dimensional Cutting Stock Problem“, Department of Mechanical Engineering, University of Canterbury, 2002

Крагујевац, 2020.

Ментор
др Ненад Филиповић, ред.проф.

Садржај

Abstract

One of the main problems in the industry is efficient packaging of smaller forms within a larger one in order to make better use of the surface. One of the areas in which this problem occurs is processing and cutting trees.

In this paper, an algorithm in the form of Windows application that describes the optimization of the utilization of a wooden board when it is cut into triangles is presented. The dimensions of the board are 400 cm x 400 cm, and the dimensions of the triangles are entered by the user.

The application calculates how many triangles of given dimensions can be obtained by cutting a given board based on the entered values, and displays the results to the user. In addition to the number of triangles, a graphical representation of the results, as well as the percentage of plate surface utilization is presented.

UML diagrams and examples of using applications were also showed.

Key Words: Wood panel cutting, Optimization Algorithm, Windows application, utilization percentage

Сажетак

Ефикасно паковање мањих облика унутар већег у циљу бољег искоришћења површине један је од основних проблема у индустрији. Једна од области у којој се јавља је и обрада, тј. сечење дрвета.

У овом раду је представњено неколико алгоритама којима се може извршити оптимизација искоришћености дрвене плоче када се иста сече на једнакостраничне троуглове, а један је имплементиран у *Windows* апликацију. Димензије плоче су 400 cm x 400 cm, а димензије троуглова уноси корисник.

На основу унетих вредности апликација израчунава колико се троуглова задатих димензија може добити сечењем дате плоче и резултате приказује кориснику. Поред броја троуглова, добија се и графички приказ резултата као и проценат искоришћености површине плоче.

У раду су приказани и UML дијаграми апликације као и примери њеног коришћења.

Кључне речи: Сечење дрвене плоче, Алгоритам за оптимизацију, *Windows* апликација, проценат искоришћености.

1. Увод

Проблем уклапања објеката се јавља у индустрији, приликом складиштења предмета, а свакако је познат и људима који се баве оригамијем. Под термином паковање мисли се на смештање истих или различитих облика, на/унутар неког другог облика.

Један од најчешће истраживаних проблема је ефикасност уклапања троуглова унутар различитих облика. У овом раду се конкретно мисли на уклапање једнакокраких троуглова на квадратну плочу тако да приликом сечења неискоришћена површина буде минимална. Такође, приликом распоређивања троуглова не сме доћи до њиховог преклапања. Како би се што ефикасније искористила плоча која се сече, потребно је искористити неки од алгоритама по коме ће се вршити сечење.

Ови алгоритми су нарочито корисни када се врши сечење дрвета. То је један од поступака у процесу обраде коме се посвећује значајна пажња. Под појмом сечење мисли се на резање употребљивих делова који се користе за склапање функционалних целина. Битан параметар који треба имати на уму је спречавање прекомерне експлоатације дрвета што значи да максимално искоришћење приликом сечења. Другим речима, искористити стабло тако да има што мање отпада.

Тема овог завршног рада је пројектовање *Windows* апликације којом се приказује један од ефикасних начина за сечење плоче. Циљ је направити максималан број једнакокраких троуглова, произвољних димензија, од плоче која је квадратног облика. Оваква апликација може се користити самостално или у склопу неке од машина за обраду дрвета и других материјала.

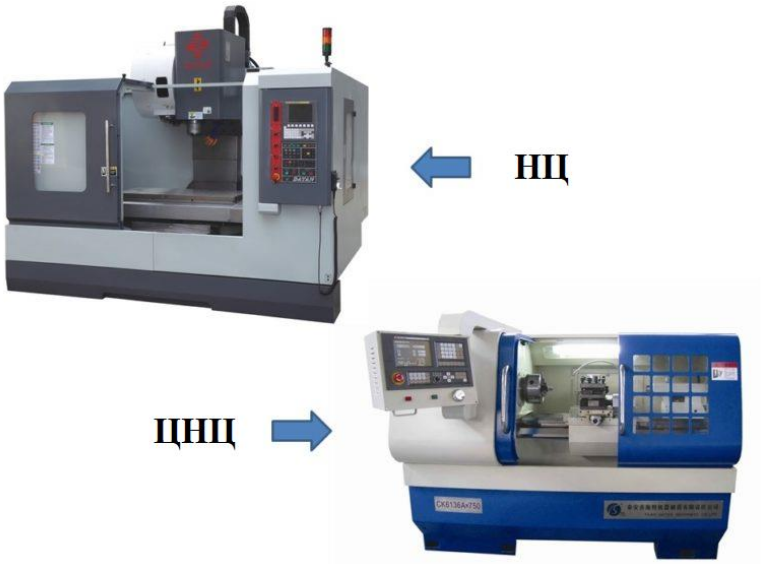
Поделу машина за обраду материјала могуће је извршити по више основа. Неке од подела су:

- према врсти производње и усвојеној технологији и машинским операцијама;
- према начину кретању између мешине и предмета обраде;
- према степену механизације и аутоматизације и сл.

Конвенционалним машинама и алатима материјали се могу обрађивати на различите начине:

- обрада резањем;
- обрада глодањем и рендисањем;
- обрада стругањем;
- обрада бушењем;
- обрада брушењем;
- обрада деформацијом.

Савремени машински системи представљају комбинацију више мањих система чиме се омогућава већи степен слободе и већа флексибилност. У ове системе спадају НЦ (*Numerical Control*) машине и ЦНЦ (*Computer Numerical Control*) машине. НЦ машине раде на основу инструкција уписаних на картицу која се убацује у машину. ЦНЦ машине су опремљене програмским рачунаром преко кога се врши контрола извршења основних и напредних функција [1]. На Слика 1 приказан је изглед по једне од обе врсте ових машина.



Слика 1. Изглед НЦ и ЦНЦ машина [2].

У другом делу рада су приказани неки од алгоритама за ефикасно уклапање троуглова и детаљније описан алгоритам који је имплементиран у саму апликацију. У трећем делу дипломског рада је приказан процес креирања апликације у *Visual Studio*-у. У овом делу су приказана делови програмског кода апликације са детаљним објашњењима. У четвртом делу су приказани UML дијаграма саме апликације, а у петом је приказан пример употребе апликације који се може посматрати као кратко корисничко упутство. У закључку су сумирани добијени резултати, тј. описане предности коришћења пројектоване апликације и дати предлози за њено даље унапређење.

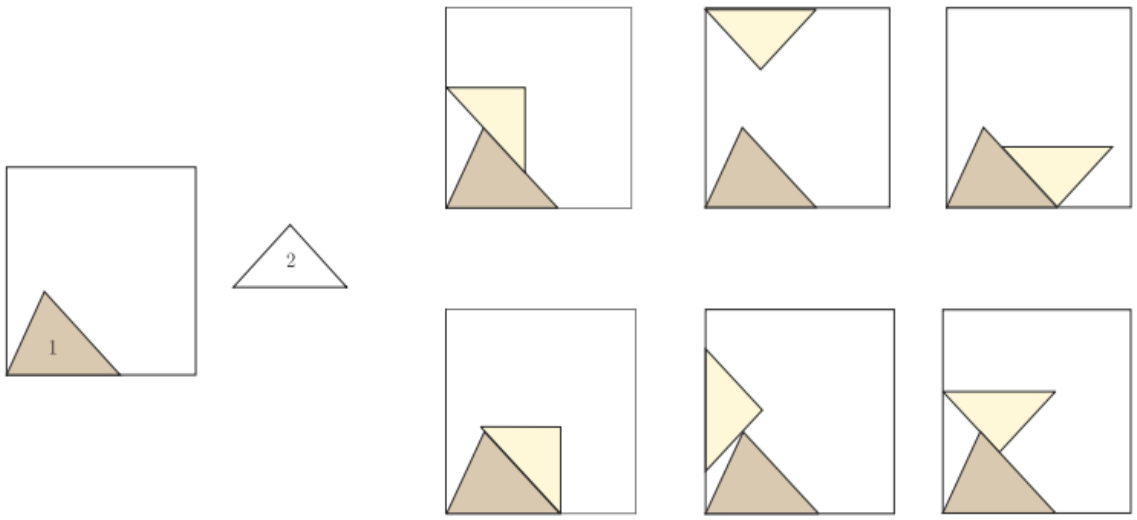
2. Оптимизација сечења површине

У наставку ће бити наведени неки од алгоритама који се могу применити за решавање поменутог проблема. Један од њих је детаљније објашњен и имплементиран у апликацију која је реализована у оквиру овог рада.

2.1 Алгоритми за оптимизацију сечења површине

Ефикасан алгоритам за решавање проблема паковања троуглова, базиран на орјентацији троуглова, предложен је у раду [3]. Према овом алгоритму, сваки троугао треба да буде постављен на положај који заузима празан угао формиран неким другим претходно постављеним троугловима. На тај начин троугао има ограничен број позиција и оријентације.

Концепт овог алгоритма је графички представљен на Слика 2.

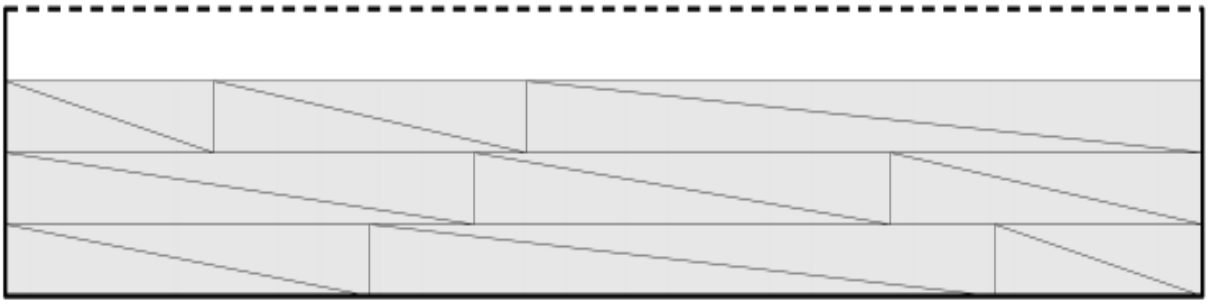


Слика 2. Графички приказ алгоритма базианог на орјентацији троуглова [3].

Троугао означен бројем 1 је први постављени троугао, а троугао означен бројем 2 је троугао који тек треба поставити. У овом илустративном примеру је приказан најкомпликованији случај (троуглови који се уклапају су различити).

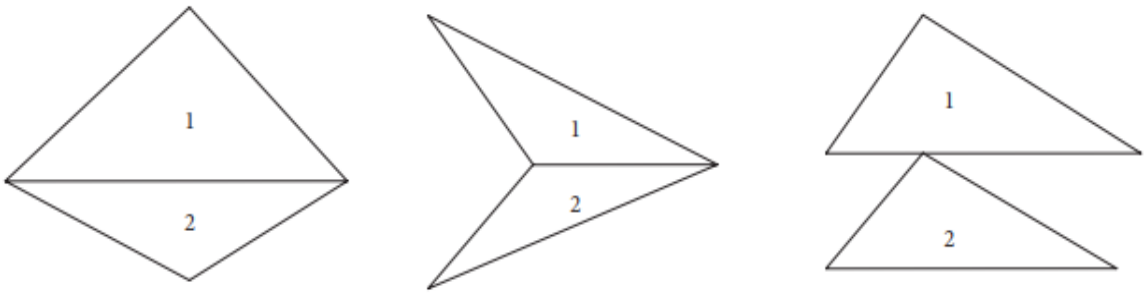
Детаљна анализа са предлогом решења овог проблема приказана је и у [4]. Уовом раду аутор је представио три случаја уклапање троуглова. У првом се разматра уклапање правоуглих троуглова унутар правоугаоника. У другом случају је објашњено уклапање правоуглих троуглова унутар правоуглог троугла, док трећи случај приказује уклапање једнакостраничних троуглова унутар једнакостраничног троугла. За овај рад је од највећег интереса први случај.

Троуглови које треба уклопити се могу поделити у три групе, тј. постоји три типа различитих троуглова. Израчунавањем се показује да је најефикасније уклапање троуглова могуће постићи, ако се подударни троуглови постављају тако да формирају правоугаоник. На Слика 3 је приказан принцип уклапања оваквих троуглова.



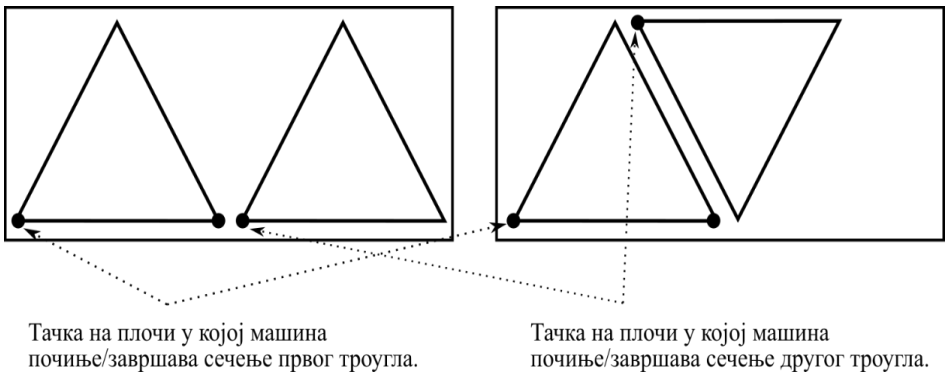
Слика 3. Паковање три типа правоуглих троуглова унутар правоугаоника [4].

Још један алгоритам за уклапање правоуглих троуглова приказан је у [5]. Овај алгоритам заснива се на формирању угаоне регије (угаоне области) између троуглова. Ако странице троуглова имају само заједничку тачку и граде угао мањи од π тако да унутар њега нема других страница онда та два троугла граде угаону регију. Формирање угаоне регије не зависи од величине и облика троугла, а у зависности од међусобне позиције троуглови могу да формирају нула, један или два угаона региона (Слика 4).



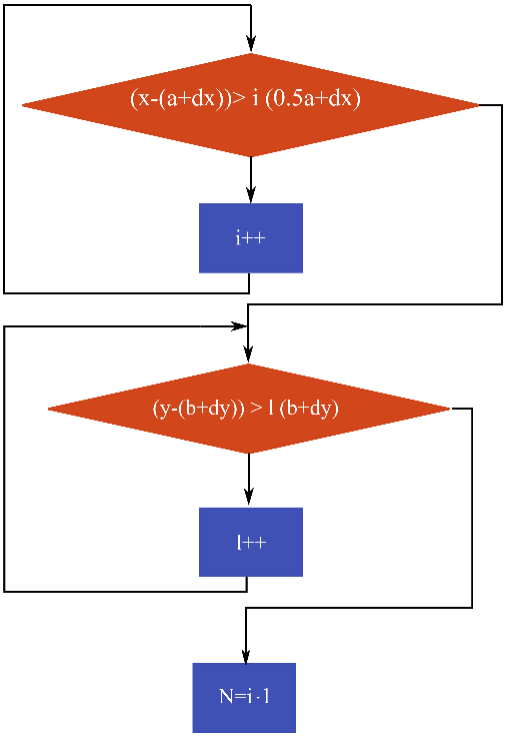
Слика 4. Нула (лево), један (у средини) и два (десно) угаона региона између троуглова [5].

Алгоритам који је детаљније анализиран и имплементиран унутар апликације може се посматрати као појединачан случај прво-приказаног алгоритма. Уклапање се врши тако што се следећи троугао поставља поред претходно постављеног заротиран за 180˚, све док се не попуни плоча (Слика 5). Приликом преласка у нови ред нема ротације троуглова. На Слика ***6*** је приказан дијаграм тока овог алгоритма.



Слика 5. Утицај положаја и орјентације облика које треба исећи на искоришћеност плоче. Лоше искоришћена плоча (лево) и добро искоришћена плоча (десно).

На приказаном алгоритму ознаке **x** и **y** означавају димензије плоче по истоименим осама, респективно. Ознаком **a** обележена је ширина, а ознаком **b** висина тоугла који треба исећи. Растојање између суседних троуглова по осама је означено са **dx** и **dy** за x-осу и y-осу, респективно. Приликом одабира ових вредности треба имати увид у механичке карактеристике машине за сечење, односно самог сечива. Ова растојања морају бити већа (у идеалном случају једнака) дебљини сечива машине. Укупан број троуглова који се може сместити на плочи, тј. исећи са плоче, по x-оси је на алгоритму обележен ознаком **i**. Исти параметар по другој, y-оци, обележен је ознаком **l**.



Слика 6. Алгоритам за израчунавање броја троуглова који се могу исећи од једне плоче.

Као што је већ речено, суседни троуглови по x-оси су ротирани један у односу на други. То омогућава да се приликом сечења искористи површина изнад страница једног троугла. Једноставније речено, у истом вертикалном прозору (део плоче који се посматра) долази до преклапања троуглова, али само по x-оси. Уколико су сви троуглови које треба добити сечењем плоче, истих домензија (као што је то случај који се разматра у овом раду) онда се на плочи, по x-оси могу смештати троуглови све док је целобројни умножак збира половине ширине троугла и растојања између суседних троуглова мањи од ширине неискоришћене плоче. Овај, наизглед компликован, прорачун приказан је првом *While* петљом на алгоритму.

Број троуглова по y-оси рачуна се истом принципу, где се параметар ширине троугла замењује висином троугла. Наравно, потребно је променити и параметре везане за осе. Једина разлика је у томе што се уместо половине користи цела висина троугла јер у овом случају нема међусобног проклапања троуглова. Овај поступак се врши другом *While* петљом на приказаном алгоритму.

Након израчунавања броја троуглова по обема осама (дужини и ширини), укупан број троуглова једнаких димензија **N** једноставно се израчунава множењем два претходно добијена параметра ($N = i \cdot l$).

3 Реализација апликације за ефикасније искоришћење површине

Ефикасно искоришћење дрвених материјала има вишеструки значај [6]. Најбитнији је свакако очување животне средине. Међутим, било би потребно уложити значајну количину енергије и времена уколико би се оваква ирачунавања вршила ручно. На срећу, увођењем рачунара у индустријским процесима овај поступак је, између осталих, умногоме олакшан.

Захваљујући *software*-ским алатима могуће је креирати апликацију (програм) којим се може реализовати тражени захтев. Конкретно, у овом раду, реч је о апликацији која је написана у програмском језику *C++*, користећи интегрисано програмско окружење *Visual Studio* 2019 [7], креирано од старане *Microsoft*-а.

Апликација треба да испуњава следеће захтеве:

1) плоча која се сече је фиксних димензија (400 cm x 400 cm);

2) од плоче треба изрезати максималан број једнакокраких троуглова;

3) димензије троуглова (дужина основе и висина) уноси корисник;

4) повратна информација коју добија корисник након што апликација заврши са рачунањем су број троуглова који је могуће исећи и графички приказ њиховог распореда на плочи;

5) упозорење уколико се за неки од параметара унесе недозвољена вредност (нпр. висина троугла 0 cm).

3.1 Учитавање *header* фајлова

Све функције које се користе у коду налазе се у постојећим библиотекама програмског окружења или се морају ручно написати. Дефиниција непостојећих функција врши се преко такозваних *header* фајлова, а укључивање већ уграђених функција врши се једноставним позивањем библиотека у којима су те функције имплементиране. На Слика 7 приказан је део кода програма којим се укључују потребни *header* фајлови и биоблиотеке.

```
#include "framework.h"
#include "WindowsProject.h"
#include <windows.h>
#include <iostream>
#include "Lib_Bitmap.h"
#include "Lib_Draw.h"
#include "Lib_GUI.h"
```

Слика 7. *Header* фајлови који се користи при реализацији кода апликације.

Header фајлови такође садрже (могу, али и не морају) и дефиниције променљивих које се користе у коду, а и сама библиотека се може учитати кроз h*eade*r фајл. Прва три фајла су стандардни системски фајлови у којима су дефинисане променљиве и укључене неке библиотеке, а испод тога је укључена и једна стандардна библиотека.

Како се преко апликације добија и графички приказ резултата неопходно је дефинисати функције којима се врши исцртавање (у овом случају правоугаоника и троуглова). Ове функције, наСлика 8, су дефинисане и имплементиране у *header* фајлу ***Lib_Draw.h***.

```
void LineToEx(int startX, int startY, int endX, int endY, int lineWidth, COLORREF color) {
    HPEN    hPen = CreatePen(PS_SOLID, lineWidth, color);
    HGDIOBJ hOldPen = SelectObject(hBufferDC, hPen);
    MoveToEx(hBufferDC, startX, startY, NULL);
    LineTo(hBufferDC, endX, endY);
    DeleteObject(hPen);
    DeleteObject(hOldPen);
}

void DrawRectangle(int x, int y, int width, int height, int lineWidth, COLORREF color) {
    LineToEx(x, y, x+width, y, lineWidth, color);
    LineToEx(x+width, y, x+width, y+height, lineWidth, color);
    LineToEx(x+width, y+height, x, y+height, lineWidth, color);
    LineToEx(x, y+height, x, y, lineWidth, color);
}

void DrawTriangle(int x, int y, int width, int height, int orientation, int lineWidth, COLORREF color) {
   LineToEx(x, y, x+width, y, lineWidth, color);
   LineToEx(x+width, y, x+(int)(width*0.5), y+orientation*height, lineWidth, color);
   LineToEx(x+(int)(width*0.5), y+orientation*height, x, y, lineWidth, color);
}
```

Слика 8. Функције којима се врши исцртавање правоугаоника (квадрата) и троугла.

У *header* фајлу ***Lib_GUI.h*** су дефинисане додатне функције неопходне за рад апликације приказане на Слика 9.

```
HWND CreateButtonEx(const char * caption, UINT type, int x, int y, int width, int height, HWND whichWindow, int id, HBITMAP icon) {
    HWND button = CreateWindowEx(WS_EX_CLIENTEDGE, "button", caption, type | WS_TABSTOP | WS_VISIBLE | WS_CHILD, x, y, width, height,
        whichWindow, (HMENU)id, GetModuleHandle(NULL), NULL);
    if (type == BS_BITMAP) {
        SendMessage(button, BM_SETIMAGE, IMAGE_BITMAP, (LPARAM)icon);
    }
    return button;
}

HWND CreateEditEx(const char* caption, int x, int y, int width, int height, HWND whichWindow, int id) {
    return CreateWindowEx(WS_EX_CLIENTEDGE, "edit", caption, WS_CHILD | WS_VISIBLE | WS_TABSTOP | WS_BORDER | ES_LEFT,
        x, y, width, height, whichWindow, (HMENU)id, GetModuleHandle(NULL), NULL);
}

int GetEditValue(HWND whichWindow, int id) {
   char value[MAX_LENGTH];
   if(GetDlgItem(whichWindow, id)) {
      GetDlgItemText(whichWindow, id, value, MAX_LENGTH);
      return atoi(value);
   }
   else {
      return 0;
   }
}
```

Слика 9. Додатне функције помоћу којих се креирају тастери, уносе вредности и исте прослеђују програму.

CreateButtonEx је функција којом се креирају тастери који се користе за прелазак са једног прозора на други, за покретање израчунавања и поновно покретање апликације након завршеног израчунавања (ресет апликације).

Функцијом ***CreateEditEx*** се креирају поља у којима корисник уноси потребне вредности (дужина основе и висина троугла). Такође, унутар једног оваквог поља се исписује вредност која се добија израчунавањем унутар апликације. Након уноса вредности од стране

корисника у одговарајућа поља, исте се прослеђују делу програма који врши израчунавање. Програм преузима ове вредности преко функције ***GetEditValue***.

3.2 Иницијализација променљивих и функција

Након учитавања *header* фајлова, следећи корак је дефинисати променљиве и функције које се користе унутар програма (Слика 10). Неке вредности су константне унутар целог програма без обзира на вредности које уноси корисник док вредности осталих променљивих могу да се мењају.

```
#define W_NAME      "Diplomski rad"
#define W_HEIGHT     650
#define W_WIDTH      650
#define W_INIT_X     100
#define W_INIT_Y     100

#define ID_TIMER_01  101

#define ID_BUTTON_01 201
#define ID_BUTTON_02 202
#define ID_BUTTON_03 203
#define ID_BUTTON_04 204

#define ID_EDIT_01   301
#define ID_EDIT_02   302
#define ID_EDIT_03   303

void LoadBitmapFiles();
void InitStartScreen();
void ClearScreen();
void GetValues();
void DrawPattern(int);
void DrawSecondScreen();
void DrawThirdScreen();

HWND hwnd;
bitmap* bProfile[10];
bitmap* iProfile[10];
int     eValue[10];
int     profileId;
float   triangleP;
bool    pClicked = false;
```

Слика 10. Дефинисање променљивих и функција.

За дефинисање вредности које се не мењају користи се директива **#define** којом се омогућава да се лабели (тексту) додели бројчана вредност, или друга лабела, која ће бити убачена у програм на сваком месту на коме је написана та лабела. На пример: На сваком

месту у програму, на коме је наведена лабела ID_BUTTON_01, програм ће је препознати као вредност 201.

На описани начин се дефинишу име апликације (**W_NAME**), димензије прозора апликације (**W_HEIGHT, W_WIDTH**), координате почетне тачке (**W_INIT_X**, **W_INIT_Y**) за исцртавање плоче задатих димензија (односи се само на визуелни приказ, али не и на параметре израчунавања). Такође, у ову групу спадају и лабеле којима се дефинишу интерни тајмер (**ID_TIMER_01**) који контролише време одзива акција унутар програма и четири тастера (од **ID_BUTTON_01** до **ID_BUTTON_04**) који се користе за покретање одговарајућих операција унутар програма. Последње три лабеле дефинисане овом директивом се односе на едитујућа поља, тј. места на коме корисник уноси параметре за израчунавање и читање израчунате вредности (од **ID_EDIT_01** до **ID_EDIT_03**).

Дефинисаним функцијама се извршавају појединачни делови програма (апликације) или се једном функцијом позива неколико других функција. Учитавање слике троугла која се користи за реализацију графичког приказа резултата користи се функција ***LoadBitmapFiles()***. При покретању апликације први приказ (изглед првог прозора) уређује се функцијом ***InitStartScreen()***. Чишћење, односно брисање изменљивог садржаја на тренутно приказаном прозору извршава се функцијом ***ClearScreen()***. Вредности који уноси корисник, а користе се за израчунавање, програм чита (преузима) из одговарајућих поља функцијом ***GetValue()***. Функција ***DrawPattern()*** се користи за исцртавање троуглова и квадрата приликом визуелног, тј. графичког приказа резултата. Пошто се апликација извршава преко три различита прозора, изглед другог и трећег прозора се реализује преко функција ***DrawSecondScreen()*** и ***DrawThirdScreen()***, респективно. Наведене функције ће детаљније бити описане у наставку рада.

У доњем делу слике 10 се могу видети променљиве различитог типа које се користе у програму, а које могу имати различите вредности у току извршавања програма. Неке од њих су површина троугла који треба исећи дефинисана као ***float triangleP***, и променљива ***bool pClicked*** која показује да ли је корисник активирао тастер за израчунавање површине тог троугла. Ово су променљиве које се могу користити у било ком делу програма, док се унутар сваке од наведених функција налазе додатне променљиве које се могу користити само у оквиру функције у којој су наведене.

3.3 Функција *LoadBitmapFiles()*

Како би употреба апликације била једноставнија, параметри које уноси корисник, као и они који се добијају након израчунавања се приказују и визуелно. Тачније, након покретања апликације (биће приказано касније) кориснику се приказује изглед плоче од које се секу троуглови.

На следећем прозору се приказује изглед троугла са означеним димензијама које уноси корисник, а на трећем прозору се унутар плоче исцртавају троуглови на начин на који ће бити сечени.

Како би било могуће нацртати све геометријске фигуре (у овом случају квадрат и троугао) потребно је у програм увести одговарајуће фајлове са екстензијом **.bmp**. На тај

начин се фигуре реализују тако што се унутар апликације приказују импортоване слике у одговарајућој размери. На Слика 11 је приказан код који се извршава унутар ове функције.

```
void LoadBitmapFiles() {
    bProfile[0] = new bitmap("triangle.bmp", 96, 96);
    iProfile[0] = new bitmap("triangleConstraints.bmp", 450, 450);
    iProfile[1] = new bitmap("rectangleConstraints.bmp", 450, 450);
}
```

Слика 11. Учитавање фајлова унутар функције *LoadBitmapFiles()*.

3.4 Функција *InitStartScreen()*

Оно што корисник види непосредно након покретања апликације реализовано је функцијом *InitStartScreen()*, чија имплементација је приказана наСлика 12.

```
void InitStartScreen() {
    BeginPaintEx(hwnd);
    DrawObject(iProfile[1], 0, 0);
    DrawTextEx2("a = ", 450, 65, RGB(0, 0, 0));
    DrawTextEx2("400 cm", 490, 65, RGB(0, 0, 0));
    DrawTextEx2("b = ", 450, 115, RGB(0, 0, 0));
    DrawTextEx2("400 cm", 490, 115, RGB(0, 0, 0));
    DrawTextEx2("P = ", 450, 165, RGB(0, 0, 0));
    DrawTextEx2("160000 cm²", 490, 165, RGB(0, 0, 0));
    DrawTextEx2("Opis zadatka:", 50, 400, RGB(0, 0, 0));
    DrawTextEx2("Imamo plocu datih dimenzija, potrebno je iscrtati", 50, 450, RGB(0, 0, 0));
    DrawTextEx2("maksimalan moguci broj trouglova na datoj ploci,", 50, 475, RGB(0, 0, 0));
    DrawTextEx2("cije se dimenzije mogu izabrati na sledecem ekranu,", 50, 500, RGB(0, 0, 0));
    DrawTextEx2("klikom na dugme Next.", 50, 525, RGB(0, 0, 0));
    EndPaintEx(hwnd);
    CreateButtonEx("Next", BS_TEXT, 495, 550, 125, 50, hwnd, ID_BUTTON_01, NULL);
}
```

Слика 12. Програмски код функције *InitStartScreen()*.

На почетку се позива функција ***DrawObject()*** која се користи за цртање. На овом месту се црта правоугаоник тако што се функцији, као аргумент, предаје променљива iProfile[1] преко које је раније учитан фајл у коме је дефинисан квадрат.

Највећи број елемената на првом прозору апликације је реализовано као текст. За то је искоришћена функција ***DrawTekstEx2()***. На прозору треба исписати колике су димензије плоче која се сече и колика је њена површина. Такође ту се исписује и кратко упутство како би рад корисницима који први пут користе апликацију био олакшан.

На крају, генерише се тастер који има улогу да прикаже следећи прозор апликације при активацији од стране корисника. Као аргумент за сваку од наведених функција могуће је навести координате тачке у којој ће почети исцртавање фигура, односно исписивање текста. На тај начин је могуће уредити изглед појединачног прозора апликације.

3.5 Функција *ClearScreen()*

Као што је речено у претходном делу рада, приликом коришћења апликације смењује се неколико прозора. Апликација је тако организована ради прегледности и једноставнијег коришћења. Прегледност се постиже тиме што на једном прозору постоје подаци који су повезани само са тренутним активностима које корисник извршава (унос димензија троугла и израчунавање његове површине на пример). Ово уједно и доприноси једноставности коришћења апликације у смислу да се смањује могућност корисничких грешака. Наиме, уколико би на једном прозору било реализовано неколико различитих функција, могућност да се податак унесе у погрешно поље би била већа.

Између преласка са једног прозора на други активира се функција која се не види са корисничке стране. Њена улога је да очисти садржај са претходног прозора и тиме отклони потенцијалне нелогичности и нејасноће у графичком интерфејсу. Та функција је *ClearScreen()*, а програмски код њене имплементације приказан је наСлика 13.

```
void ClearScreen() {
    HWND hTempWindow;
    for (int i = 0; i < 8; i++) {
        hTempWindow = GetDlgItem(hwnd, i + ID_BUTTON_01);
        if (hTempWindow) {
            DestroyWindow(hTempWindow);
        }
        hTempWindow = GetDlgItem(hwnd, i + ID_EDIT_01);
        if (hTempWindow) {
            DestroyWindow(hTempWindow);
        }
    }
}
```

Слика 13. Програмски код функције *ClearScreen()*.

Конкретно, у овом случају, када се ова функција искористи све вредности које зависе од стања тастера, као и сама стања тих тастера се постављају на подразумевану вредност. Такође, вредности које је корисник у међувремену уписивао у поља чије вредности могу да се мењају, бришу се.

3.6 Функција *GetValue*()

Гледано са стране обимности и функционалности програмског кода може се рећи да је ово једна од најједностављавнијих, ако не и најједноставнија функција која се користи у главном коду. На Слика **14** је приказана њена имплементација.

```
void GetValues() {
    int c;
    eValue[0] = 400;
    eValue[1] = 400;
    eValue[2] = GetEditValue(hwnd, ID_EDIT_01);
    eValue[3] = GetEditValue(hwnd, ID_EDIT_02);
}
```

Слика 14. Програмски код функције *GetValue()*.

Јасно је да ова функција на неки начин представља надоградњу функције *GetEditValue()*. Њоме се фиксне (димензије плоче која се сече) и променљиве (димензије троугла које уноси корисник) вредности предају главном делу програма у коме се врши израчунавање. Веза између функције *GetEditValue()* и ове функције је у томе што прва преузима вредност из едитујућег поља и уписује у променљиву, а друга вредност и променљиве предаје главном програму.

3.7 Функција *DrawPattern()*

У оквиру ове функције се врши израчунавање броја троуглова задатих димензија који се могу добити сечењем дате плоче. Врши се исцртавање троглова на на самој плочи, а резултат израчунаваља се исписује у облику текстуалне поруке који корисник може да прочита.

Израчунавање се врши на основу алгоритма који је раније описан и приказан на Слика 6, тако да на овом месту неће бити детаљније објашњења самог програмског кода који је приказан на Слика 15.

Порука која се приказује кориснику састоји се из два дела. Први део поруке представља вредност броја троуглова задатих димензија који се могу исећи са дате плоче, и ова вредност одговара броју нацртаних троуглова на плочи (графички део приказа резутата). Други део поруке садржи вредност коју указује на ефикасност искоришћења дате површине при датом сечењу. Искоришћење дате плоче, изражена у процентима се може израчунава на основу следеће једначине:

$$\text{Искоришћеност плоче [\%]} = \frac{\text{Број троуглова} \times \text{Површина једног троугла}}{\text{Површина плоче}} \times 100\ [\%]$$

```
void DrawPattern() {
    int i, l, x, y, dx, dy;
    CreateButtonEx("Reset", BS_TEXT, 495, 550, 125, 50, hwnd, ID_BUTTON_02, NULL);
    BeginPaintEx(hwnd);
    DrawRectangle(25, 25, eValue[0], eValue[1], 2, RGB(0, 0, 0));
    dx = (int)(eValue[2] * 0.20);
    dy = (int)(eValue[3] * 0.20);
    i = 0;
    while (eValue[0] - (eValue[2] + dx) > i * ((eValue[2] * 0.50) + dx)) {
        i++;
    }
    l = 0;

    while (eValue[1] - (eValue[3] + dy) > l * (eValue[3] + dy)) {
        l++;
    }
    int trianglesNum = i * l;
    y = dy + 25;
    while (l > 0) {
        int c = i, sign = 1, tempY = y;
        x = dx + 25;
        while (c > 0) {
            DrawTriangle(x, tempY, eValue[2], eValue[3], sign, 2, RGB(0, 0, 0));
            x += dx + (int)(eValue[2] * 0.50);
            tempY += (int)(eValue[3] * sign);
            sign *= -1;
            c--;
        }
        y += dy + eValue[3];
        l--;
    }
    DrawTextEx2("Nakon preracunavanja dobijeni su sledeci podaci:", 25, 450, RGB(0, 0, 0));
    DrawTextEx2("Broj trouglova:", 25, 500, RGB(0, 0, 0));
    char char_arr[100];
    sprintf_s(char_arr, "%.02f", (float)trianglesNum);
    DrawTextEx2(char_arr, 175, 500, RGB(0, 0, 0));
    DrawTextEx2("Iskoriscenost povrsine (%):", 25, 525, RGB(0, 0, 0));
    float efficiencyPercentage = (float)((trianglesNum * triangleP) / 160000) * 100;
    sprintf_s(char_arr, "%.02f", efficiencyPercentage);
    DrawTextEx2(char_arr, 300, 525, RGB(0, 0, 0));
    EndPaintEx(hwnd);
}
```

Слика 15. Програмски код функције *DrawPattern()*.

3.8 Функције *DrawSecondScreen()* и *DrawThirdScreen()*

Програмски код ових функција приказан је на Слика 16, а њихова функција је како сам назив каже цртање (приказ) другог и трећег прозора апликације.

```
void DrawSecondScreen() {
    BeginPaintEx(hwnd);
    DrawObject(iProfile[0], 0, 50);
    DrawTextEx2("Unesite sirinu i visinu trougla", 50, 35, RGB(0, 0, 0));
    DrawTextEx2("W:", 450, 110, RGB(0, 0, 0));
    CreateEditEx("", 490, 115, 75, 20, hwnd, ID_EDIT_01);
    DrawTextEx2("cm", 575, 110, RGB(0, 0, 0));
    DrawTextEx2("H:", 450, 160, RGB(0, 0, 0));
    CreateEditEx("", 490, 165, 75, 20, hwnd, ID_EDIT_02);
    DrawTextEx2("cm", 575, 160, RGB(0, 0, 0));
    CreateButtonEx("P", BS_TEXT, 470, 215, 75, 30, hwnd, ID_BUTTON_03, NULL);
    CreateEditEx("", 450, 265, 120, 20, hwnd, ID_EDIT_03);
    DrawTextEx2("cm²", 575, 260, RGB(0, 0, 0));
    EndPaintEx(hwnd);
    CreateButtonEx("Next", BS_TEXT, 495, 550, 125, 50, hwnd, ID_BUTTON_04, NULL);
}

void DrawThirdScreen() {
    GetValues();
    ClearScreen();
    DrawPattern();
}
```

Слика 16. Програмски кодови функција *DrawSecondScreen()* и *DrawThirdScreen()*.

На другом прозору апликације се приказује троугао са означеним параметрима које корисник треба да унесе, као и едитујућа поља у која се уносе одговарајући параметри. У овом случају се ради о једнакокраким троугловима па су параметри које уноси корисник дужина основе и висина троугла.

У оквиру трећег, уједно и последњег, прозора се извршавају три операције (функције). Вредности које је унео корисник се прослеђују делу програма у коме се врши израчунавање. Затим се бришу непотребне информације које остају са претходног прозора и на крају се врши израчунавање и приказ резултат што је заправо и главни део који је кориснику од интереса.

3.9. Корекција погрешно унетих вредности од стране корисника

Као и у било ком другом индустријском процесу, реално је очекивати да у се у неком тренутку деси грешка и дође до непланираног застоја. Аутоматизацијом је та могућност сведена на минимум, али она и даље постоји. У овом случају се не може рећи да је процес потпуно аутоматизован јер је неопходна улога корисника (оператера) који уноси вредности о димензијама троугла. Управо на том месту се могу десити одређене грешке.

Једна од њих је да корисник не унесе вредности о ширини (дужина основице једнакокраког троугла) или о висини троугла пре него што активира тастер којим се израчунава његова површина. Такође, грешком се сматра и ако се за неку од ове две вредности упише нула.

Друга грешка која може да се јави при раду је да корисник активира тастер за прелазак на прозор у коме се израчунава и графички представља број троуглова који се може исећи са плоче, а да пре тога није активирао тастер за израчунавање површине троугла.

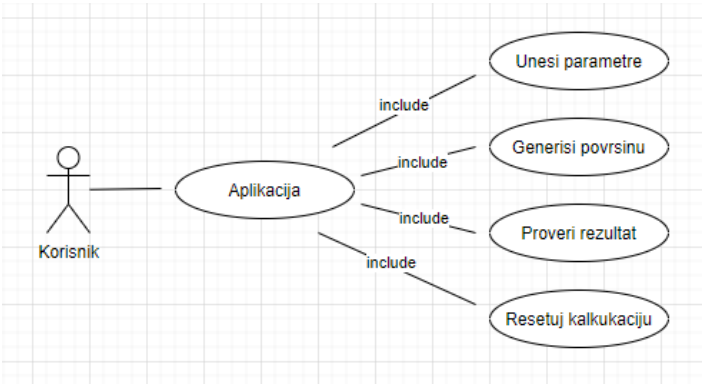
Како би се овакве грешке избегле, а самим тим и поновно покретање апликације и унос вредности уколико креирана је посебна процедура која контролише рад тастера (Слика 17). Уколико се деси један од два наведена сценарија не прелази се на следећи прозор апликације већ се исписује одговарајуће упозорење које је дефинисано кодом, на основу кога корисник може да исправи своје грешке.

```
case ID_BUTTON_04:
{
    if (GetEditValue(hwnd, ID_EDIT_01) == 0)
        MessageBox(hwnd, "Molimo vas da unesete vrednost za sirinu trougla.", "Upozorenje", MB_OK);
    else if(GetEditValue(hwnd, ID_EDIT_02) == 0)
        MessageBox(hwnd, "Molimo vas da unesete vrednost za visinu trougla.", "Upozorenje", MB_OK);
    else if(!pClicked)
        MessageBox(hwnd, "Molimo vas da klikom na dugme P preracunate povrsinu trougla.", "Upozorenje", MB_OK);
    else
        DrawThirdScreen();
} break;
```

Слика 17. Део програмског кода процедуре којом се контролише извршавање акција активацијом тастера.

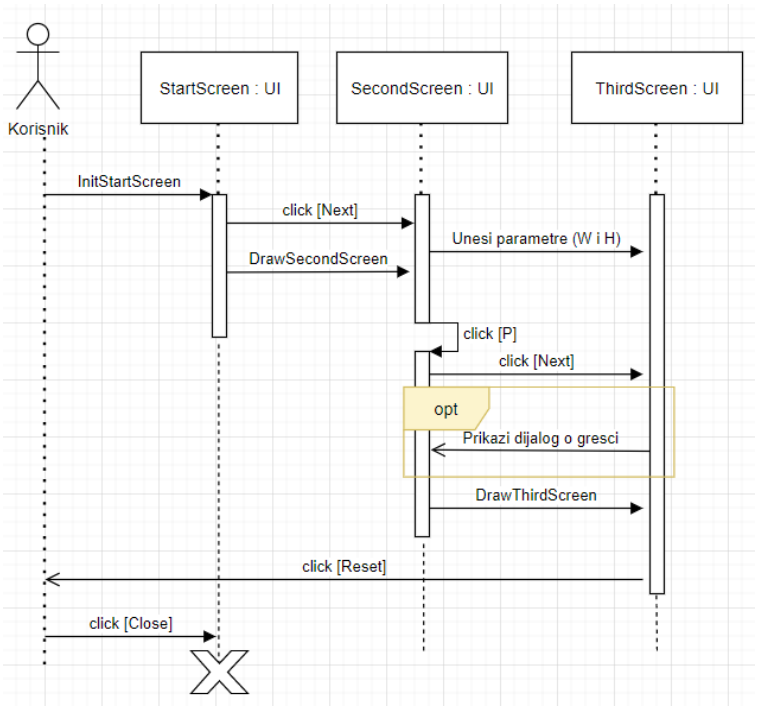
4. UML дијаграм пројектоване апликације

UML је акроним од назива *Unified Modeling Language* којим се означава скуп графичких нотација које се користе приликом описа и дизајна *software*-ских система [8]. Постоји неколико врсти ових дијаграма који се у пракси. Апликација која се описује у овом раду може се описати преко дијаграма који су приказани у наставку. На Слика 18 је приказан ***Use case*** дијаграм.



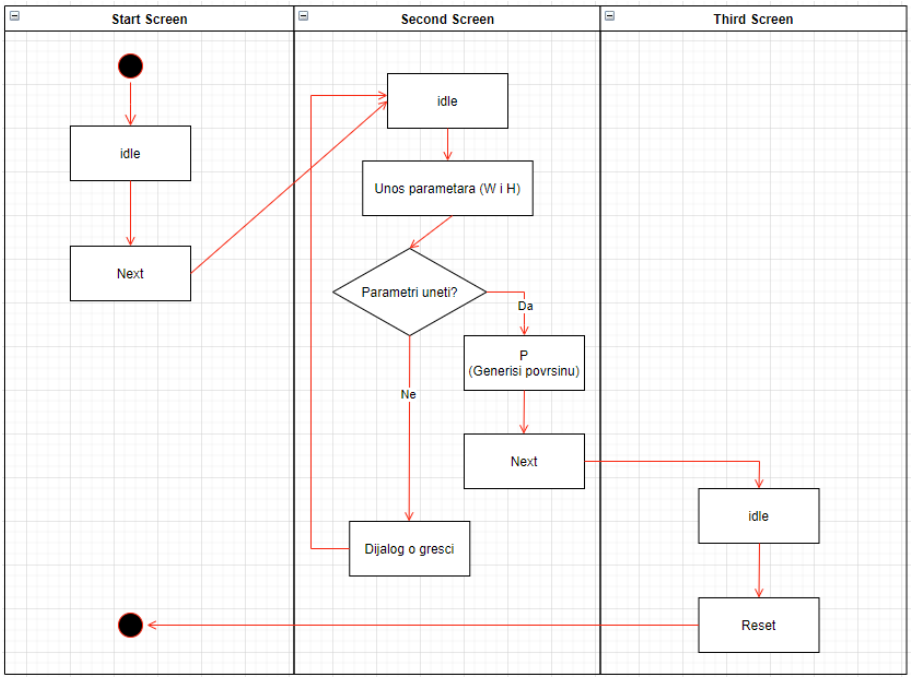
Слика 18. *UML Use case* дијаграм.

Дијаграм употребе случаја је најједноставнији облик приказа корисничке интеракције са апликацијом. На Слика 19 је приказан је дијаграм секвенци којим се приказује временски распоред активности унутар ње.



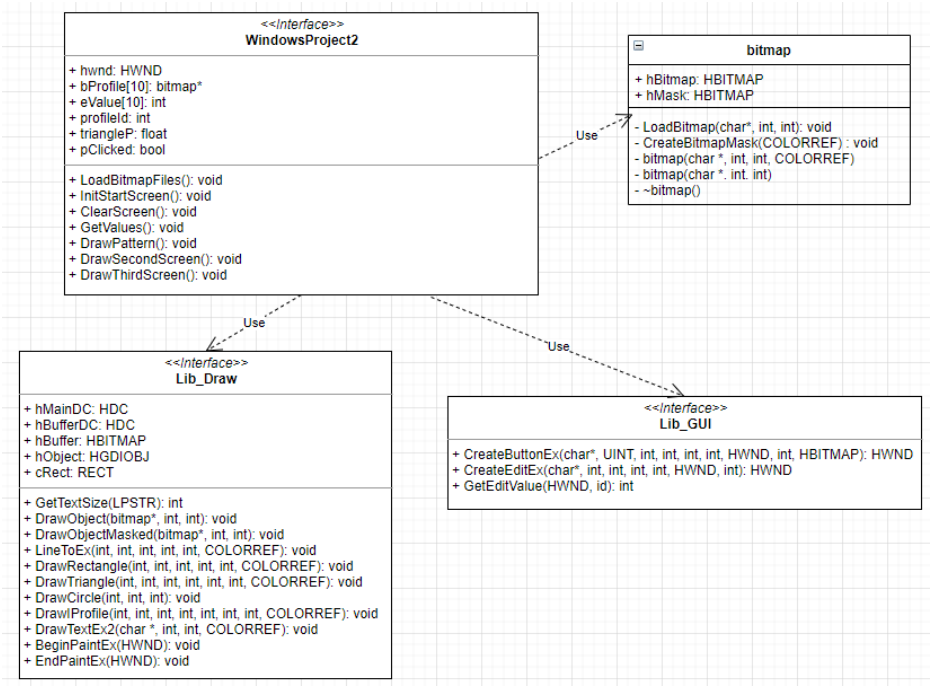
Слика 19. *UML* дијаграм секвенци.

Дијаграм активности приказан је на Слика 20. Он обухвата кораке и ток активности са приказом избора, итерације и временске зависности (да ли се активности дешавају истовремено или не).



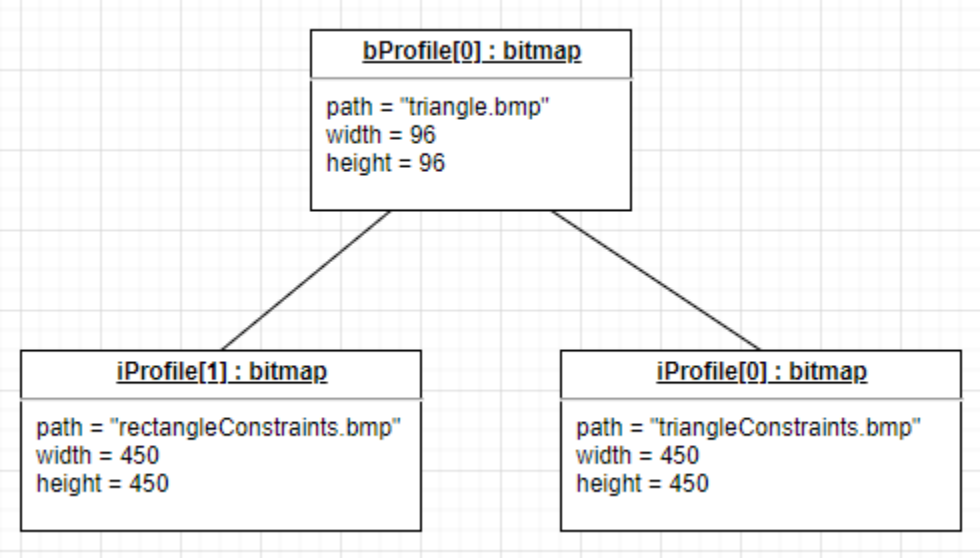
Слика 20. *UML* дијаграм активности.

Дијаграмом класа се описује структура система, приказујући његове класе, њихове атрибуте, операције и односе између објеката (Слика 21).



Слика 21. *UML* дијаграм класа.

На Слика 22 је представљен дијаграм објеката. Њиме се приказује делимичан или потпун изглед дизајниране структуре (у овом случају се мисли на апликацију) у датом тренутку.



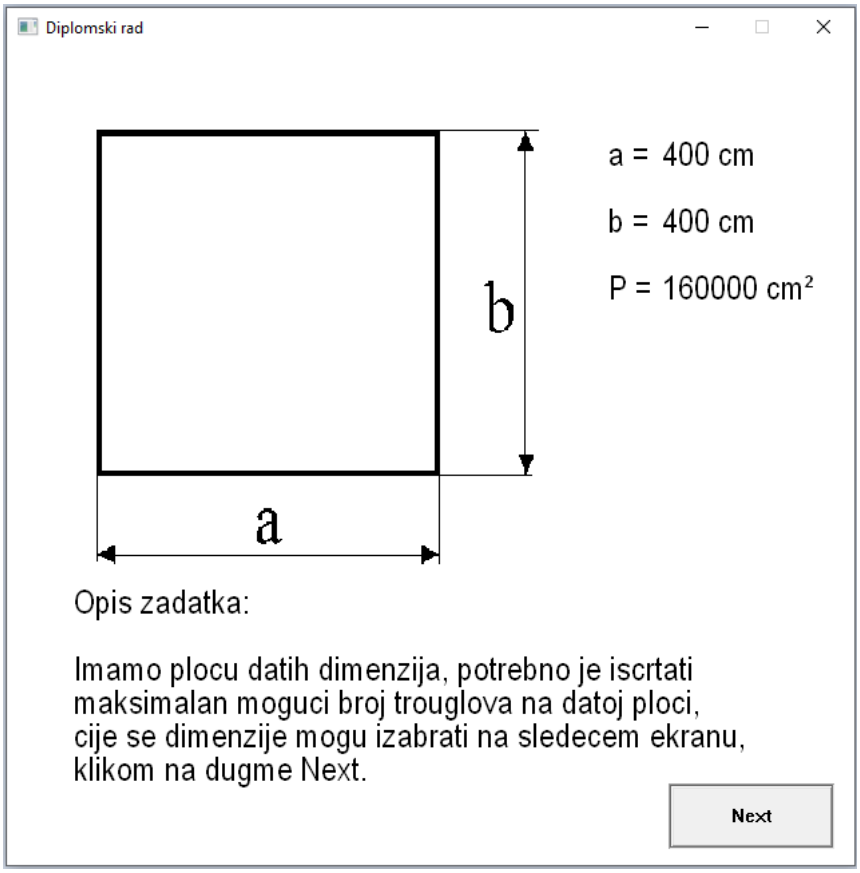
Слика 22. *UML* Дијаграм објеката.

5. Пример употребе апликације

У овом делу биће представљен један пример употребе апликације са коректно унетим вредностима, као и један пример када то није случај. Оба примера су праћена одговарајућим илустрацијама.

5.1 Израчунавање броја троуглова са коректно унетим вредностима

Након покретања апликације кориснику се приказује почетни екран, или како је раније у тексту наведено први прозор. Изгледа тог прозора приказан је на Слика 23.

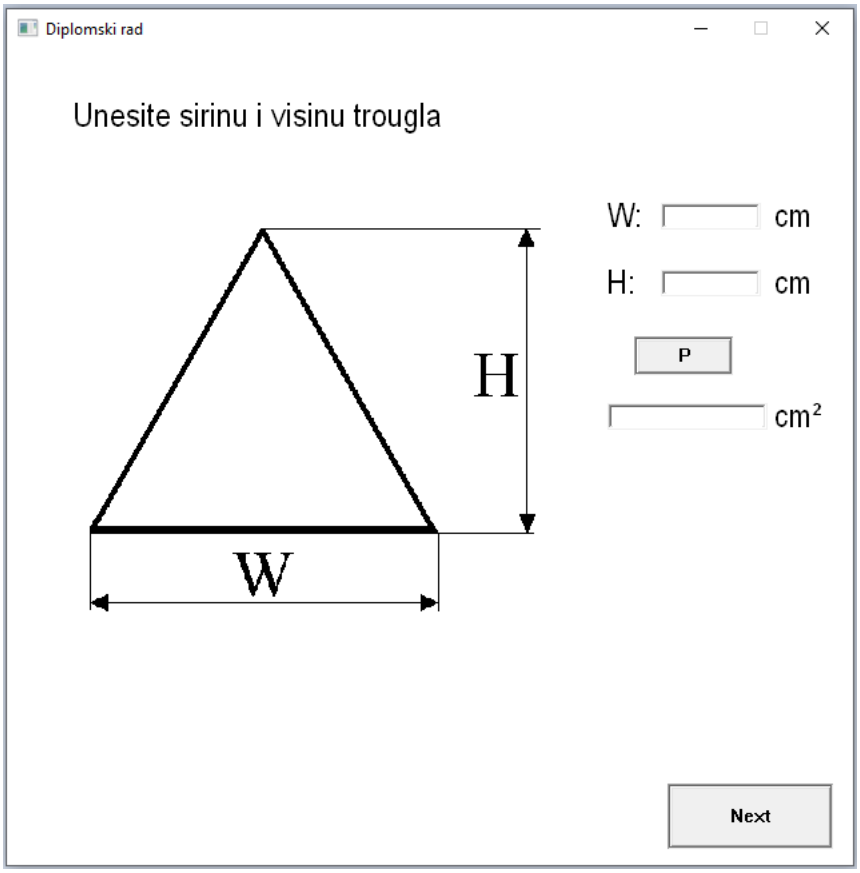


Слика 23. Изглед првог прозора апликације.

На њему се приказује да се ради са квадратном плочом (димензије обе странице плоче су једнаке) и приказује се њена површина. Испод графичког приказа облика плоче укратко је описан задатак саме апликације. У оквиру тог описа нализи се и кратко упутство за наставак рада са апликацијом.

Активацијом (кликом) тастера *Next* прелази се на следећи (други) прозор који је приказан на Слика 24. На овом прозору се приказује слика троугла који треба исећи. Пошто је реч о једнакостраничном троуглу на слици су обележене ширина (дужина основице) и висина тог троугла. У празним пољима (поља у коду чија вредност може да се мења) означеним

словима W и H уписује се ширина (дужина основице) и висина троугла изражене у центиметрима, респективно. Мали број параметара који је неопходно да унесе корисник је главна одлика једноставности коришћења ове апликације.

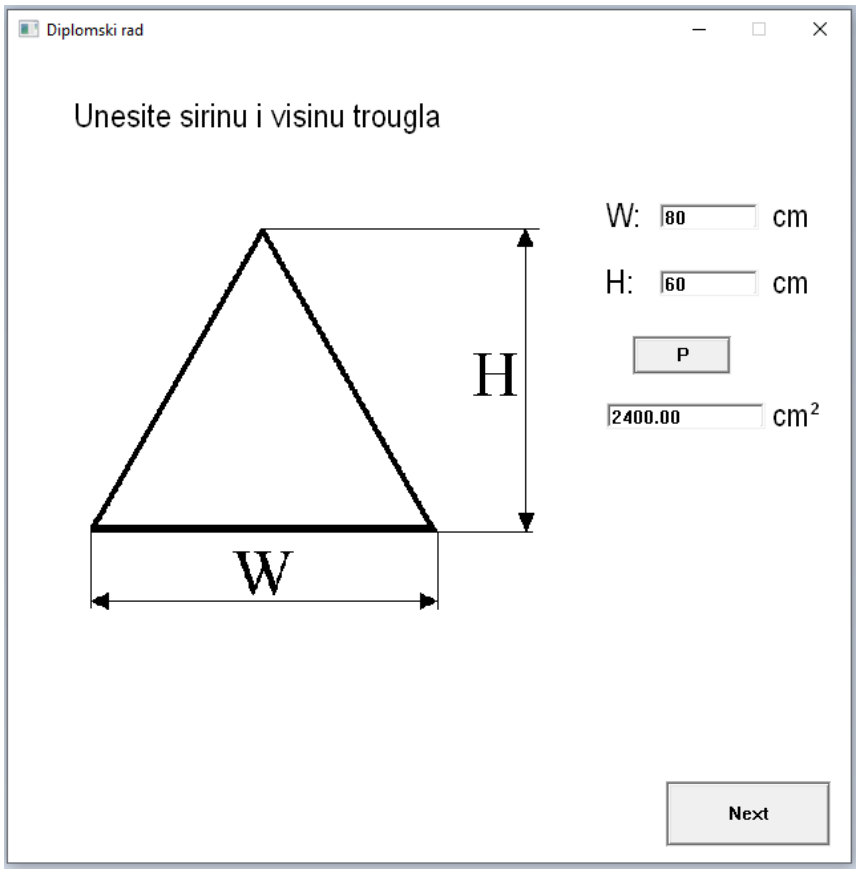


Слика 24. Изглед другог прозора апликације.

Следећи корак је израчунавање површине троугла на основу уписаних вредности. Кликом на тастер обележен словом P добија се вредност површине троугла изражена у квадратним центиметрима.

За потребе овог илустративног примера изабрано је да вредности које се уносе буду: 80 cm за ширину (дужину основе) троугла и 60 cm за висуну троугла. На основу тих вредности очекује се да је површина троугла 2400 cm^2. Управо је то вредност која је добијена и у апликацији (Слика 25).

Завршетак рада на овом прозору врши се кликом на тастер *Next* чиме се прелази на трећи уједно и последњи прозор у апликацији приказан на Слика 26.



Слика 25. Изглед другог прозора апликације након уноса димензија троугла и израчунавања његове површине.

За корисника је свакако трећи прозор најбитнији. Главна информацију коју је потребно да добије (број троуглова који се може исећи са дате плоче) из целе апликације налази се на овом прозору.

У овој апликацији је имплементиран само један алгоритам за израчунавање оптималног броја троуглова који се могу исећи са плоче. Не може се рећи да је то максималан број троуглова зато што није имплементиран ниједан други алгоритам да би било могуће извршити поређење. Без обзира на то корисник може да, на основу добијене вредности, стекне увид о ефикасности искоришћења површине плоче уколико са она сече на троуглове задатих димензија.

Провере ради, искоришћеност плоче на основу унетих вредности би требало да буде:

$$\frac{30\ x\ 2400}{160000}\ x\ 100\ \% =\ 0.45\ x\ 100\ \% = 45\ \%,$$

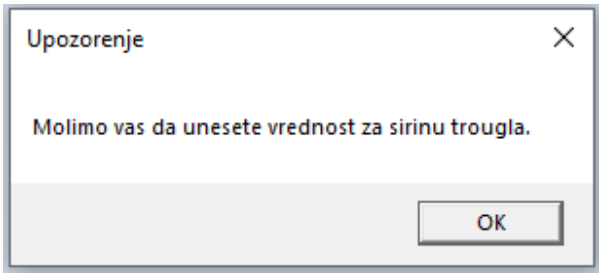
што и јесте вредност добијена у апликацији.

Слика 26. Изглед трећег прозора апликације.

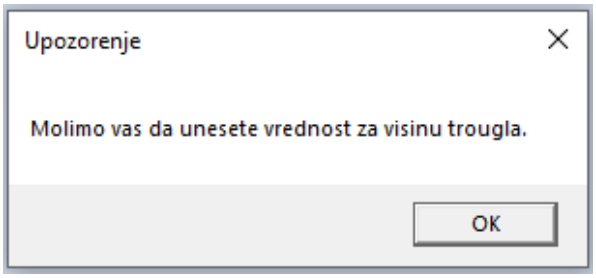
5.2 Израчунавање броја троуглова са погрешно унетим вредностима

У претходном поглављу су описана два сценарија до којих може доћи уколико се догоди пропуст од стране оператера. Још једном треба нагласити да се грешка у раду апликацији може доготити само на месту на коме се уносе димензије троугла.

Подсећања ради, једна грешка је недостатак димензије троугла, а други покушај приказа резултата без претходног израчунавање површине троугла. На Слика 27 и Слика 28 су приказана упозорења која корисник добија у случају да не унесе неку од вредности димензија троугла, дужину основе или висину.



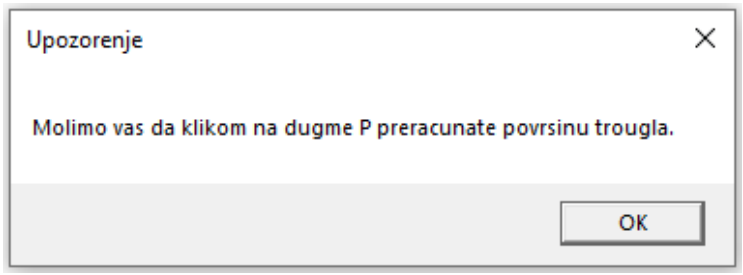
Слика 27. Упозорење да није унета вредност за ширину (дужину основе) троугла.



Слика 28. Упозорење да није унета вредност за висину троугла.

У самом коду апликације, подразумеване вредности променљивих у које се смештају вредности које се уносе су једнаке нули. Због тога је рачунање површине могуће и када корисник не унесе ове вредности, а резултат, тј. вредност површине, ће наравно бити једнак нули. Међутим, у том случају ће се приказати једно од два приказана упозорења када корисник покуша да активира прозор апликације на коме се приказују резултати.

Када корисник активира тастер за приказ резултата израчунавања, а да претходно површина троугла није израчуната, приказаће му се упозорење као на Слика 29.



Слика 29. Упозорење да површина није израчуната.

Уколико дође до пропуста и приказивања неког од упозорења рад са апликацијом се може наставити поступањем по упутству које је у њему наведено. Ова чињеница омогућава да се настави са радом без потребе за поновним покретањем апликације.

6. Закључак

Вишеструка примена дрвета и све јача оријентација прерађивача на ужи асортиман и масовну производњу, довеле су до великог броја различитих специјалних конструкција машина за обраду дрвета, намењених изради одређеног финалног производа.

При механичкој преради дрвета јављају се отпаци у облику технички неупотребљивих комада и пиљевине. Ови отпаци су се раније сагоревали или уништавали на други начин. Данас се то више не чини, јер је дрво скуп материјал. Отпаци се користе за производњу више дрвних полупроизвода.

Апликација приказана у овом раду има за циљ да олакша израчунавање искоришћености површине, односно проценат отпада при сечењу плоче квадратног облика на троуглове. У приказаној верзији апликације плоча која се сече је фиксне дужине, а сечењем се добијају једнакостранични троуглови. За израчунавање броја троуглова који се могу исећи искоришћен је алгоритам по коме су суседни троуглови орјентисани супротно један у односу на други.

Предност коришћења ове апликације повећање брзине израчунавања искоришћености површине у односу на ручно израчунавање истог параметра. Друга битна карактеристика је широка могућност унапређења саме апликације по неколико особина.

Једноставно се може реализовати да плоча која се сече буде произвољних димензија, у складу са захтевима производње. Такође, може се реализовати да троуглови који се секу не буду нужно једнакокраки. Као значајно побољшање може се увести да се плоча не сече само на троуглове, већ и на друге геометрисјке облике. Што се тиче истраживачког дела, имплементација још једног или више алгоритама за израчунавање оптималног броја троуглова представљала би побољшање апликације. На тај начин би било могуће поредити ефикасност више алгоритама по којима се сече плоча.

Литература

[1] *Website*: https://www.ganeshprecision.com/blog/difference–nc–cnc–machine/#:~:text=NC%20stands%20for%20Numerical%20Control,and%20symbols%20called%20a%20program. Приступљено: 14.06.2020. год.

[2] *Website*: https://www.mechanicalbooster.com/2016/12/difference-between-nc-and-cnc-machine.html Приступљено: 14.06.2020. год.

[3] Mao Chen, Wenqi Huang, „*Heuristic Algorithm for Packing Triangles into a Square Container*“, International Journal of Information and Management Sciences, 20 (2009), 255-268.

[4] Amy Chou, „*NP-Hard Triangle Packing Problems*“, January 20, 2016

[5] Wei, Guoliang, Wang, Ruimin, Luo, Yuqiang, Dong, Jianqiang, Liu, Shuai, Qi, Xiaozhuo, „*A Heuristic Algorithm for Solving Triangle Packing Problem*“, Discrete Dynamics in Nature and Society, Hindawi Publishing Corporation, 2013/12/19, doi.org/10.1155/2013/686845.

[6] Hamis T. Dean, „Minimizing Waste in the 2-Dimensional Cutting Stock Problem“, Department of Mechanical Engineering, University of Canterbury, 2002.

[7] *Website*: https://visualstudio.microsoft.com/ Приступљено: 25.06.2020. год.

[8] *Website*: https: // www.ucim - programiranje.com / 2012 / 12 / uml-osnove / Приступљено: 29.06.2020. год.