WYŻSZA SZKOŁA INŻYNIERYJNO – EKONOMICZNA
z siedzibą w Rzeszowie
WYDZIAŁ PRZEDSIĘBIORCZOŚCI

Kierunek: GEODEZJA I KARTOGRAFIA
Specjalność: GEOINFORMATYKA

PRACA INŻYNIERSKA

PIOTR MICHOŃ

Stworzenie aplikacji do obliczeń geodezyjnych na urządzenia z systemem Android.

Praca inżynierska
wykonana pod kierunkiem:
dr inż. Jacka Derwisza

Oświadczam, że niniejszą pracę wykonałem samodzielnie

data i podpis studenta

Przyjmuję pracę jako inżynierską

data i podpis promotora

Rzeszów 2014

Imię i nazwisko Autora Pracy	Piotr Michoń
Imię i nazwisko Promotora pracy	dr inż. Jacek Derwisz
Wydział / kierunek studiów	Przedsiębiorczości / Geodezja i Kartografia
Katedra	Geoinformatyki i kartografii numerycznej
Nadawany tytuł	Inżynier

Tytuł pracy w języku polskim	Stworzenie aplikacji do obliczeń geodezyjnych na urządzenia z systemem Android.
Słowa kluczowe	Android, Java, Eclipse, aplikacja, obliczenia geodezyjne
Streszczenie pracy	Celem pracy jest stworzenie aplikacji do obliczeń geodezyjnych na urządzenia z systemem Android. W poszczególnych rozdziałach praca zawiera: - informacje na temat systemu android, języka programowania Java oraz środowiska programistycznego Eclipse - założenia projektowe aplikacji - opis zadań obliczeniowych - opis przygotowania środowiska programistycznego i tworzenia nowego projektu - sposób instalacji stworzonej aplikacji - porównanie aplikacji do istniejącej na rynku aplikacji „Kalkulator geodezyjny” firmy Geobid.

SPIS TREŚCI

1. WSTĘP 5

1.1. CEL I ZAKRES PRACY 5

1.2. INFORMACJE O SYSTEMIE ANDROID, ŚRODOWISKU IDE I JĘZYKU PROGRAMOWANIA JAVA 5

1.2.1. ANDROID 5

1.2.2. IDE ECLIPSE 6

1.2.3. JAVA 7

2. OPIS WYKONANYCH PRAC 8

2.1. ZAŁOŻENIA PROJEKTOWE APLIKACJI 8

2.1.1. ZAŁOŻENIA UI 8

2.1.2. ZAŁOŻENIA FUNKCJONALNOŚCI 11

2.1.2.1. ODLEGŁOŚĆ I AZYMUT 12

2.1.2.2. KĄT ZE WSPÓŁRZĘDNYCH 14

2.1.2.3. TACHIMETRIA 16

2.1.2.4. MIARY BIEGUNOWE 17

2.1.2.5. DOMIARY 20

2.1.2.6. MIARY ORTOGONALNE 22

2.1.2.7. NIWELACJA TRYGONOMETRYCZNA 24

2.1.2.8. WCIĘCIE LINIOWE 26

2.1.2.9. WCIĘCIE KĄTOWE 28

2.1.2.10. WCIĘCIE WSTECZ 30

2.1.2.11. WCIĘCIE PRZESTRZENNE W PRZÓD 34

2.2. PRZYGOTOWANIE ŚRODOWISKA I TWORZENIE APLIKACJI 36

2.2.1. POBIERANIE I INSTALACJA IDE 36

2.2.2. TWORZENIE NOWEGO PROJEKTU 38

2.3. STRUKTURA APLIKACJI 42

2.4. KOD MODUŁU OBLICZENIOWEGO 44

2.5. KOMPILACJA I EKSPORT DO APK 52

2.6. WYMAGANIA APLIKACJI 55

2.7. INSTALACJA APLIKACJI NA URZĄDZENIU Z SYSTEMEM ANDROID 55

3. PODSUMOWANIA I WNIOSKI 58

3.1. PORÓWNANIE APLIKACJI DO INNYCH DOSTĘPNYCH NA RYNKU 58

3.1.1. LAYOUT 58

3.1.2. FUNKCJONALNOŚĆ (MODUŁY OBLICZENIOWE) 58

3.1.3. INTERFEJS UŻYTKOWNIKA 59
3.2. WNIOSKI 61
4. LITERATURA 62
5. SPIS TABEL, RYSUNKÓW I WZORÓW 62

1. WSTĘP

1.1. CEL I ZAKRES PRACY

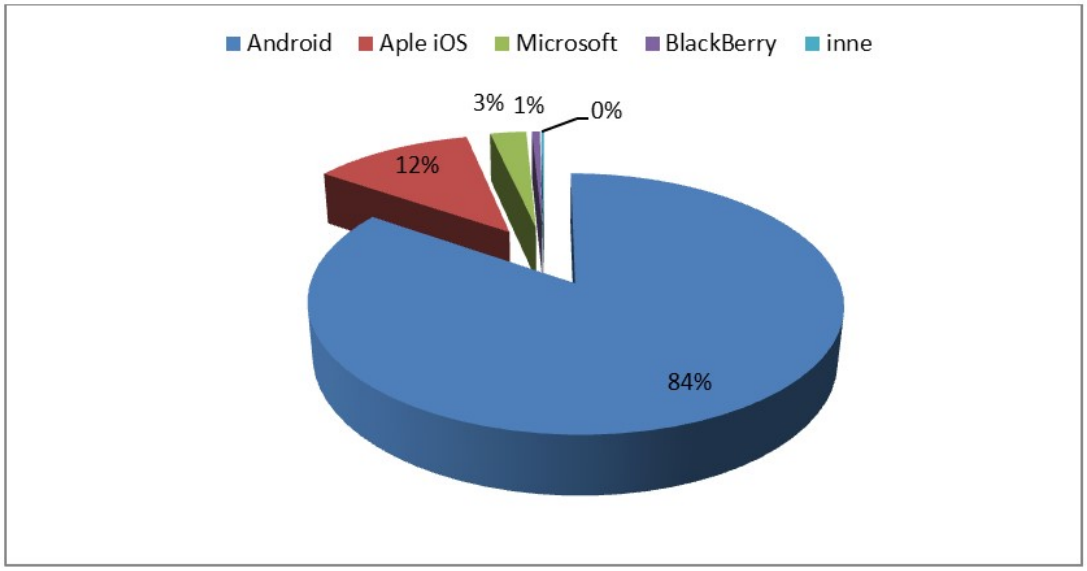
Celem pracy było stworzenie aplikacji do podstawowych obliczeń geodezyjnych dla urządzeń z systemem Android. Aplikacja powstała z myślą o geodetach i studentach geodezji. Na rynku istnieje tylko jedna aplikacja tego typu, „Kalkulator Geodezyjny" firmy Geobid. Świadczy to o bardzo małym zainteresowaniu wśród deweloperów tworzeniem aplikacji wspomagających pracę geodety. „Kalkulator Geodezyjny" firmy Geobid został pobrany z Google Play ponad 5000 razy, co oznacza duże zainteresowanie tego typu aplikacjami. Przed wyborem tematu pracy inżynierskiej autor pracy przeanalizował rynek aplikacji wspomagających pracę geodety. Na podstawie własnego doświadczenia, rozmów ze studentami geodezji i geodetami pracującymi w terenie autor pracy doszedł do wniosku, że dostępna na rynku aplikacja nie spełnia stawianych przed nią oczekiwań. Po upewnieniu się, co do zasadności i potrzeby powstania aplikacji autor przystąpił do prac projektowych. Na podstawie własnego doświadczenia i rozmów z wykonawcami opracował założenia projektowe aplikacji, które zostaną omówione w kolejnych rozdziałach. Następnie przygotował środowisko programistyczne i stworzył aplikację „GeoCalcStudent".

1.2. INFORMACJE O SYSTEMIE ANDROID, ŚRODOWISKU IDE I JĘZYKU PROGRAMOWANIA JAVA

1.2.1. ANDROID

Android jest systemem operacyjnym (ang. Operating System), czyli oprogramowaniem zarządzającym systemem komputerowym, z rodziny Linux dla urządzeń mobilnych (smartfony, telefony komórkowe, tablety, netbooki itp.). Pierwsza wersja Android 1.0 „Apple Pie" została wydana 23 września 2008 roku. Co kilka miesięcy publikowana jest nowa wersja systemu Android. Najnowszą wersją systemu jest Android 5.0.1 „Lollipop" wydany 2 grudnia 2014 roku oparty na jądrze Linux 3.4. [http://pl.wikipedia.org/wiki/Wersje_systemu_Android dostęp 28.12.2014].
Android jest najpopularniejszym systemem na platformy mobilne. W drugim kwartale 2014 roku udział w rynku mobilnych systemów wyniósł 84,6% i stale rośnie.

[http://mobirank.pl/2014/08/01/android-osiagnal-85-udzial-rynku-smartfonow/ dostęp 28.12.2014]



Rysunek 1. Udział mobilnych systemów operacyjnych na świecie w drugim kwartale 2014 roku. Źródło: opracowanie własne.

1.2.2. IDE ECLIPSE

IDE (ang. Integrated Development Environment), czyli zintegrowane środowisko programistyczne to zestaw aplikacji służących do tworzenia, testowani, modyfikowani i konserwacji oprogramowania.

Najpopularniejsze IDE dla systemu Android to Eclipse oraz Android Studio. Do napisania aplikacji autor użył IDE Eclipse. Na wybór IDE miały wpływ dostępność literatury i wsparcie dla środowiska.

IDE Eclipse zostało stworzone przez IBM i udostępnione użytkownikom na zasadach otwartego oprogramowania (ang. open source movement). Obecnie IDE Eclips jest rozwijane przez Fundację Eclipse.

Do tworzenia aplikacji dla Androida niezbędne jest wtyczka rozszerzająca możliwość Eclipse – ADT (ang. Android Development Tools). Wtyczka ta wykorzystuje pakiet SDK (ang. Software Development Kit). Pakiet SDK zawiera narzędzia niezbędne do tworzenia aplikacji na platformę Android.

1.2.3. JAVA

Java to obiektowy język programowania do tworzenia programów źródłowych kompilowanych do kodu binarnego, który jest wykonywany przez maszynę wirtualną, czyli samodzielne środowisko w systemie operacyjnym, które imituje rzeczywisty sprzęt.

Pierwsza wersja Javy została udostępniona w 1996 roku prze firmę Sun Microsystem. Obecnie właścicielem Java jest Oracle. Programy napisane w Java mogą być uruchamiane na wielu urządzeniach (telefony, komputery, tablety) oraz na różnych systemach operacyjnych przy użyciu zainstalowanej w danym systemie wirtualnej maszyny Java. Twórcy Java ustalili 11 cech, jakimi miał się cechować nowy język. Miał być:

- prosty,
- obiektowy,
- sieciowy,
- niezawodny,
- bezpieczny,
- niezależny od architektury,
- przenośny,
- interpretowany,
- wysokowydajny,
- wielowątkowy,
- dynamiczny.

Tabela 1. Ewolucja języka JAVA™

Wersja	Rok wydania	Nowe funkcje języka	Liczba klas i interfejsów
1.0	1996	Powstanie języka	211
1.1	1997	Klasy wewnętrzne	477
1.2	1998	Brak	1524
1.3	2000	Brak	1840
1.4	2002	Asercje	2723
5.0	2004	Klasy sparametryzowane, pętla *for each*, atrybuty o zmiennej liczbie argumentów *varargs,* enumeracje, statyczny import	3279
6	2006	Brak	3793
7	2011	Instrukcja *switch* z łańcuchami, operator diamentowy, literały binarne, udoskonalenie mechanizmu obsługi wyjątków	4024

Źródło: Horstmann C. Cornell G. JAVA™ Podstawy, HELION, Gliwice 2014, s. 32

2. OPIS WYKONANYCH PRAC

2.1. ZAŁOŻENIA PROJEKTOWE APLIKACJI

Autor podzielił założenia projektowe na dwie części, pierwsza dotyczy UI (ang. User Interface), czyli interfejsu użytkownika natomiast druga dotyczy funkcji programu (modułów obliczeniowych)

2.1.1. ZAŁOŻENIA UI

Interfejs użytkownika decyduje o sposobie korzystania z aplikacji przez użytkowników i o ich wrażeniach estetycznych. [Lehtimäki J. Podręcznik dla projektantów Android UI, HELION, Gliwice 2013, s. 19]

Autor na podstawie swoich doświadczeń z aplikacjami na platformę Android oraz znajomości specyfiki i warunków pracy geodety w terenie określił główne cechy, jakimi powinna charakteryzować się projektowana aplikacja, czyli:

- Czytelność
- Ergonomia
- Prostota

Pierwszy etap prac projektowych to ustalenie layoutu aplikacji. Layout to element konstrukcji aplikacji, w którym ustala się wygląd (kolorystykę, rozmiar i krój czcionki, rozmieszczenie elementów itp.) aplikacji. Ponieważ ekrany smartfonów w silnym słońcu są mało czytelne autor postanowił utrzymać layout aplikacji w jasnych kolorach, co zdecydowanie poprawi komfort pracy z aplikacją przy bardzo dużym nasłonecznieniu.

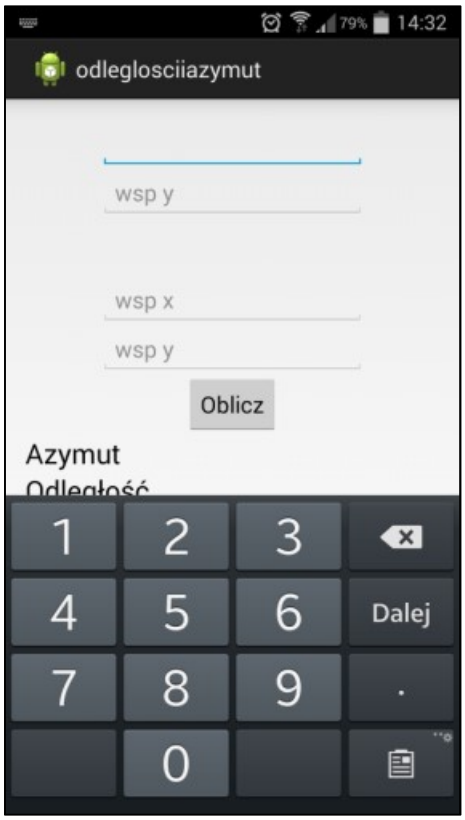
Autor postanowił, że aplikacja nie będzie ukrywała paska statusu telefonu. Powiadomienia systemu Android są widoczne tylko na pasku statusu. Dzięki temu użytkownik będzie mógł sprawdzać stan urządzenia, informacje o zdarzeniach, poziom naładowania baterii, siłę sygnału GSM czy aktualną godzinę bez wychodzenia z aplikacji.

Znając specyfikację pracy geodety w terenie autor założył, że możliwe powinno być obsługiwanie aplikacji jedną ręką. Aby to było możliwe aplikacja powinna działać wyłącznie w orientacji pionowej.

Uzyskuje się to ustawiając w pliku *AndroidManifest.xml* właściwość aktywności:

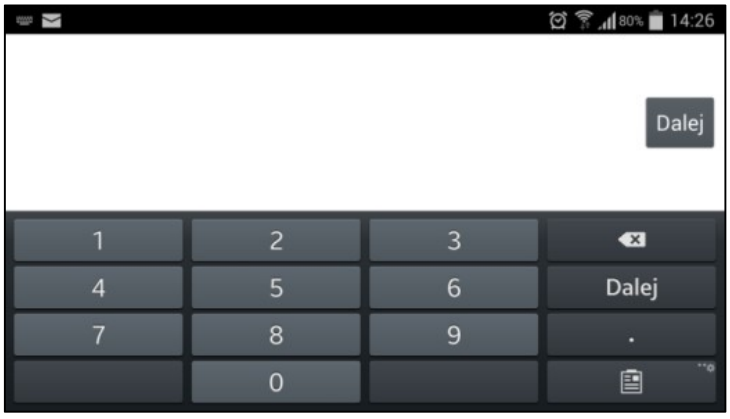
android:screenOrientation=”*portrait”*

W zależności od ustawienia urządzenia (pionowo - poziomo) w systemie android zmienia się typ klawiatury i sposób wprowadzania danych. Klawiatura pozioma zajmuje cały ekran zakrywając elementy aktywności. Klawiatura pionowa przysłania tylko dolną cześć aktywności, dzięki czemu na bieżąco widzimy, do których ediTextów wpisujemy dane oraz jest wygodniejsza do obsługi jedną ręką.



Rysunek 2. Klawiatura pionowa (Android 4.4.2)

Źródło: opracowanie własne



Rysunek 3. Klawiatura pozioma (Android 4.4.2)

Źródło: opracowanie własne

Dodatkowo przy każdej zmianie orientacji urządzenia zmienia się sposób wyświetlania, co powoduje opóźnienia w działaniu aplikacji. Dzięki temu, że aplikacja będzie działała tylko w orientacji pionowej bez względu na ustawienia systemowe urządzenia powyższe niedogodności zostaną wyeliminowane.

Aby jak najłatwiej wpisywać dane do poszczególnych pól editText autor postanowił wykorzystać właściwości kontrolki editText:

- android:*selectAllOnFocus=”true*”
- android:*nextFocusDown=”@-id/*id_kolejnej_kontrolki”

Pierwsza właściwość ułatwia edycję danych wprowadzonych do pola editText. Gdy zorientujemy się, że dane są błędne możemy je zmienić na dwa sposoby: wpisując od nowa lub zmieniając poszczególne znaki we wpisanych danych. Po ustawieniu tej właściwości jednokrotne dotknięcie pola editText powoduje otworzenie klawiatury ekranowej i zaznaczenie całego ciągu znaków w polu editText. Naciśnięcie klawisza klawiatury powoduje skasowanie wprowadzonych błędnie danych i rozpoczęcie wpisywania nowych. Jeśli chcemy edytować pojedyncze znaki wystarczy dotknąć pole editText dwukrotnie, wyświetli się kursor i będziemy mogli edytować poszczególne znaki bez potrzeby wpisywania ich od nowa.

Druga właściwość powoduje, że przycisk „OK.” na klawiaturze ekranowej zmienia się w „Dalej”. Dzięki temu nie musimy pokazywać aplikacji, które pole editText chcemy wypełnić, jako kolejne. Po naciśnięciu klawisza „Dalej” kursor przeskakuje do wskazanego przez programistę kolejnego pola editText. Po uzupełnieniu wszystkich pól editText klawisz „Dalej” zmienia się w „OK.” Naciśnięcie go powoduje zamknięcie klawiatury ekranowej.

Aby aplikacja była czytelna dla użytkowników autor postanowił użyć podpowiedzi w polach editText. Dzięki właściwości:

android:hint=”*@string/tekst_podpowiedzi”*

W polach editText jest wyświetlany tekst informujący o rodzaju danych, które powinny być wprowadzone do pola editText.

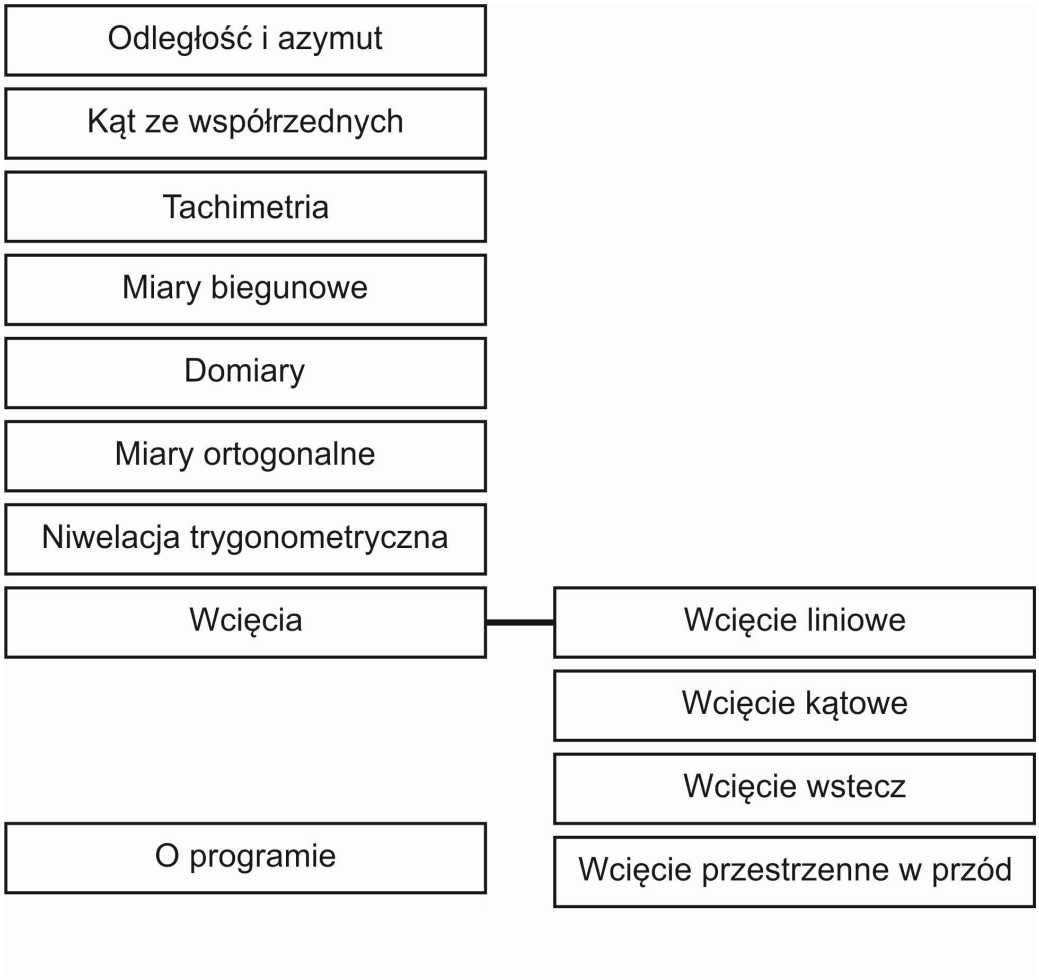
Do każdego modułu obliczeniowego autor dołączy rysunek ilustrujący funkcję modułu dostępny w opcji „POMOC”

Pod polami editText, do których wprowadzamy dane będzie znajdował się duży klawisz „OBLICZ” (buton). Po naciśnięciu go, aplikacja sprawdzi czy wprowadzone dane są poprawne (w razie błędów wyświetli odpowiedni komunikat *toast*), wykona obliczenia i wyświetli wynik. Komunikaty *toast* to małe okna wyświetlane na kilka sekund w oknie aktywności. [Darwin I. Android receptury, HELION, Gliwice 2012, s. 315]

Jako, że aplikacja ma między innymi wspomagać pracę geodety w terenie autor postanowił, że zamiast kątów do obliczeń będzie się podawało kierunki.

2.1.2. ZAŁOŻENIA FUNKCJONALNOŚCI

Aplikacja będzie składać się z 11 modułów obliczeniowych. Dane wejściowe będą wpisywane w metrach oraz w gradach. Wyniki będą podawane w metrach z dokładnością do dwóch miejsc po przecinku i w gradach z dokładnością do czterech miejsc po przecinku, współrzędne są obliczane do dwóch miejsc po przecinku. W kolejnych podrozdziałach opisane zostaną poszczególne moduły obliczeniowe.

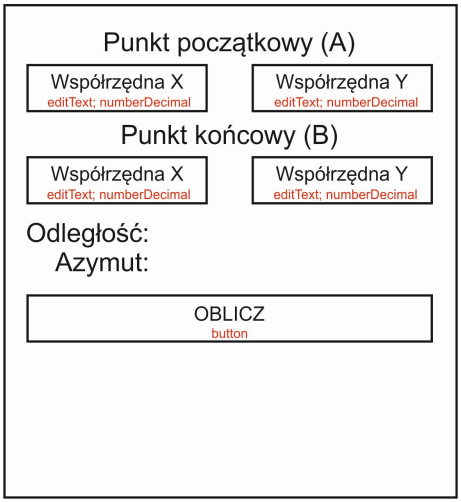


Rysunek 4. Struktura aplikacji

Źródło: opracowanie własne

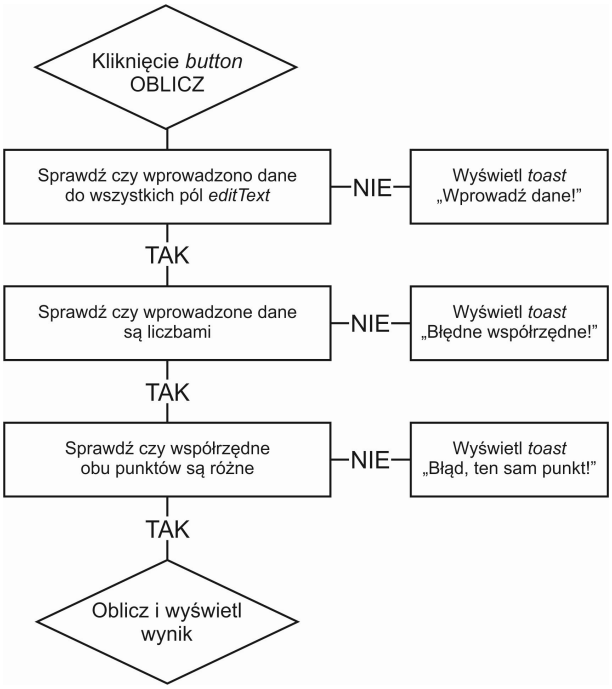
2.1.2.1.ODLEGŁOŚĆ I AZYMUT

Moduł liczy odległość pomiędzy dwoma punktami o znanych współrzędnych oraz azymut tworzonego przez te punkty boku.



Rysunek 5. Layout modułu „Odległość i azymut”

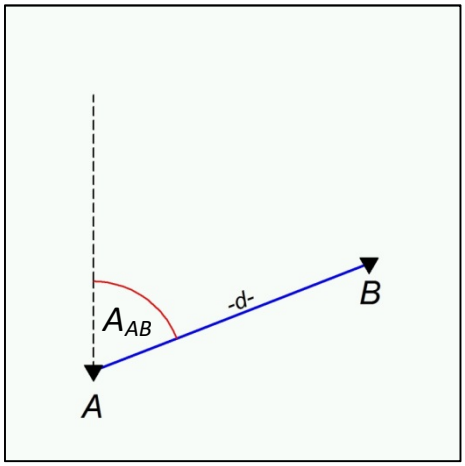
Źródło: opracowanie własne



Rysunek 6. Diagram modułu „Odległość i azymut”

Źródło: opracowanie własne

Azymut boku AB (A_{AB}) jest to kąt poziomy zawarty w przedziale $A_{AB} \in < 0^g; 400^g)$ pomiędzy kierunkiem północy (oś X) wychodzącym z punktu A a danym bokiem AB, liczonym od kierunku północy w prawo, czyli zgodnie z ruchem wskazówek zegara. [Jagielski A. Geodezja I, GEODPIS, Kraków 2002, s. 183]



Rysunek 7. Odległość i azymut

Źródło: opracowanie własne

Aby obliczyć azymut należy najpierw policzyć wartość czwartaka (φ) na podstawie jego funkcji tangens. Następnie liczymy azymut na podstawie zależności pomiędzy azymutem a czwartakiem określonym na podstawie przyrostów współrzędnych X i Y.

Tabela 2. Zależności pomiędzy azymutem a czwartakiem

Numer ćwiartki azymutu	Znaki przyrostów		Zależność pomiędzy azymutem a czwartakiem
	ΔX	ΔY	
I	+	+	A = φ
II	-	+	A = 200^g - φ
III	-	-	A = 200^g + φ
IV	+	-	A = 400^g - φ

Źródło: opracowanie własne

$$\boldsymbol{\varphi = \tan^{-1}\left|\frac{\Delta Y}{\Delta X}\right|}$$

Wzór 1. Wzór na obliczenie czawrtaka

ΔY – przyrost współrzędnej Y

ΔX – przyrost współrzędnej X

$$\boldsymbol{d = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}}$$

Wzór 2. Wzór na obliczenie długości

X_A- współrzędna X punktu A

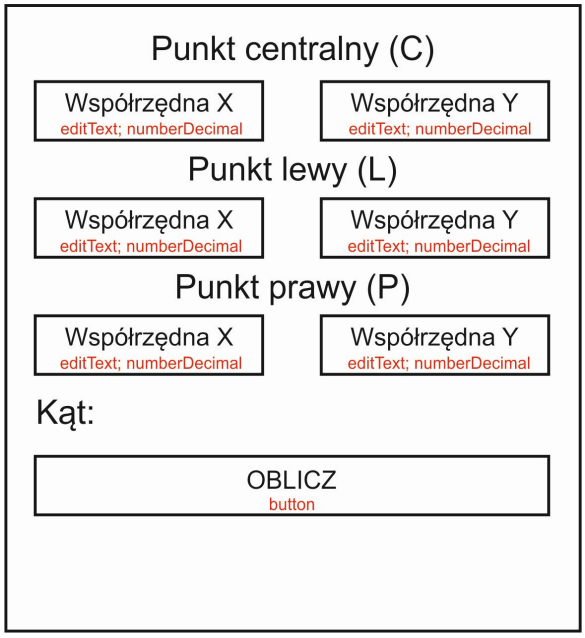
Y_A- współrzędna Y punktu A

X_B- współrzędna X punktu B

Y_B- współrzędna Y punktu B

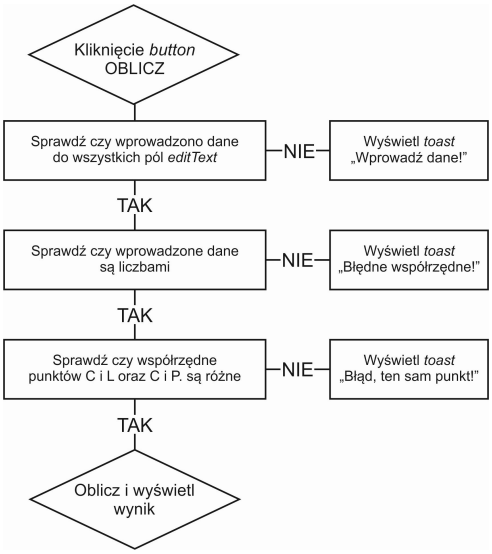
2.1.2.2.KĄT ZE WSPÓŁRZĘDNYCH

Moduł liczy kąt płaski zawarty pomiędzy dwoma odcinkami wychodzącymi z jednego punktu.



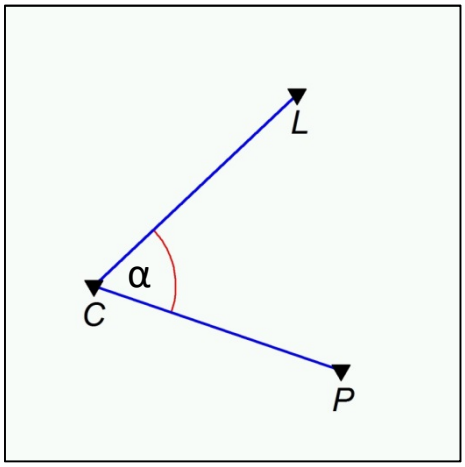
Rysunek 8. Layout modułu „Kąt ze współrzędnych”

Źródło: opracowanie własne



Rysunek 9. Diagram modułu „Kąt ze współrzędnych"

Źródło: opracowanie własne



Rysunek 10. Kąt ze współrzędnych

Źródło: opracowanie własne

$$\boldsymbol{\alpha = A_{CP} - A_{CL}}$$

Wzór 3. Wzór na obliczenie kąta płaskiego

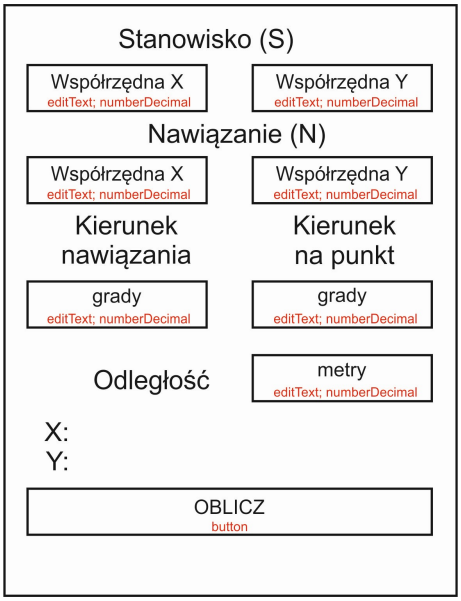
A_{CP} – azymut boku CL

A_{CL} – azymut boku CP

Jeżeli w wyniku obliczeń uzyskamy wynik mniejszy od zera ($\alpha < 0$) to do wyniku dodajemy kąt pełny.

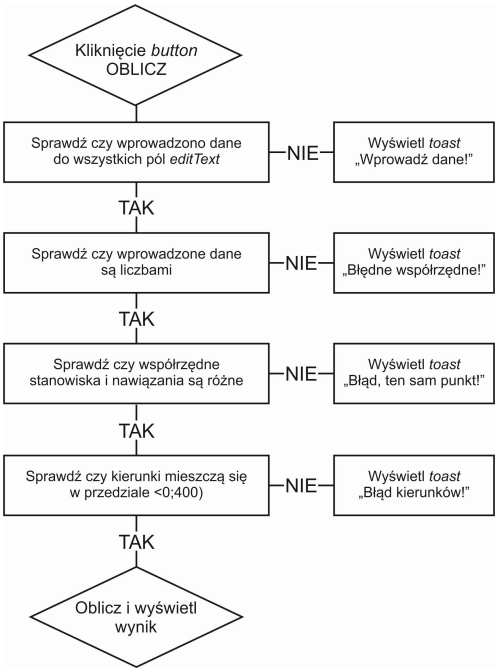
2.1.2.3. TACHIMETRIA

Moduł liczy współrzędne punktu na podstawie miar biegunowych (kąt i odległość).



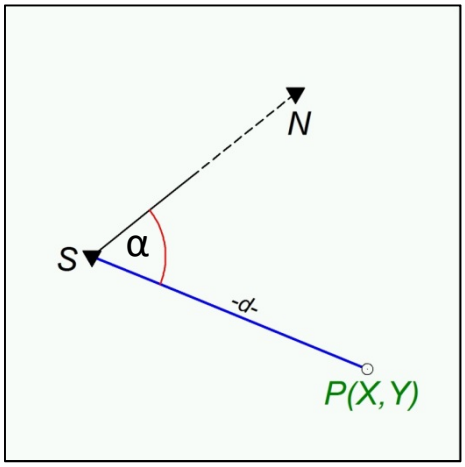
Rysunek 11. Layout modułu „Tachimetria”

Źródło: opracowanie własne



Rysunek 12. Diagram modułu „Tachimetria”

Źródło: opracowanie własne



Rysunek 13. Tachimetria

Źródło: opracowanie własne

$$\boldsymbol{X_P = X_S + \cos A_{SP} \cdot d_{SP}}$$

Wzór 4. Wzór na obliczenie współrzędnej X

$$\boldsymbol{Y_P = Y_S + \sin A_{SP} \cdot d_{SP}}$$

Wzór 5. Wzór na obliczenie współrzędnej Y

$$\boldsymbol{A_{SP} = A_{SN} + \alpha}$$

Wzór 6. Wzór na obliczenie azymutu boku SP

X_P – obliczana współrzędna X

Y_P – obliczana współrzędna Y

X_S – współrzędna X stanowiska

Y_S – współrzędna Y stanowiska

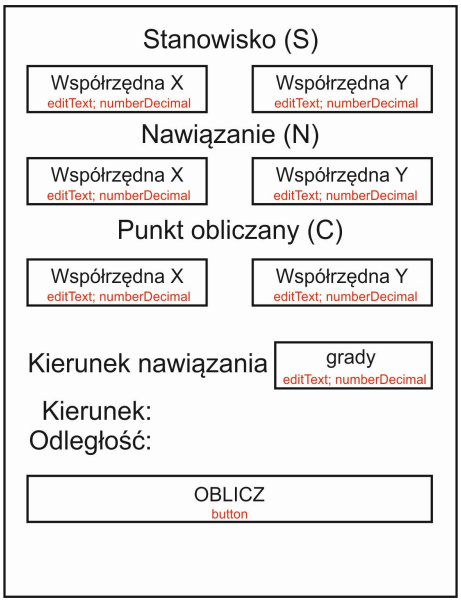
d_{SP} – pomierzona długość

α – pomierzony kąt

A_{SN} – azymut boku SN

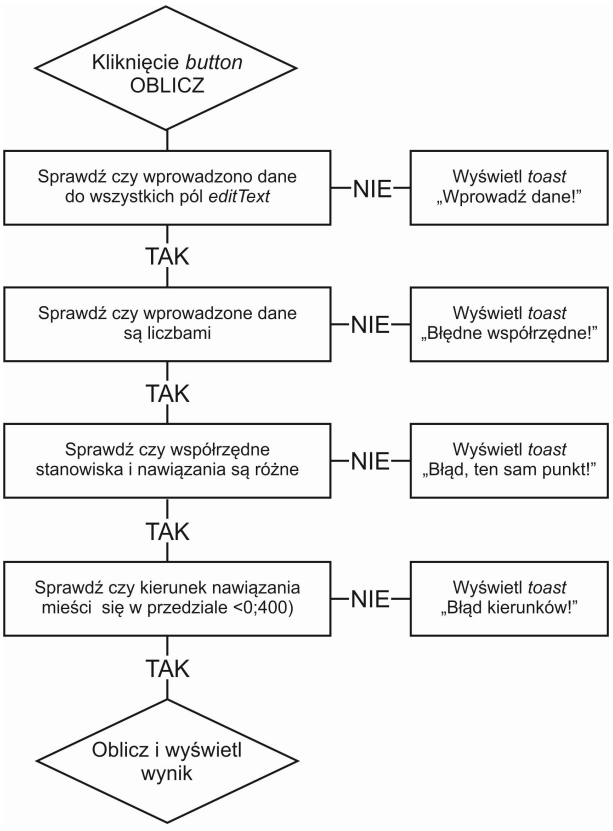
2.1.2.4.MIARY BIEGUNOWE

Moduł oblicza miary biegunowe (kat i odległość) na podstawie współrzędnych punktów stanowiska, nawiązania i tyczonego punktu.



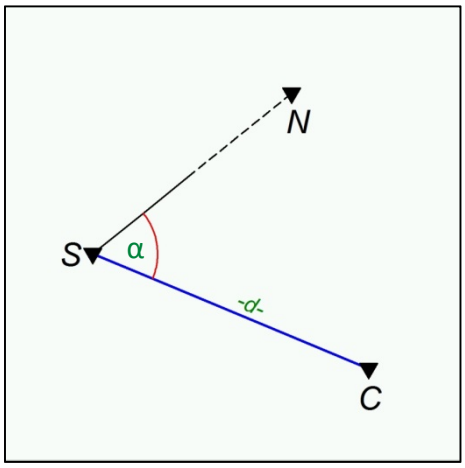
Rysunek 14. Layout modułu „Miary biegunowe”

Źródło: opracowanie własne



Rysunek 15. Diagram modułu „Miary biegunowe”

Źródło: opracowanie własne



Rysunek 16. Miary biegunowe

Źródło: opracowanie własne

$$\boldsymbol{\alpha = A_{SC} - A_{SN}}$$

Wzór 7. Wzór na obliczenie kąta płaskiego

$$\boldsymbol{d = \sqrt{(X_C - X_S)^2 + (Y_C - Y_S)^2}}$$

Wzór 8. Wzór na obliczenie długości

X_C- współrzędna X punktu C

Y_C- współrzędna Y punktu C

X_S- współrzędna X punktu S

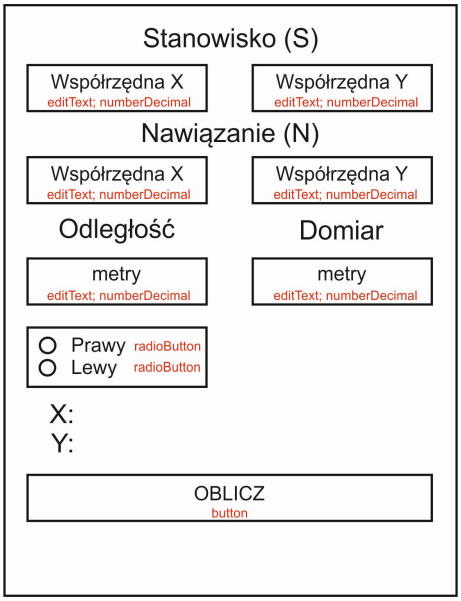
Y_S- współrzędna Y punktu S

A_{SC} – azymut boku SC

A_{SN} – azymut boku SN

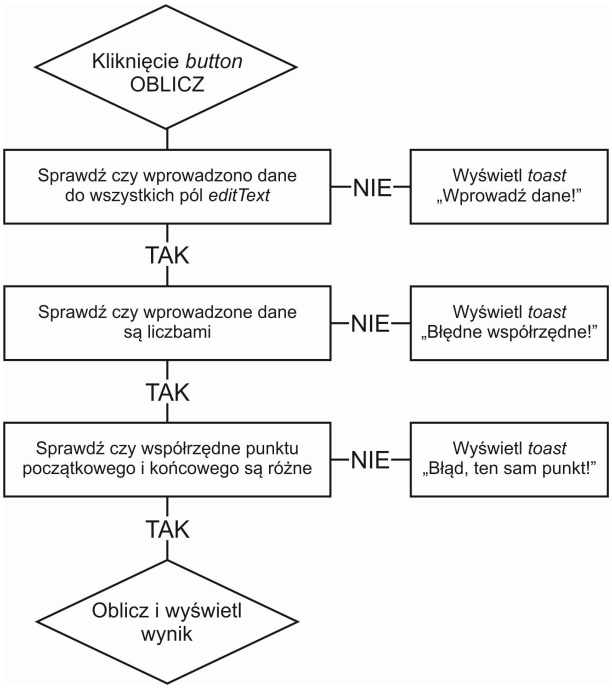
2.1.2.5.DOMIARY

Moduł liczy współrzędne punktu na podstawie domiarów prostokątnych (rzędna i odcięta).



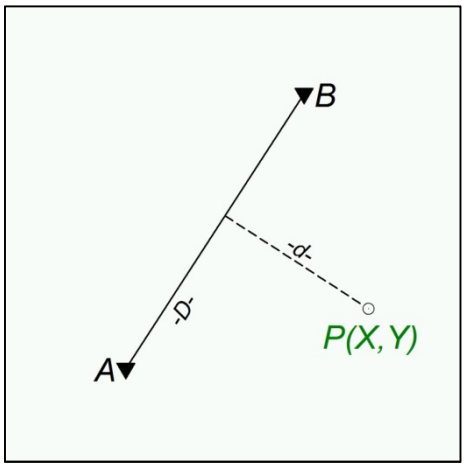
Rysunek 17. Layout modułu „Domiary”

Źródło: opracowanie własne



Rysunek 18. Diagram modułu „Domiary”

Źródło: opracowanie własne



Rysunek 19. Domiary

Źródło: opracowanie własne

$$\boldsymbol{X_P = X_A + D \cdot \cos A_{AB} - (\pm d) \cdot \sin \quad _{AB}}$$

Wzór 9. Wzór na obliczenie współrzędnej X

$$\boldsymbol{Y_P = Y_A + D \cdot \sin A_{AB} + (\pm d) \cdot \cos A_{AB}}$$

Wzór 10. Wzór na obliczenie współrzędnej Y

X_A – współrzędna X początku linii pomiarowej

Y_A – współrzędna Y początku linii pomiarowej

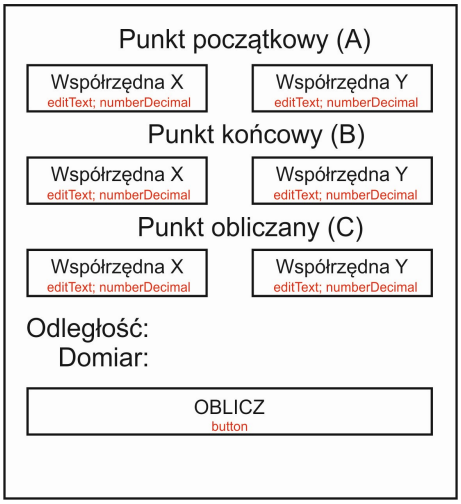
D – rzędna

d – odcięta, na prawo od linii pomiarowej (+), na lewo od linii pomiarowej (-)

A_{AB} – azymut boku AB

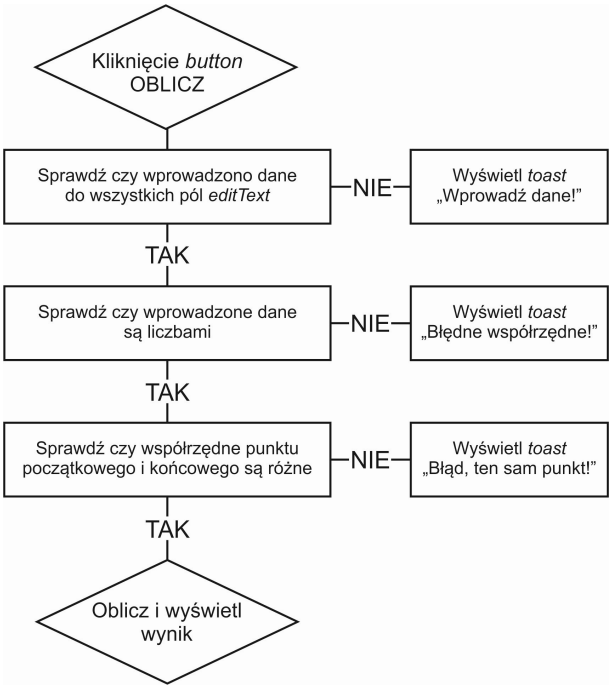
2.1.2.6.MIARY ORTOGONALNE

Moduł liczy rzędną i odciętą na podstawie współrzędnych punktów.



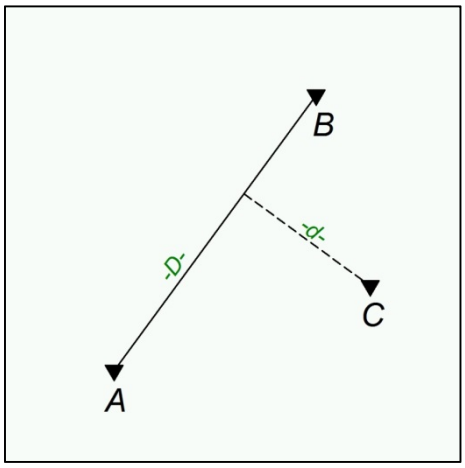
Rysunek 20. Layout modułu „Miary ortogonalne”

Źródło: opracowanie własne



Rysunek 21. Diagram modułu „Miary ortognalne”

Źródło: opracowanie własne



Rysunek 22. Miary ortogonalne

Źródło: opracowanie własne

Rzędna i odcięta są liczone na podstawie wzorów:

$$\boldsymbol{D = \cos\alpha \cdot d_{AC}}$$

Wzór 11. Wzór na obliczenie długości rzędnej

$$\boldsymbol{d = \sin\alpha \cdot d_{AC}}$$

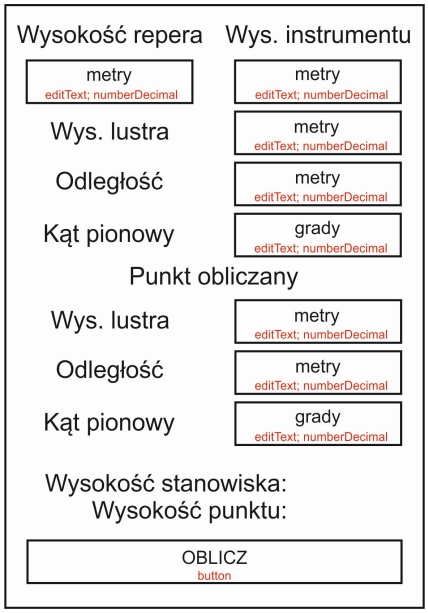
Wzór 12. Wzór na obliczenie długości odciętej

α- kat pomiędzy bokami AB i AC

d_{AC} – odległość pomiędzy punktami A i C

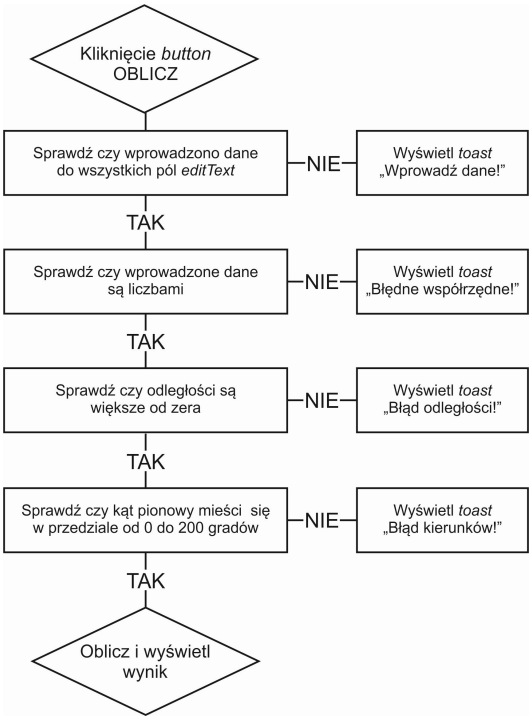
2.1.2.7.NIWELACJA TRYGONOMETRYCZNA

Moduł liczy wysokość punktów na podstawie pomierzonych odległości i katów pionowych.



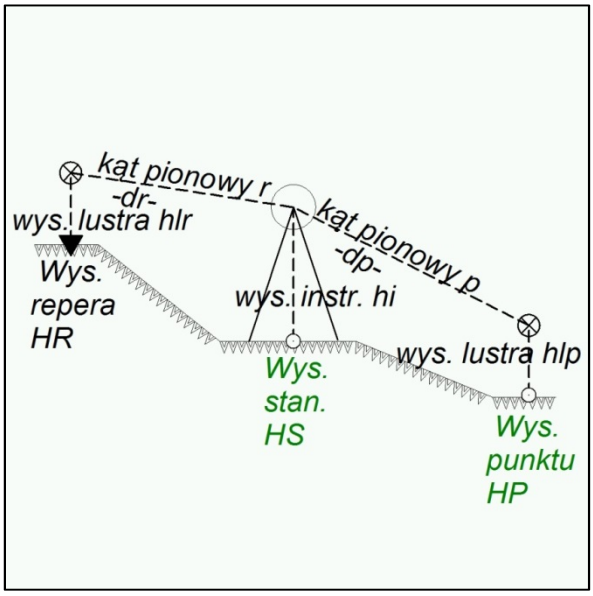
Rysunek 23. Layout modułu „Niwelacja trygonometryczna”

Źródło: opracowanie własne



Rysunek 24. Diagram modułu „Niwelacja trygonometryczna”

Źródło: opracowanie własne



Rysunek 25. Niwelacja trygonometryczna

Źródło: opracowanie własne

Wysokości stanowiska i punktu mierzonego liczymy na podstawie wzorów:

$$\boldsymbol{H_S = H_R + h_{lr} \mp \tan \alpha \cdot d_r - h_i}$$

Wzór 13. Wzór na obliczenie wysokości stanowiska

$$\boldsymbol{H_P = H_S + h_i \mp \tan \beta \cdot d_p - h_{lp}}$$

Wzór 14. Wzór na obliczenie wysokości punktu

H_R – wysokość repera

h_{lr} – wysokość lustra na reperze

h_i – wysokość instrumentu

h_{lp} – wysokość lustra na mierzonym punkcie

d_r – odległość pomiędzy reperem a stanowiskiem

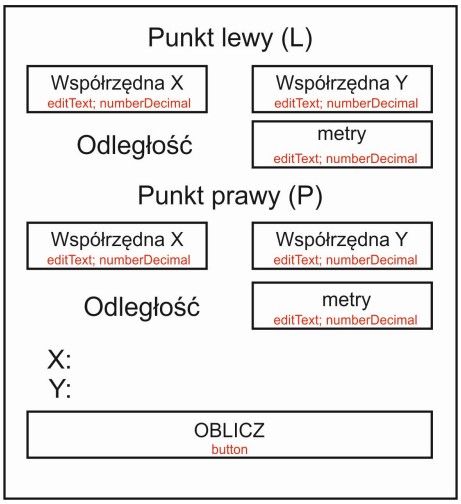
d_p – odległość pomiędzy stanowiskiem a mierzonym punktem

α – kąt pomiędzy płaszczyzną a katem pionowym na reper

β – kąt pomiędzy płaszczyzną a katem pionowym na punkt

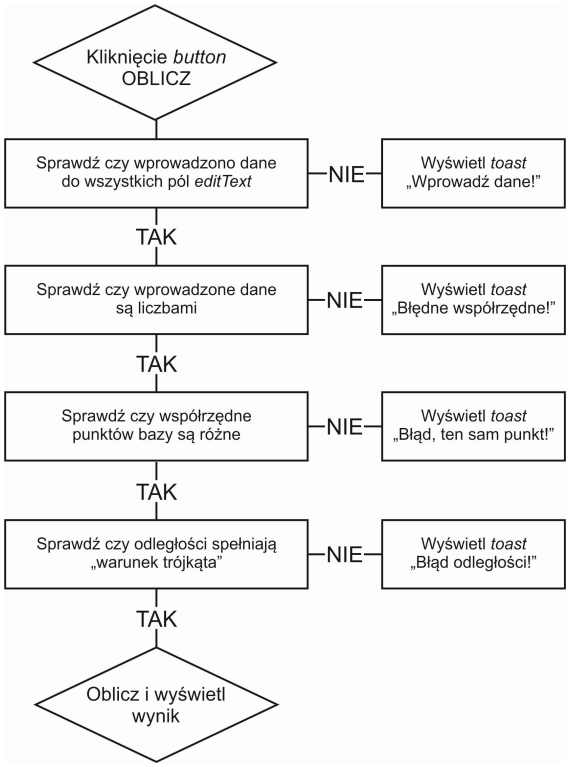
2.1.2.8. WCIĘCIE LINIOWE

Pojedyncze wcięcie liniowe polega na wyznaczeniu współrzędnych punktu wcinanego w oparciu o pomierzone odległości pomiędzy punktami wyznaczającymi bazę wcięcia. [Jagielski A. Geodezja II, GEODPIS, Kraków 2003, s. 250]



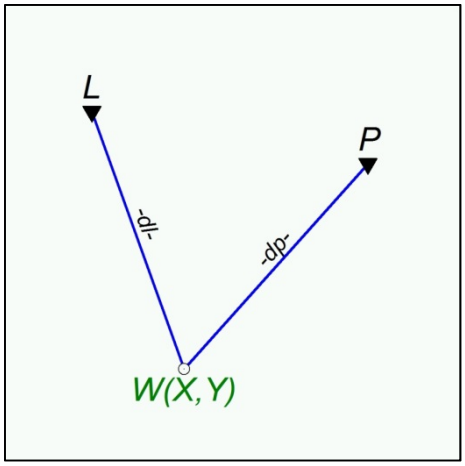
Rysunek 26. Layout modułu „Wcięcie liniowe”

Źródło: opracowanie własne



Rysunek 27. Diagram modułu „Wcięcie liniowe”

Źródło: opracowanie własne



Rysunek 28. Wcięcie liniowe

Źródło: opracowanie własne

Współrzędne obliczanego punktu liczymy ze wzorów:

$$\boldsymbol{X_W = X_L + dl \cdot \cos A_{LW}}$$

Wzór 15. Wzór na obliczenie współrzędnej X wcinanego punktu

$$\boldsymbol{Y_W = Y_L + dl \cdot \sin A_{LW}}$$

Wzór 16. Wzór na obliczenie współrzędnej Y wcinanego punktu

$$\boldsymbol{A_{LW} = A_{LP} + \alpha}$$

Wzór 17. Wzór na obliczenie azymutu boku LW

$$\boldsymbol{\alpha = \cos^{-1} \frac{dp^2 - \left(d_{LP}{}^2 + dl^2\right)}{-2 \cdot d_{LP} \cdot dl}}$$

Wzór 18. Wzór na obliczenie kąta pomiędzy bokami LP i LW

X_L – współrzędna X punktu L

Y_L – współrzędna Y punktu L

A_{LW} – azymut boku LW

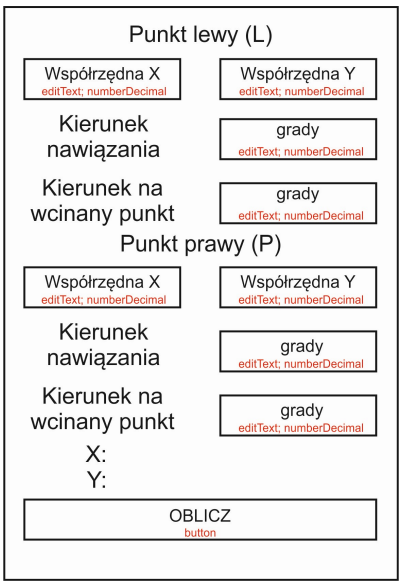
dl – długość boku WL

dp – długość boku WP

d_{LP} – długość boku LP

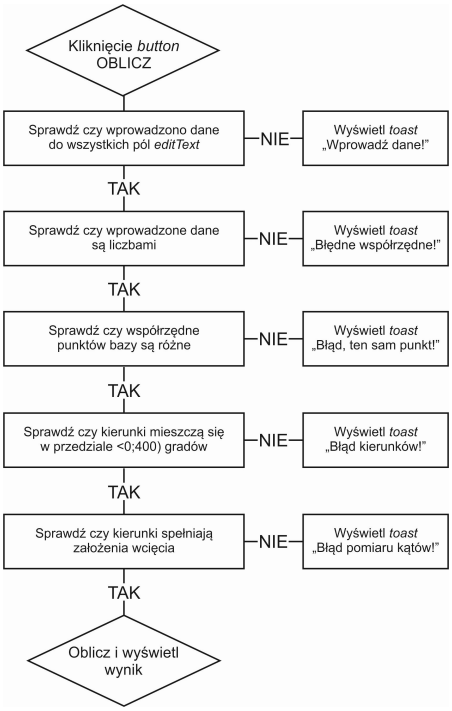
2.1.2.9.WCIĘCIE KĄTOWE

Kątowe wcięcie w przód polega na określeniu współrzędnych punktu wcinanego na podstawie pomiaru dwóch kątów zawartych pomiędzy bazą wcięcia a punktem wcinanym.



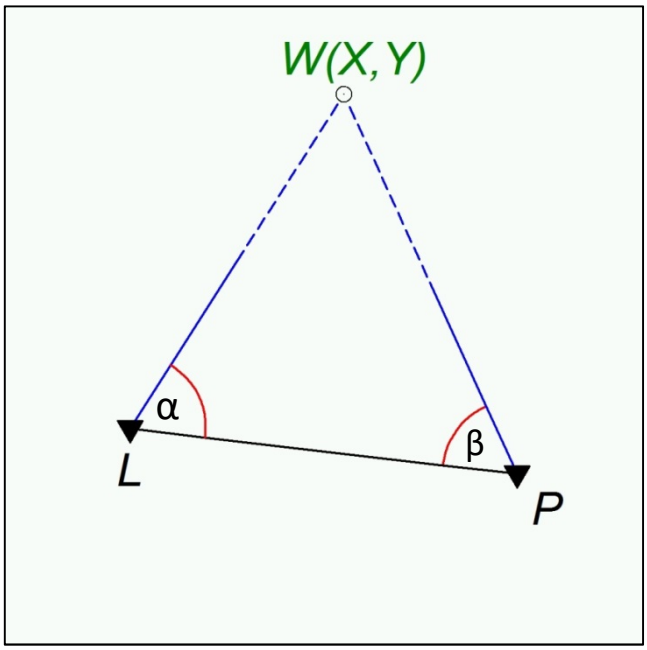
Rysunek 29. Layout modułu „Wcięcie kątowe”

Źródło: opracowanie własne



Rysunek 30. Diagram modułu „Wcięcie kątowe”

Źródło: opracowanie własne



Rysunek 31. Wcięcie kątowe

Źródło: opracowanie własne

Współrzędne obliczanego punktu liczymy ze wzorów:

$$\boldsymbol{X_W = X_L + d_{LW} \cdot \cos A_{LW}}$$

Wzór 19. Wzór na obliczenie współrzędnej X wcinanego punktu

$$\boldsymbol{Y_W = Y_L + d_{LW} \cdot \sin A_{LW}}$$

Wzór 20. Wzór na obliczenie współrzędnej Y wcinanego punktu

$$\boldsymbol{d_{LW} = \frac{d_{LP} \cdot \sin \alpha}{\sin(\alpha + \beta)}}$$

Wzór 21. Wzór na obliczenie długości boku LW

$$\boldsymbol{A_{LW} = A_{LP} - \alpha}$$

Wzór 22. Wzór na obliczenie azymutu boku LW

X_L – współrzędna X punktu L

Y_L – współrzędna Y punktu L

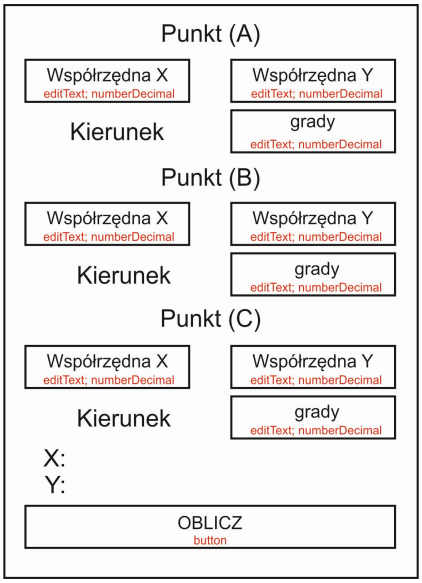
A_{LP} – azymut boku LP

α – kąt lewy

β – kąt prawy

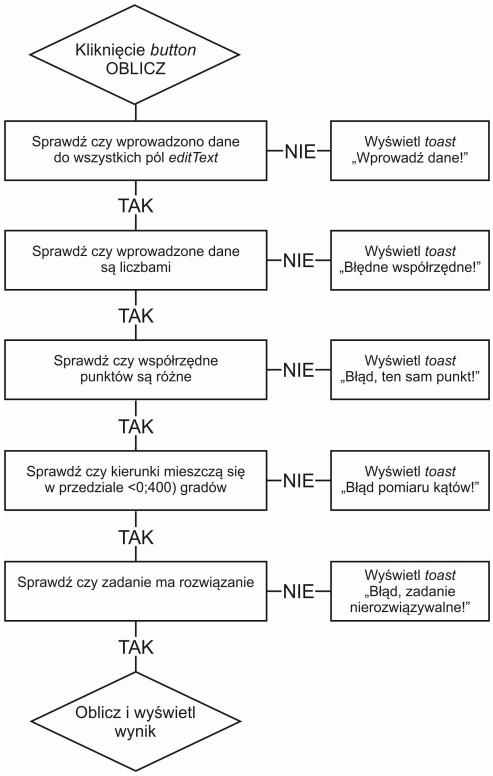
2.1.2.10. WCIĘCIE WSTECZ

Pojedyncze wcięcie wstecz polega na określeniu współrzędnych punktu na podstawie pomierzonych kątów z punktu wcinanego do trzech punktów o znanych współrzędnych.



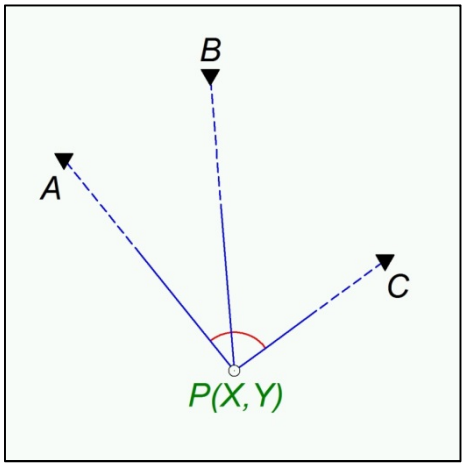
Rysunek 32. Layout modułu „Wcięcie wstecz”

Źródło: opracowanie własne



Rysunek 33. Diagram modułu „Wcięcie wstecz”

Źródło: opracowanie własne

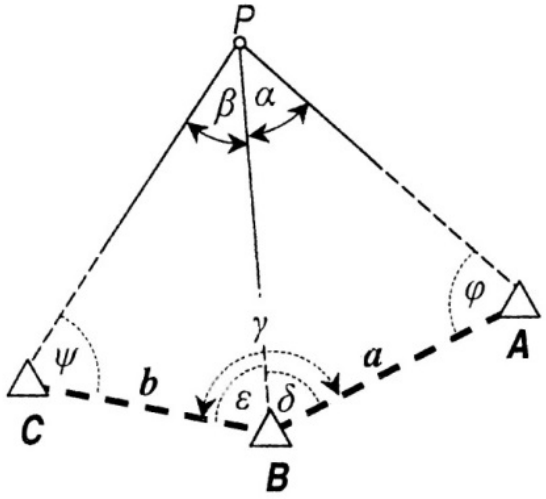


Rysunek 34. Wcięcie wstecz

Źródło: opracowanie własne

Rozwiązanie wcięcia wstecz sposobem Kästnera.

Sposób ten polega na znalezieniu kątów pomocniczych φ i ψ a następnie do sprowadzenia zadania do typowego wcięcia w przód.



Rysunek 35. Kąty pomocnicze φ i ψ

Źródło: Jagielski A. Geodezja II, GEODPIS, Kraków 2003, str 255

Znajomość współrzędnych punktów A, B, C pozwala na obliczenie kąta γ, wyznaczenie długości AB i BC oraz azymutów tych boków.

Po wprowadzeniu oznaczeń:

$$\varphi =< PAB \; oraz \; \psi =< PCB$$

na podstawie sumy kątów w czworoboku ABCP można napisać:

$$\varphi + \beta + \gamma + \varphi + \psi = 400^g$$

stąd:

$$\varphi + \psi = 400^g - (\alpha + \beta + \gamma)$$

Połowa sumy kątów pomocniczych wyniesie więc:

$$\frac{\varphi + \psi}{2} = \frac{400^g - (\alpha + \beta + \gamma)}{2}$$

Celem dalszego postępowania, prowadzącego do określenia wartości kątów φ i ψ jest wyznaczenie połowy różnicy tych kątów. Na podstawie twierdzenia sinusów w trójkątach ABP i BCP można dwukrotnie zapisać wzory na długość ich wspólnego boku BP, a następnie zrównać ze sobą prawe strony obu równań:

$$\frac{a}{\sin\alpha} \cdot \sin\varphi = \frac{b}{\sin\beta} \cdot \sin\psi$$

Przekształcenie tej równości daje następującą proporcję:

$$\sin\psi : \sin\varphi = (a \cdot \sin\beta) : (b \cdot \sin\alpha)$$

Wyrażenie występujące po prawej stronie powyższego równania jest znaną wielkością, która może stanowić tangens pewnego, pomocniczego kąta μ, zaś sposób obliczenia funkcji tg μ określa wzór:

$$\tan\mu = \frac{a \cdot \sin\beta}{b \cdot \sin\alpha}$$

lecz jednocześnie:

$$\tan\mu = \frac{\sin\psi}{\sin\varphi}$$

a więc:

$$\tan(50^g - \mu) = \frac{1 - \tan\mu}{1 + \tan\mu} = \frac{1 - \frac{\sin\psi}{\sin\varphi}}{1 + \frac{\sin\psi}{\sin\varphi}} = \frac{\sin\varphi - \sin\psi}{\sin\varphi + \sin\psi}$$

Na podstawie znanych wzorów trygonometrycznych na różnicę i sumę sinusów kątów możemy napisać:

$$\sin\varphi - \sin\psi = 2\sin\frac{\varphi - \psi}{2} \cdot \cos\frac{\varphi + \psi}{2}$$

oraz:

$$\sin\varphi + \sin\psi = 2\cos\frac{\varphi - \psi}{2} \cdot \sin\frac{\varphi + \psi}{2}$$

stąd:

$$\tan(50^g - \mu) = \frac{2\sin\frac{\varphi-\psi}{2}\cdot\cos\frac{\varphi+\psi}{2}}{2\cos\frac{\varphi-\psi}{2}\cdot\sin\frac{\varphi+\psi}{2}} = \tan\frac{\varphi-\psi}{2}\cdot\frac{1}{\tan\frac{\varphi+\psi}{2}}$$

po prostym przekształceniu zapiszemy równanie na obliczenie tangensa połowy różnicy kątów pomocniczych φ i ψ.

$$\tan\frac{\varphi-\psi}{2} = \tan\frac{\varphi+\psi}{2}\cdot\tan(50^g - \mu)$$

Na podstawie wartości połowy sumy i połowy różnicy kątów φ i ψ możemy teraz wyznaczyć oba poszukiwane kąty pomocnicze:

$$\varphi = \frac{\varphi+\psi}{2} + \frac{\varphi-\psi}{2}$$

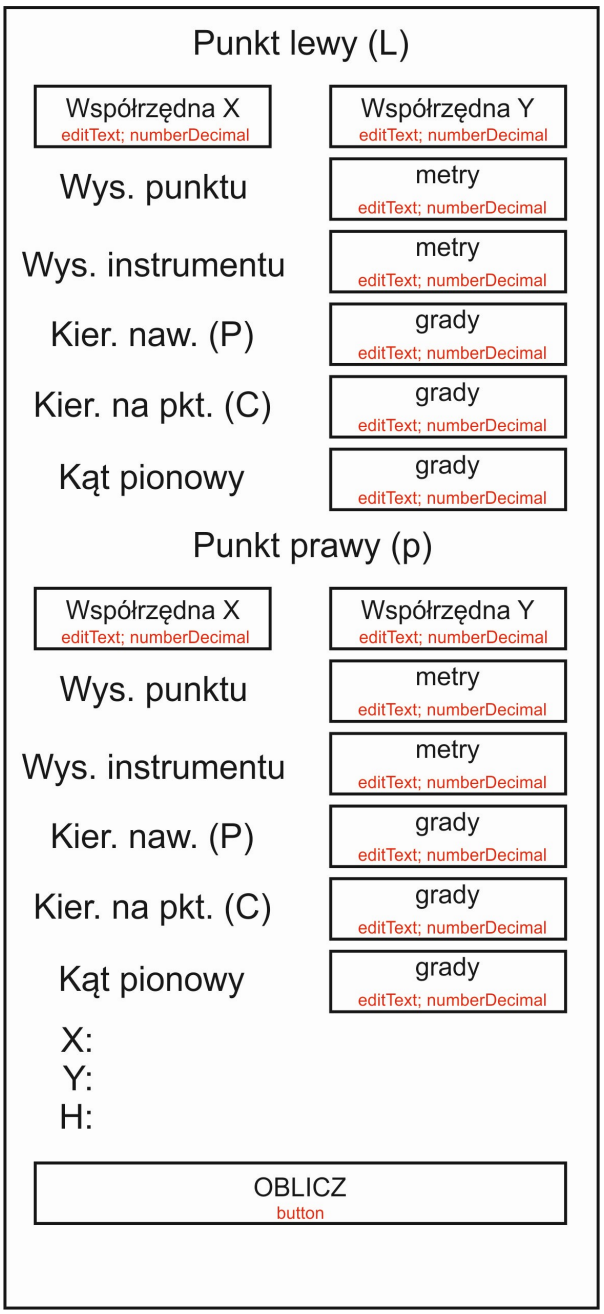
$$\psi = \frac{\varphi+\psi}{2} + \frac{\varphi-\psi}{2}$$

Znając wartość kątów α, γ, φ i elementów trójkąta ABP, obliczamy kąt δ, a następnie współrzędne punktu P według znanej procedury kątowego wcięcia w przód. [Jagielski A. Geodezja II, GEODPIS, Kraków 2003, s. 255-257]

Sposób obliczania wcięcia kątowego opisano w rozdziale 2.1.2.9.

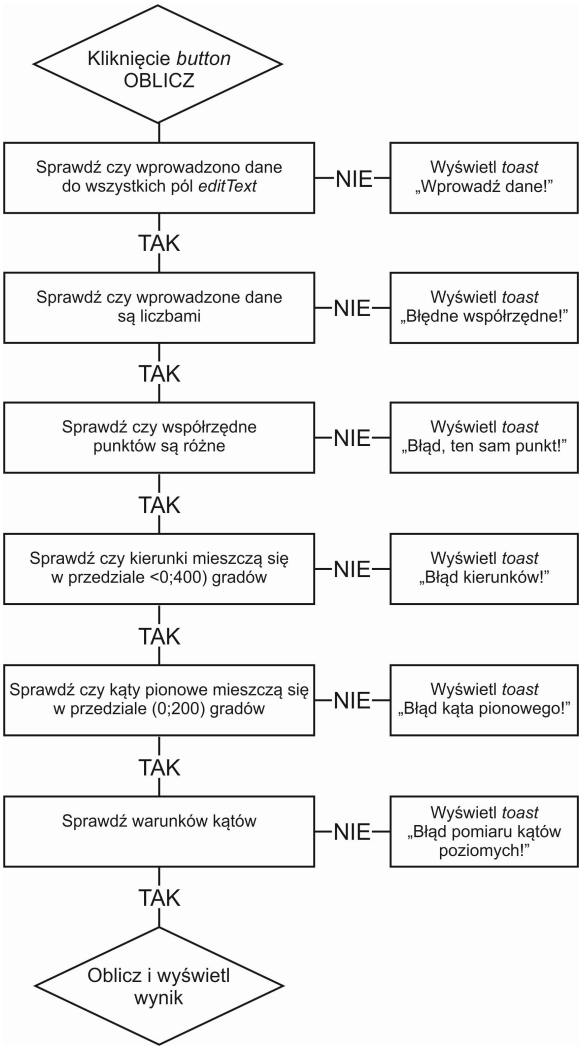
2.1.2.11. WCIĘCIE PRZESTRZENNE W PRZÓD

Wcięcie przestrzenne w przód polega na określeniu współrzędnych i wysokości punktu niedostępnego.



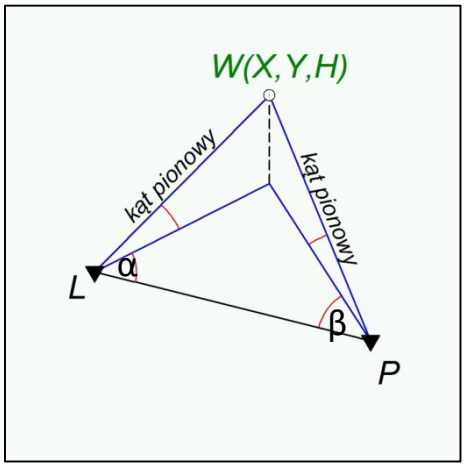
Rysunek 36. Layout modułu „Wcięcie przestrzenne w przód”

Źródło: opracowanie własne



Rysunek 37. Layout modułu „Wcięcie przestrzenne w przód”

Źródło: opracowanie własne



Rysunek 38. Wcięcie przestrzenne w przód

Źródło: opracowanie własne

Określenie współrzędnych punktu niedostępnego (W) polega na obliczeniu wcięcia kątowego opisanego w rozdziale 2.1.2.9.

Wysokość punktu niedostępnego obliczamy metodą niwelacji trygonometrycznej opisanej w rozdziale 2.1.2.7.

2.2. PRZYGOTOWANIE ŚRODOWISKA I TWORZENIE APLIKACJI

2.2.1. POBIERANIE I INSTALACJA IDE

Pakiet oprogramowania pobrano ze strony:

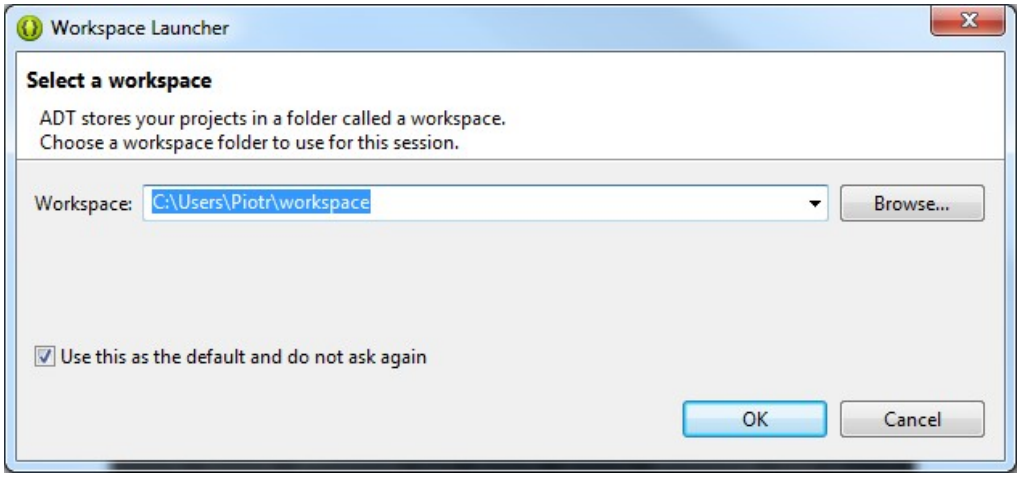
https://developer.android.com/sdk/index.html?hl=i#download

Pakiet zawierał:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools

Po rozpakowaniu ściągniętego pliku autor przystąpił do konfiguracji IDE. Eclipse nie wymaga instalacji, wystarczy uruchomić plik eclipse.exe.

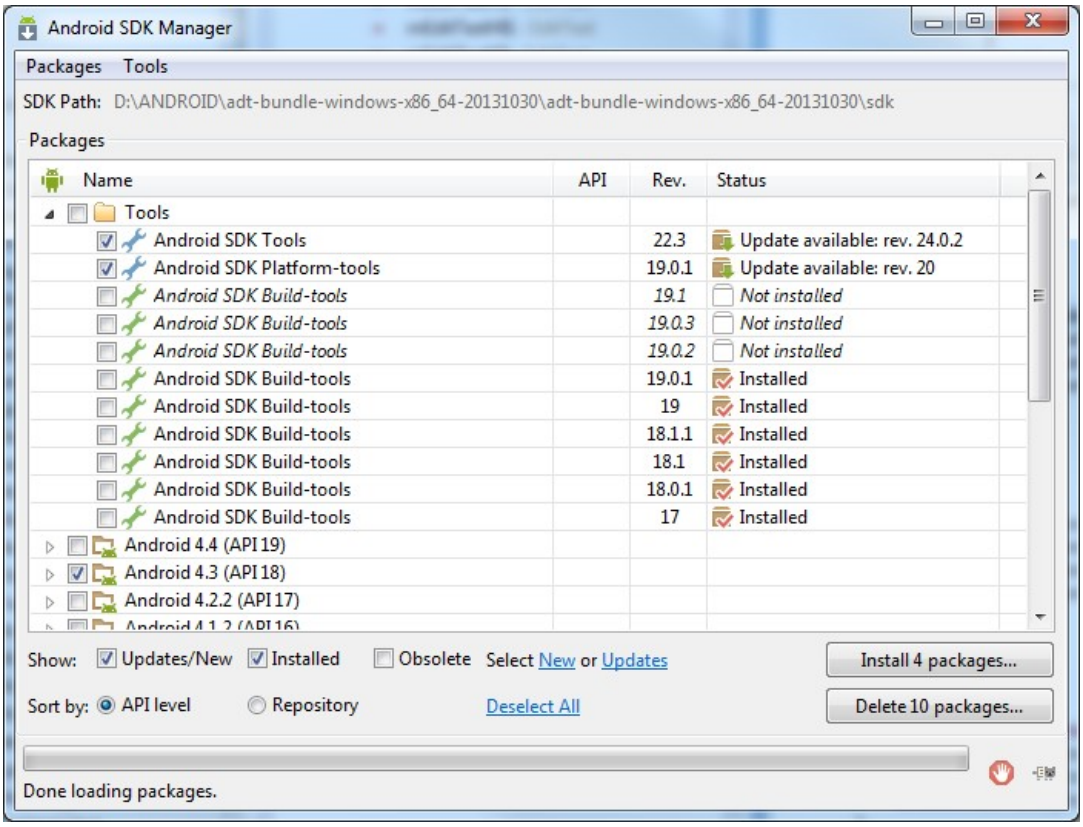
Przy pierwszym uruchomieniu program pyta o lokalizację folderu gdzie będą przechowywane pliki tworzonych aplikacji. Po wskazaniu ścieżki, zaznaczono opcję „Use this as default and do not ask again” (użyj tych ustawień i nie pytaj ponownie) i kliknięto przycisk OK.



Rysunek 39. Okno wyboru „Workspace”

Źródło: opracowanie własne

Po uruchomieniu się programu uruchomiono Android SDK Menager. Z paska menu programu Eclipse wybrano: Window > Android SDK Menager. Otworzyło się okno „Android SDK Manager”.



Rysunek 40. Okno „SDK Menager”

Źródło: opracowanie własne

Android SDK to zestaw narzędzi umożliwiający tworzenie aplikacji na platformę Android. Po otworzeniu Menagera wybrano pakiety:

- Android SDK Tools
- Android SDK Platform-Tools
- Android 4.3 API 18

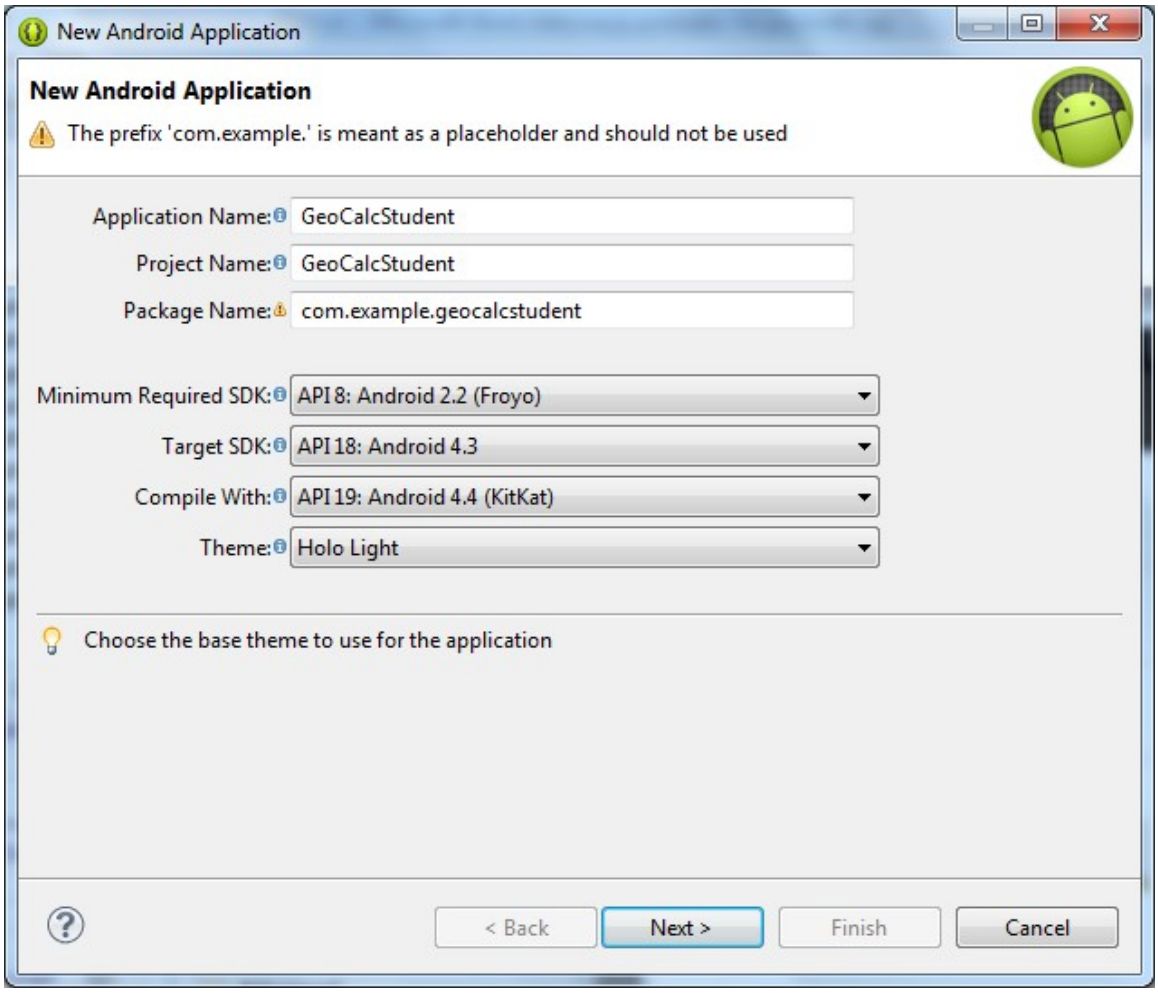
a następnie zainstalowano je klikając na przycisk „Install _ packages…”

Tym sposobem środowisko programistyczne zostało przygotowane do pracy.

2.2.2. TWORZENIE NOWEGO PROJEKTU

Po przygotowaniu środowiska programistycznego autor przystąpił do stworzenia nowego projektu.

Z paska menu wybrano File > New > Android Aplication Project. Uruchomił się kreator nowego projektu.

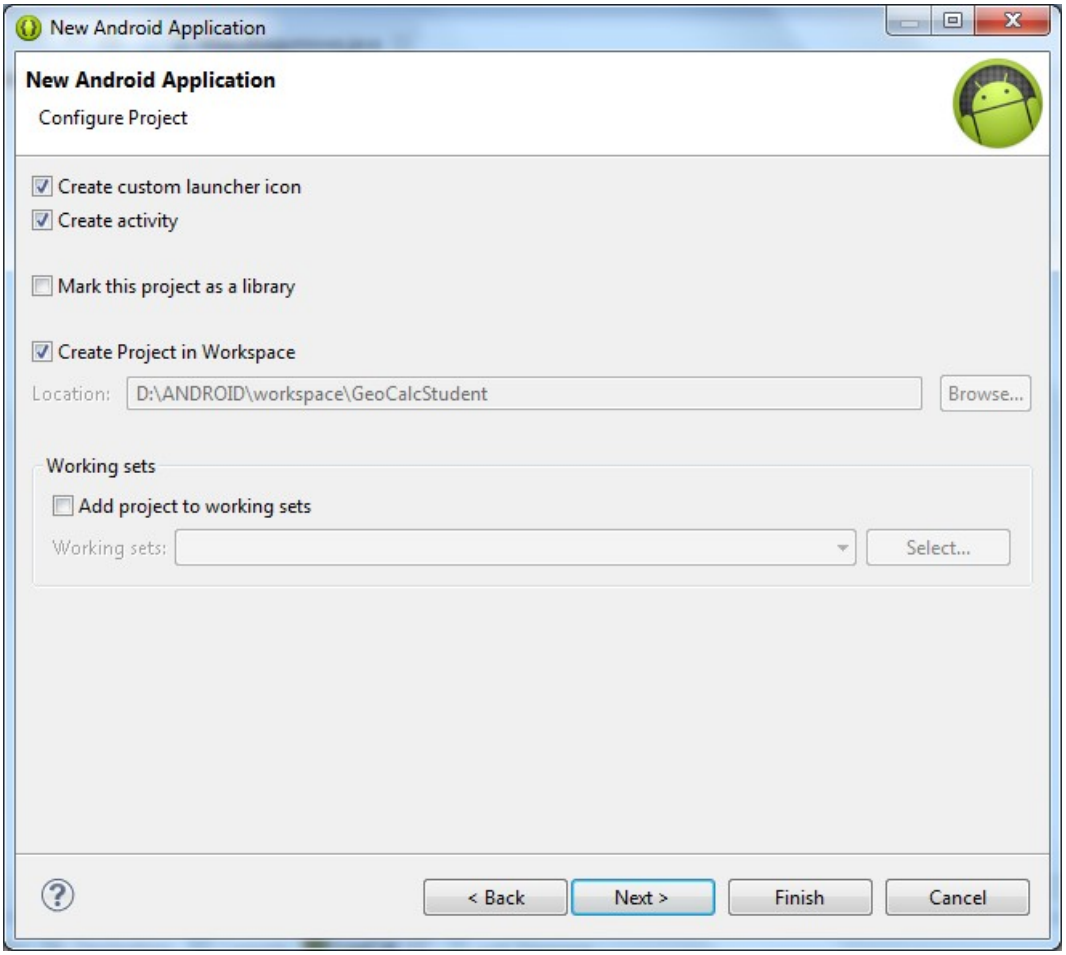


Rysunek 41. Okno kreatora

Źródło: opracowanie własne

W tym oknie autor wpisał nazwę aplikacji (Aplication Name), następnie określił minimalną wersję Android, jaka będzie wymagana do uruchomienia aplikacji (Minimum Required SDK), czyli Android 2.2 Froyo, wersje docelową – Android 4.3 (Target SDK), wersję Androida, do której będzie kompilowana aplikacja – Android 4.4 KitKat (Compile With) oraz temat (Theme).

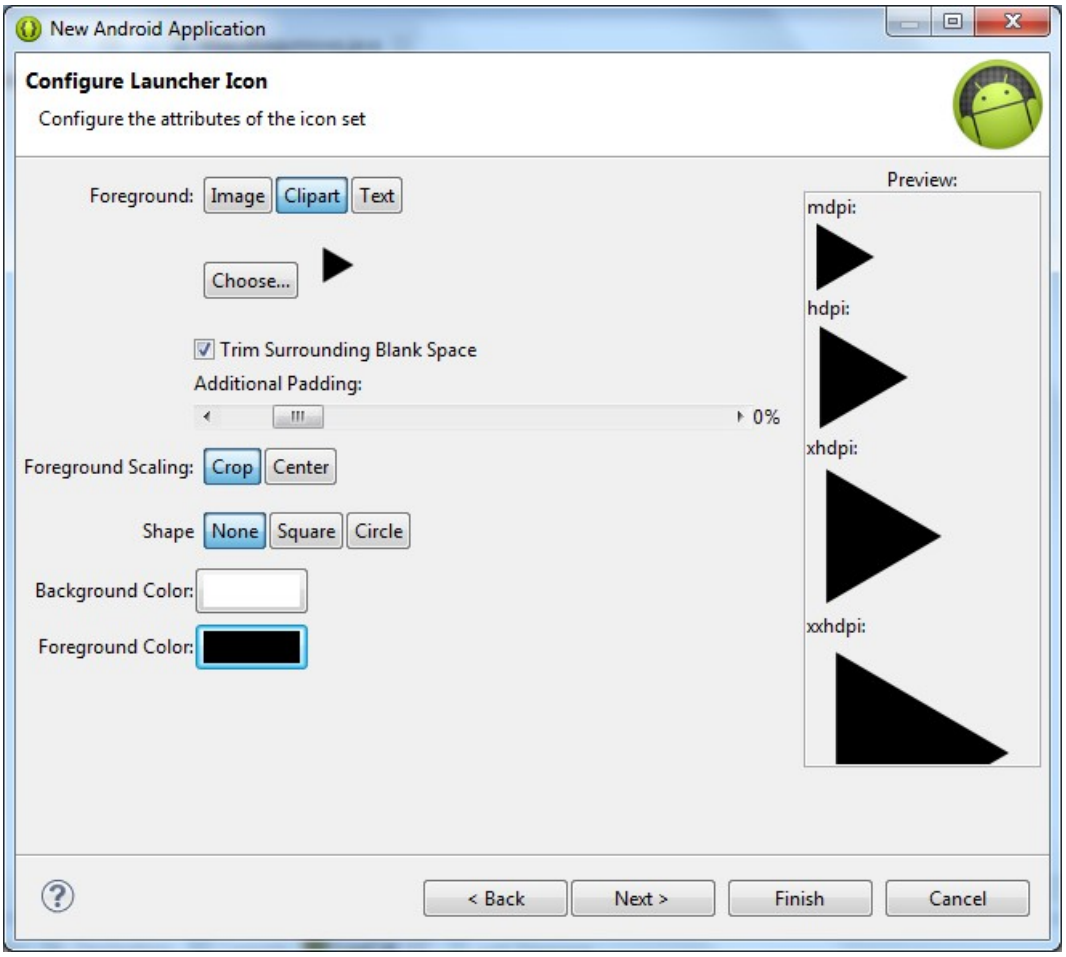
Po kliknięciu przycisku „Next” otworzyło się okno konfiguracji projektu.



Rysunek 42. Okno konfiguracji projektu

Źródło: opracowanie własne

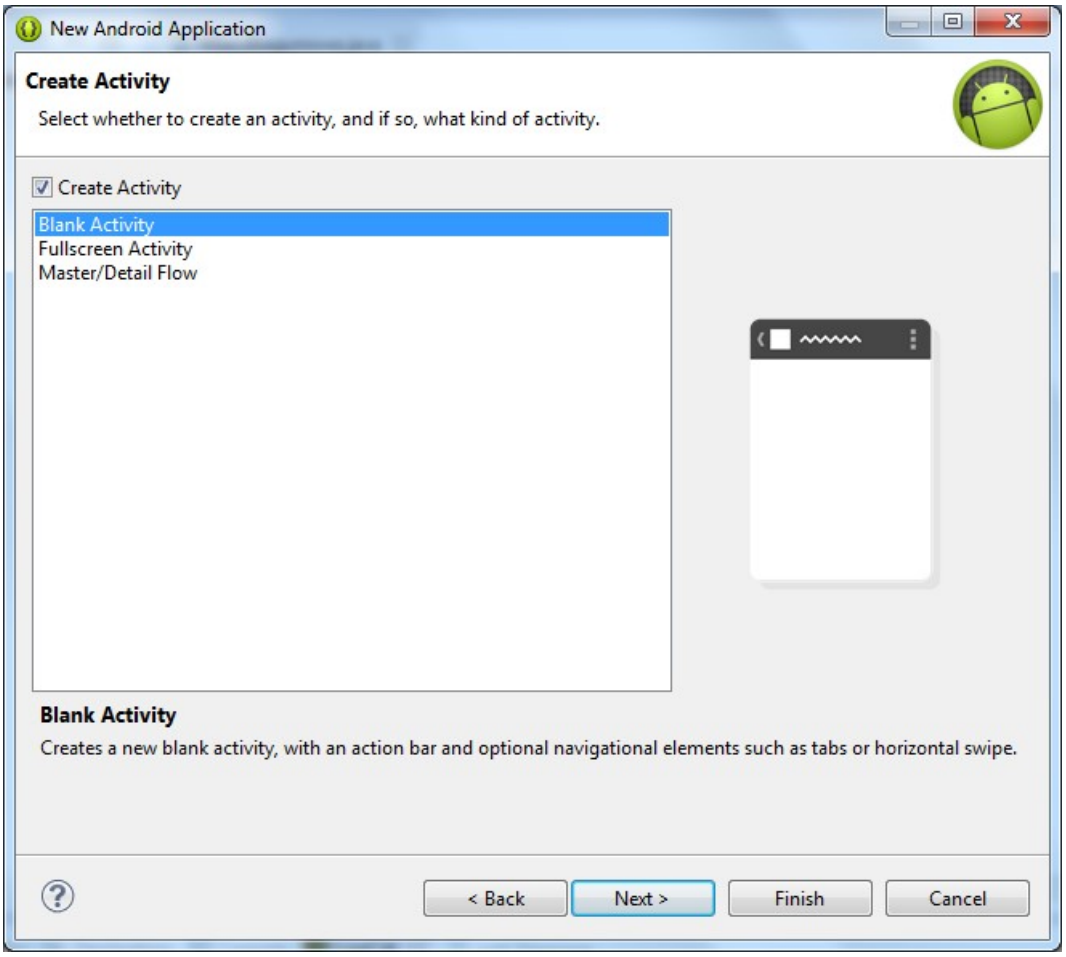
W tym oknie autor nie zmienił żadnych ustawień, nacisnął przycisk „Next” co spowodowało otwarcie okna konfiguracji ikony aplikacji (ikonę można zmienić również w późniejszym czasie).



Rysunek 43. Okno konfiguracji ikony

Źródło: opracowanie własne

W tym oknie autor wybrał ikone aplikacji i zatwierdził przyciskiem „Next". Otworzyło się kolejne okno, w którym tworzy się nową aktywność „New Activity"

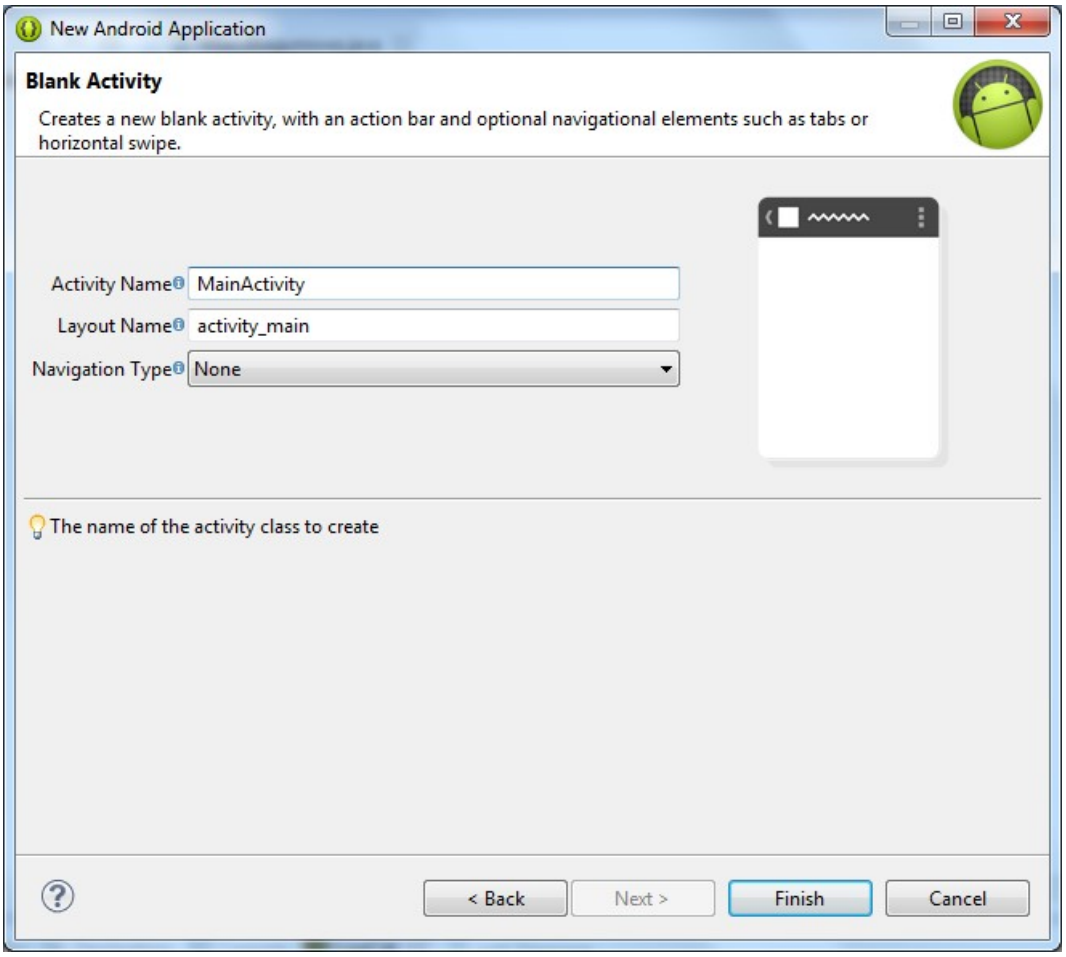


Rysunek 44. Okno tworzenia nowej aktywności

Źródło: opracowanie własne

Aktywność (Activity) to jeden z podstawowych komponentów systemu Android. Odpowiedzialny jest za interakcję z użytkownikiem, tworzenie okna aplikacji oraz uruchamianie innych podstawowych komponentów.

W tym oknie autor wybrał pustą aktywność „Blank Activity” i kliknął „Next”. Otworzyło się okno tworzenia nowej aktywności, w którym można wpisać nazwę aktywności. Nazwa aktywności musi zaczynać się wielką literą, nie może składać się ze spacji i znaków specjalnych.



Rysunek 45. Okno wyboru nazwy nowej aktywności

Źródło: opracowanie własne

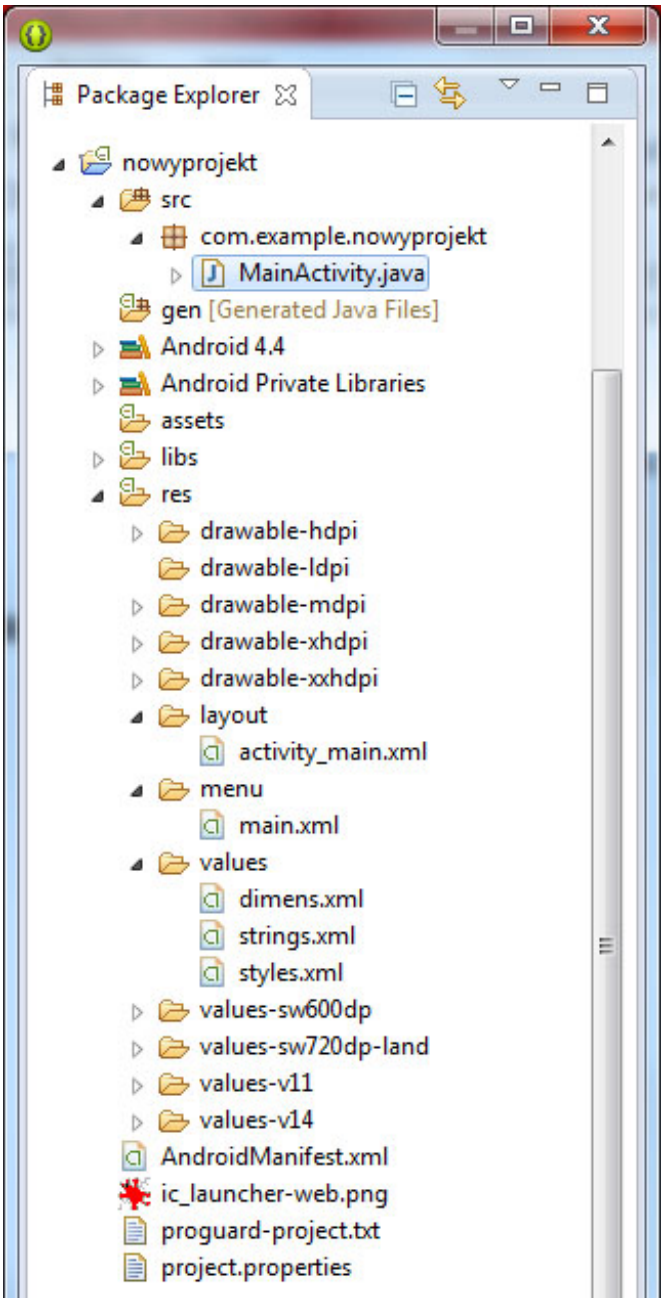
Po naciśnięciu „Finish” została utworzona nowa aktywność a autor przystąpił do programowania.

2.3. STRUKTURA APLIKACJI

Aplikacja składa się z trzech podstawowych typów elementów, którymi są:

- Komponenty – tworzą podstawową strukturę aplikacji, są kodem źródłowym aplikacji. Dzielą się na główne grupy:
  - Aktywności
  - Usługi – czynności wykonywane w tle.
  - Dostawcy treści – obiekty, które wymieniają informację pomiędzy różnymi aplikacjami
  - Powiadomienia – obiekty, które mogą wyświetlać informację w czasie działania aplikacji

- Zasoby – wszystkie elementy, które nie są kodem aplikacji takie jak pliki graficzne, multimedialne, ciągi znaków itp.
- Manifest XML – element, w którym przechowywana jest podstawowa konfiguracja aplikacji



Rysunek 46. Drzewo plików projektu

Źródło: opracowanie własne

- SRC – folder zawiera kod źródłowy aplikacji. W plikach .java znajduje się cała logika aplikacji.
- GEN – folder zawiera pliki niezbędne do poprawnego działania aplikacji generowane przez kompilator.
- ASSETS – folder zawiera dane aplikacji, pliki dowolnego formatu, które chcemy użyć w aplikacji.
- LIBS – folder zawiera biblioteki
- BIN – pliki binarne (kod bajtowy, archiwum APK aplikacji).
- RES – główny folder zawierający zasoby programu.
    - drawable – foldery zawierające pliki graficzne wykorzystywane w aplikacji.
    - layout – folder zawierający pliki .xml opisujące aktywności (widoki i rozkład UI).
    - menu – folder zawierający pliki .xml opisujące listy menu poszczególnych aktywności.
    - values – folder zawierający plik string.xml, w którym są przechowywane wszystkie teksty oraz pliki zawierające opis kolorów, wymiarów i stylów aplikacji.
- AndroidManifest.xml – plik deskryptora aplikacji. Definiuje wszystkie występujące w aplikacji aktywności, uprawnienia dla aplikacji, usługi, dostawców treści odbiorniki komunikatów, adresatów intencji.

2.4. KOD MODUŁU OBLICZENIOWEGO

Każdy moduł obliczeniowy aplikacji to oddzielna aktywność. Najważniejsze pliki w każdej aktywności to plik .java zawierający logiczną część modułu (to tu programujemy funkcję modułu, ustalamy warunki itd.) oraz plik .xml opisujący wygląd aktywności (to tu ustalamy, jakie elementy będą wyświetlane i w jaki sposób).

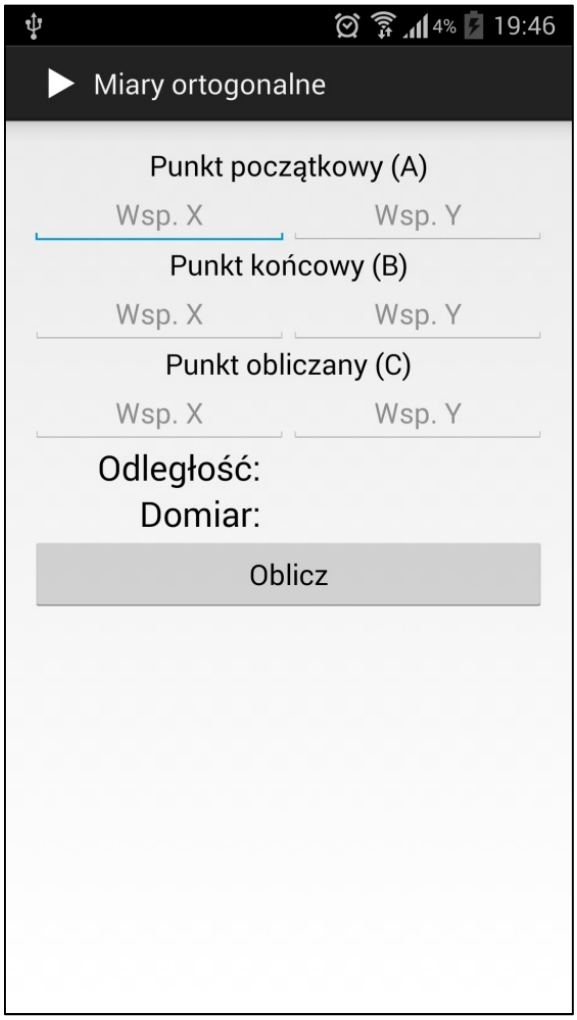
Plik activity_miary_ortogonalne.xml. Plik ten definiuje interfejs, są w nim opisane wszystkie element aktywności oraz ich właściwości. Gdy podczas tworzenia aktywności używamy interfejsu graficznego dostępnego w programie Eclipse plik .xml jest tworzony automatycznie. Można go w każdej chwili edytować lub napisać ręcznie.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MiaryOrtogonalne" >
    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" >
        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >
            <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center_horizontal" >
                <TextView
                    android:id="@+id/textViewop1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center_horizontal"
                    android:text="@string/punktpoczatkowy"
                    android:textAppearance="?android:attr/textAppearanceMedium" />
            </TableRow>
            <TableRow
                android:id="@+id/tableRow2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >
                <EditText
                    android:id="@+id/editText51"
                    android:nextFocusDown="@+id/editText52"
                    android:selectAllOnFocus="true"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:ems="10"
                    android:gravity="center_horizontal"
                    android:hint="@string/wspx"
                    android:inputType="numberDecimal" >
                    <requestFocus />
                </EditText>
                <EditText
                    android:id="@+id/editText52"
                    android:nextFocusDown="@+id/editText53"
                    android:selectAllOnFocus="true"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:ems="10"
                    android:gravity="center_horizontal"
                    android:hint="@string/wspy"
                    android:inputType="numberDecimal" />
```

```
</TableRow>
<TableRow
    android:id="@+id/tableRow3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal" >
    <TextView
        android:id="@+id/textViewop2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="@string/punktkoncowy"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</TableRow>
<TableRow
    android:id="@+id/tableRow4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <EditText
        android:id="@+id/editText53"
        android:nextFocusDown="@+id/editText54"
        android:selectAllOnFocus="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:gravity="center_horizontal"
        android:hint="@string/wspx"
        android:inputType="numberDecimal" />
    <EditText
        android:id="@+id/editText54"
        android:nextFocusDown="@+id/editText55"
        android:selectAllOnFocus="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:gravity="center_horizontal"
        android:hint="@string/wspy"
        android:inputType="numberDecimal" />
</TableRow>
<TableRow
    android:id="@+id/tableRow5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal" >
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="@string/punktobliczny"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</TableRow>
<TableRow
    android:id="@+id/tableRow6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <EditText
```

```
            android:id="@+id/editText55"
            android:nextFocusDown="@+id/editText56"
            android:selectAllOnFocus="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:gravity="center_horizontal"
            android:hint="@string/wspx"
            android:inputType="numberDecimal" />
        <EditText
            android:id="@+id/editText56"
            android:selectAllOnFocus="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:gravity="center_horizontal"
            android:hint="@string/wspy"
            android:inputType="numberDecimal" />
    </TableRow>
    <TableRow
        android:id="@+id/tableRow7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:gravity="right"
            android:text="@string/odleglosc"
            android:textAppearance="?android:attr/textAppearanceLarge" />
        <TextView
            android:id="@+id/textView55"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="left"
            android:text="@string/dwukropek"
            android:textAppearance="?android:attr/textAppearanceLarge" />
    </TableRow>
    <TableRow
        android:id="@+id/tableRow8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:gravity="right"
            android:text="@string/domiar"
            android:textAppearance="?android:attr/textAppearanceLarge" />
        <TextView
            android:id="@+id/textView57"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
```

```
                android:gravity="left"
                android:text="@string/dwukropek"
                android:textAppearance="?android:attr/textAppearanceLarge" />
        </TableRow>
        <TableRow
            android:id="@+id/tableRow9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
            <Button
                android:id="@+id/button51"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/oblicz" />
        </TableRow>
    </TableLayout>
</ScrollView>
</RelativeLayout>
```



Rysunek 47. Layout modułu „Miary ortogonalne”

Źródło: opracowanie własne

Plik MiaryOrtogonalne.java. Jest to plik kodu źródłowego Java.

```
package com.example.mgcs;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MiaryOrtogonalne extends Activity  implements OnClickListener{
	private EditText mEditTextX1;
	private EditText mEditTextY1;
	private EditText mEditTextX2;
	private EditText mEditTextY2;
	private EditText mEditTextX3;
	private EditText mEditTextY3;
	private TextView mTextO;
	private TextView mTextD;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_miary_ortogonalne);
		bindViews();
	}
	private void bindViews(){
		mEditTextX1 = (EditText) findViewById(R.id.editText51);
		mEditTextY1 = (EditText) findViewById(R.id.editText52);
		mEditTextX2 = (EditText) findViewById(R.id.editText53);
		mEditTextY2 = (EditText) findViewById(R.id.editText54);
		mEditTextX3 = (EditText) findViewById(R.id.editText55);
		mEditTextY3 = (EditText) findViewById(R.id.editText56);
		mTextO = (TextView) findViewById(R.id.textView55);
		mTextD = (TextView) findViewById(R.id.textView57);
		findViewById(R.id.button51).setOnClickListener(this);
	}
	@Override
	public void onClick(View view){
		if (view.getId() == R.id.button51){
			licz();
			return;
		}
	}
		public void licz(){
			if    (TextUtils.isEmpty(mEditTextX1.getText())
				|| TextUtils.isEmpty(mEditTextY1.getText())
				|| TextUtils.isEmpty(mEditTextX2.getText())
				|| TextUtils.isEmpty(mEditTextY2.getText())
				|| TextUtils.isEmpty(mEditTextX3.getText())
				|| TextUtils.isEmpty(mEditTextY3.getText())){
					Toast.makeText(this, "Wprowadź dane!",
Toast.LENGTH_SHORT).show();
```

```
                        }
                  else {
                        try{
                              double wspx1 =
Double.parseDouble(mEditTextX1.getText().toString());
                              double wspy1 =
Double.parseDouble(mEditTextY1.getText().toString());
                              double wspx2 =
Double.parseDouble(mEditTextX2.getText().toString());
                              double wspy2 =
Double.parseDouble(mEditTextY2.getText().toString());
                              double wspx3 =
Double.parseDouble(mEditTextX3.getText().toString());
                              double wspy3 =
Double.parseDouble(mEditTextY3.getText().toString());

                              double odlAB = 0;
                              double odlAC = 0;
                              odlAB = Math.sqrt(Math.pow((wspx1-wspx2), 2) +
Math.pow((wspy1 - wspy2), 2));
                              odlAC = Math.sqrt(Math.pow((wspx1-wspx3), 2) +
Math.pow((wspy1 - wspy3), 2));
                              if (odlAB != 0 ){
                                    double azymutAB = 0;
                                    azymutAB = (((Math.atan2((wspy2-wspy1),
(wspx2-wspx1))) *200) / Math.PI);
                                   if (azymutAB < 0){
                                          azymutAB = azymutAB + 400;
                                    }

                                    double azymutAC = 0;
                                    azymutAC = (((Math.atan2((wspy3-wspy1),
(wspx3-wspx1))) *200) / Math.PI);
                                    if (azymutAC < 0){
                                          azymutAC= azymutAC + 400;
                                    }

                                    double alfa = 0;
                                    alfa = (((azymutAC - azymutAB) * Math.PI )
/ 200);


                                    double odl = 0;
                                    double dom = 0;
                                    odl = Math.cos(alfa) * odlAC;
                                    dom = Math.sin(alfa) * odlAC;


                                    mTextO.setText(String.format(": " + "%.2f"
+ " m", odl));
                                    mTextD.setText(String.format(": " + "%.2f"
+ " m", dom));
                              }


                              else {
                                    Toast.makeText(this, "Błąd, ten sam
punkt!", Toast.LENGTH_SHORT).show();
                                    mTextO.setText(String.format(": ???"));
                                    mTextD.setText(String.format(": ???"));
```

```
                        }
                    }

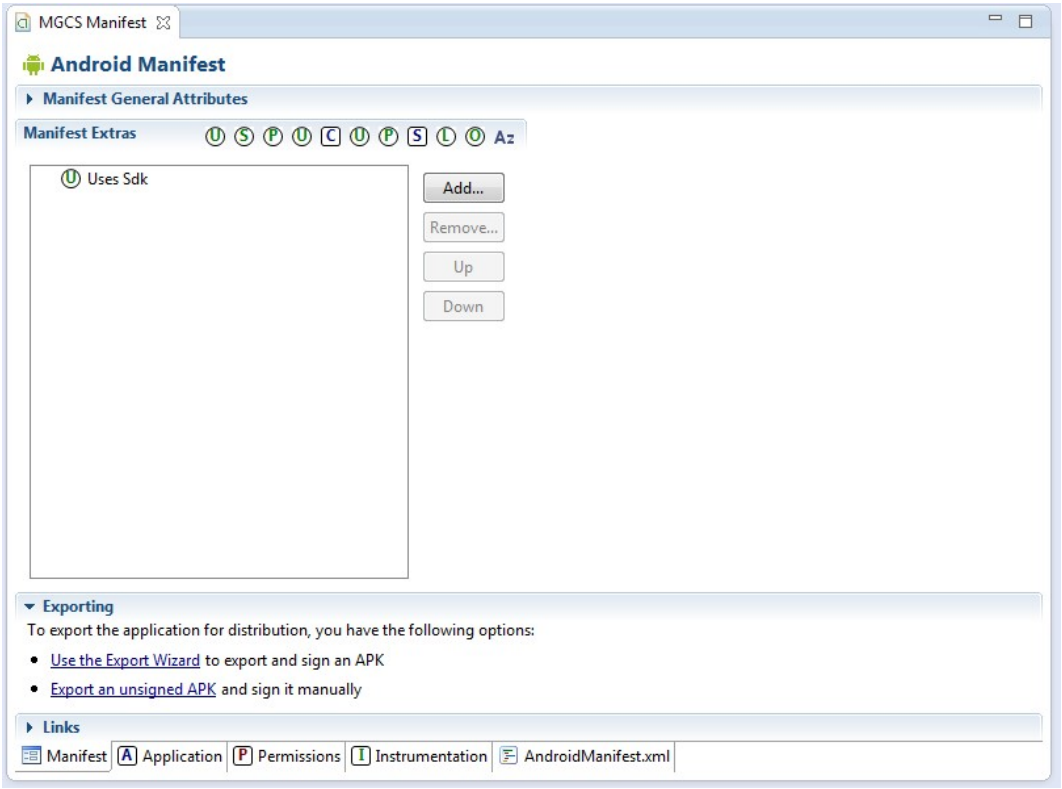
                    catch (NumberFormatException e){
                        Toast.makeText(this,  "Błędne współrzędne!",
Toast.LENGTH_SHORT).show();
                        mTextO.setText(String.format(": ???"));
                        mTextD.setText(String.format(": ???"));
                    }
                }
            }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.miary_ortogonalne, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.item1:
            startActivity(new Intent(MiaryOrtogonalne.this,
PomocMiaryOrtogonalne.class));
            break;
        }
            return true;
    }
}
```

2.5. KOMPILACJA I EKSPORT DO APK

Kompilacja to proces automatycznego tłumaczenia kodu źródłowego programu na kod wynikowy przez kompilator. Po skończeniu pisania aplikacji należy ją skompilować i wyeksportować do pliku, który będzie mógł być odczytany przez urządzenie.

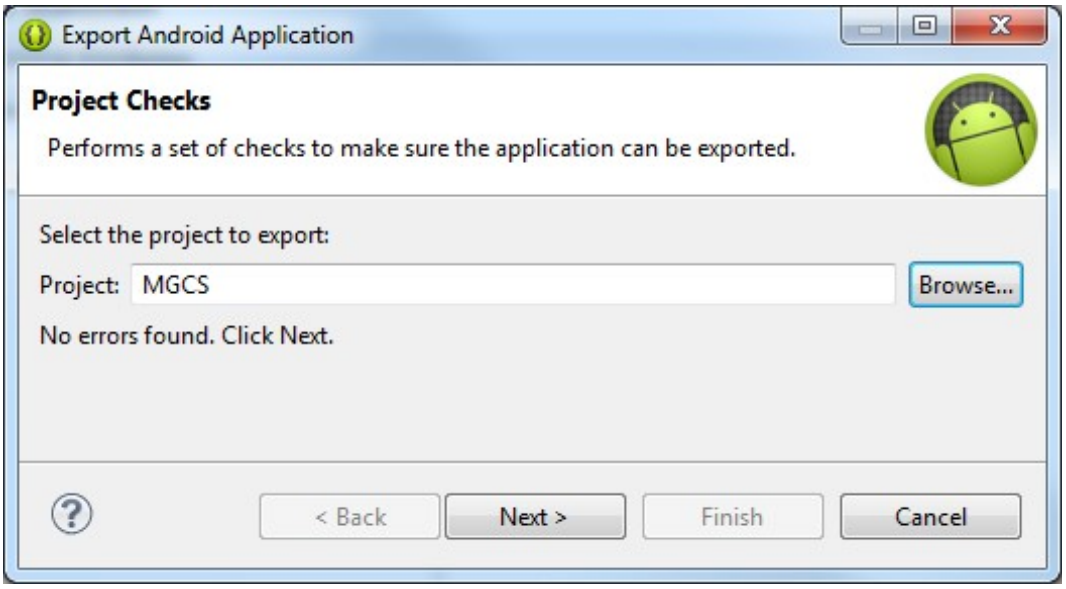
Po zakończeniu prac i przetestowaniu aplikacji autor przystąpił do eksportu aplikacji do pliku .apk, czyli takiego, który zostanie odczytany przez urządzenie z systemem android a następnie zainstalowany. Autor otworzył plik AndroidManifest.xml i wybrał zakładkę „Manifest".



Rysunek 48. Okno AndroidManifest

Źródło: opracowanie własne

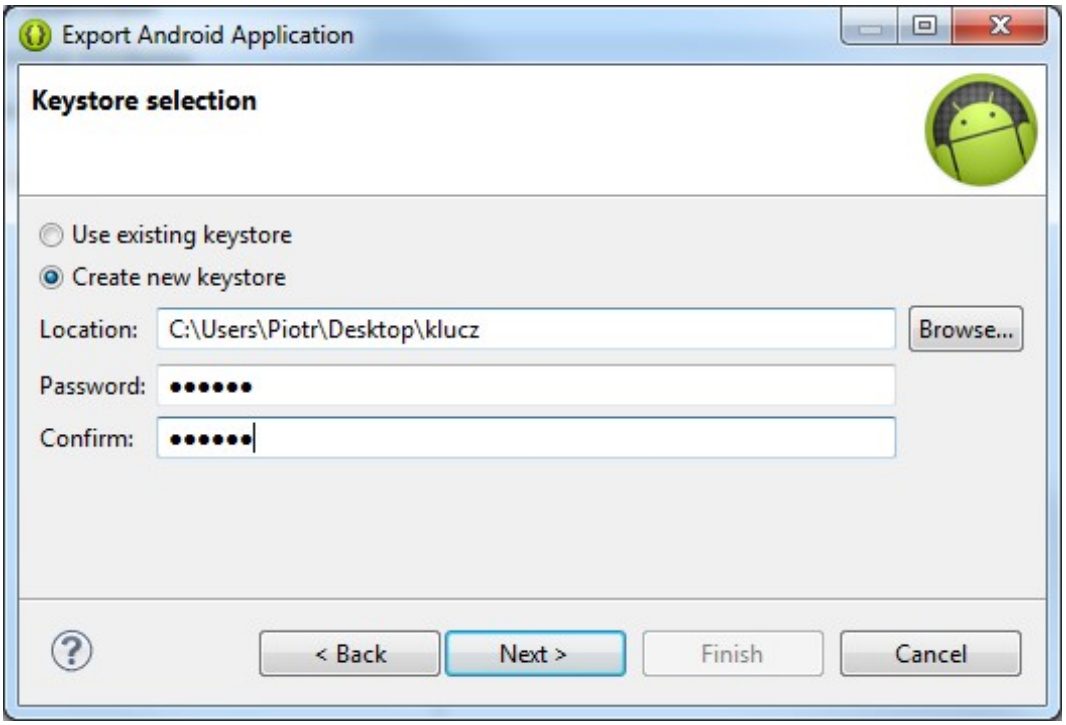
Następnie wybrał opcję „Use the Export Wizard" z grupy „Exporting". Otworzyło się okno „Export Android Aplication".



Rysunek 49. Okno eksportu aplikacji

Źródło: opracowanie własne

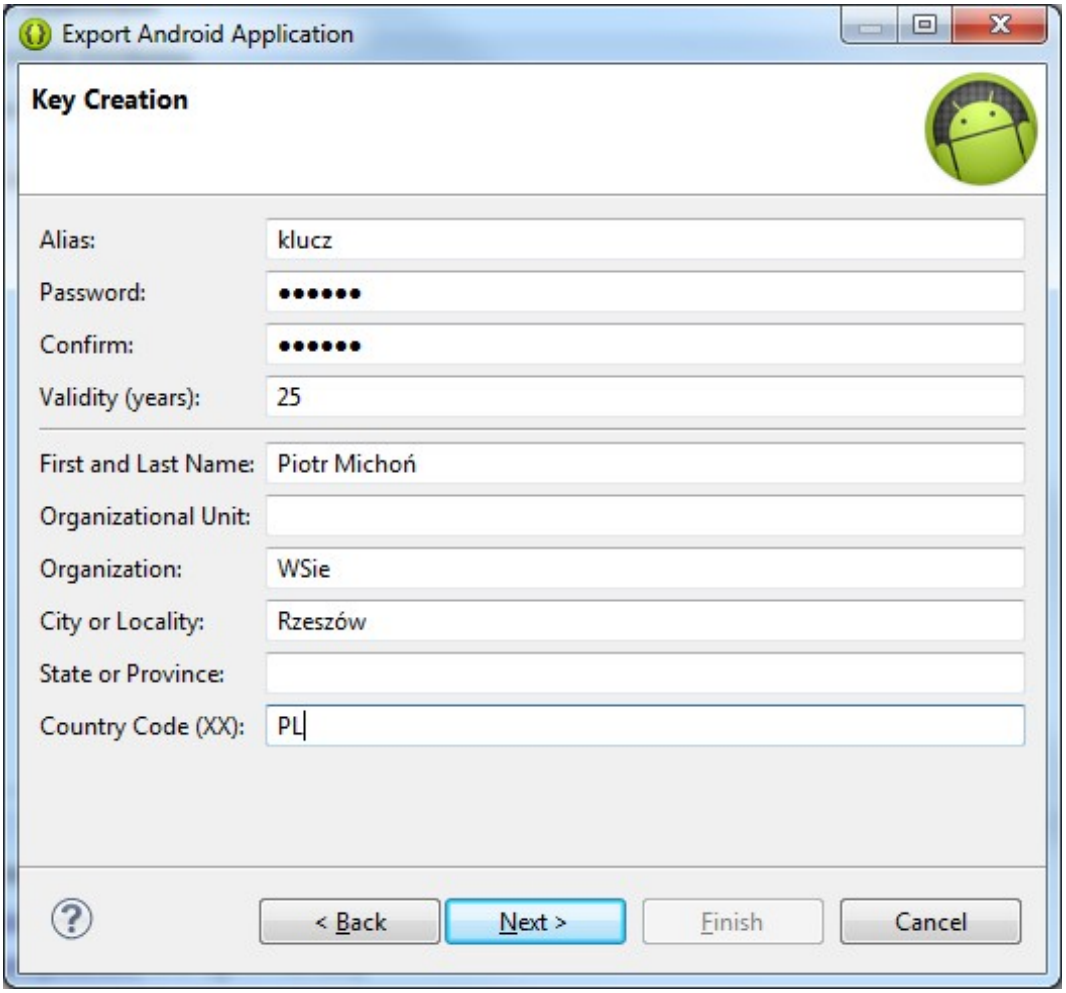
Po wciśnięciu przycisku „Next” otworzyło się okno, które pozwoli nam stworzyć klucz z certyfikatem dla aplikacji.



Rysunek 50. Okno tworzenia klucza

Źródło: opracowanie własne

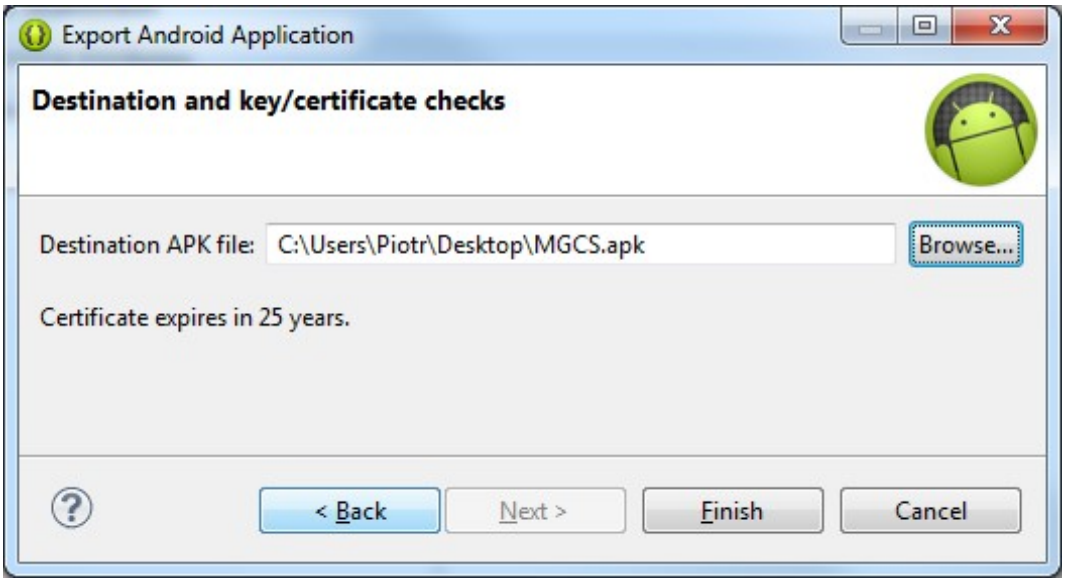
W oknie tym autor wskazał ścieżkę i nazwę pliku („Location”) oraz wpisał dwukrotnie hasło do klucza („Password” i „Confirm”). Po naciśnięciu przycisku „Next” otworzyło się kolejne okno, w którym należy wprowadzić alias klucza.



Rysunek 51. Okno tworzenia aliasu klucza

Źródło: opracowanie własne

W polu „Validity (years)” autor wpisał czas ważności klucza w latach, w pozostałych polach informację o autorze aplikacji, organizacji, mieście lokalizacji i kod kraju. Po naciśnięciu przycisku „Next” otworzyło się kolejne okno:



Rysunek 52. Okno eksportu aplikacji

Źródło: opracowanie własne

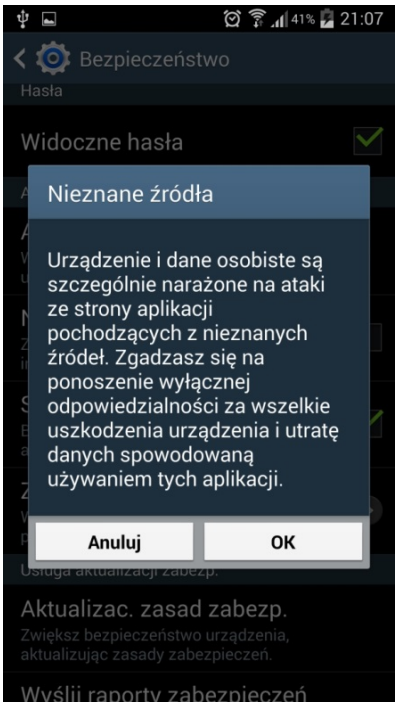
W oknie tym autor wskazał miejsce do którego został wyeksportowany plik .apk. Po naciśnięciu przycisku „Finish" aplikacja została wyeksportowana i zapisana we wskazanym miejscu.

2.6. WYMAGANIA APLIKACJI

Aplikacja do poprawnego działania wymaga urządzenia z zainstalowanym systemem minimum Android 2.2 Froyo.

2.7. INSTALACJA APLIKACJI NA URZĄDZENIU Z SYSTEMEM ANDROID

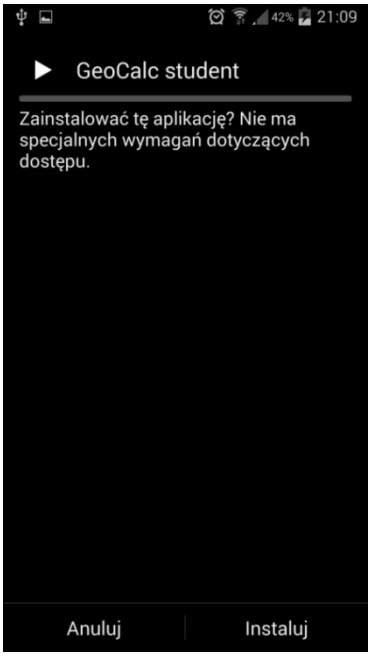
Nowostworzony plik GCS.apk autor skopiował na urządzenie z systemem android (Samsung Galaxy S4). Aby móc instalować aplikacje które nie pochodzą z Google Play należy ustawić w urządzeniu zgodę na instalację aplikacji z nieznanych źródeł. Należy wybrać w urządzeniu Ustawienia ->Więcej ->Bezpieczeństwo i zaznaczyć opcję „Nieznane źródła". Urządzenie wyświetli ostrzeżenie, należy je zaakceptować przyciskiem „OK."



Rysunek 53. Zrzut ekranu alertu nieznanych źródeł

Źródło: opracowanie własne

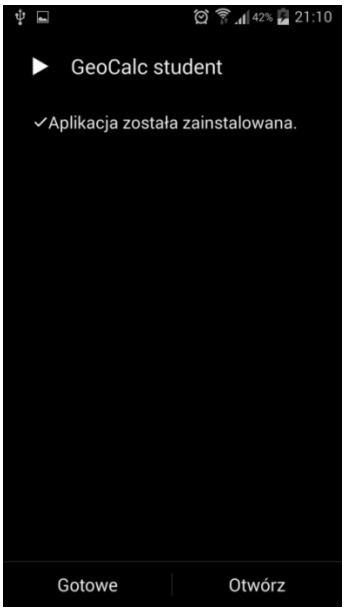
Po wykonaniu tych czynności autor w menadżerze plików urządzenia odnalazł plik GCS.apk i przystąpił do instalacji. Po kliknięciu na plik urządzenie poprosiło o potwierdzenie instalacji.



Rysunek 54. Zrzut ekranu okna instalacji aplikacji

Źródło: opracowanie własne

Po potwierdzeniu aplikacja została zainstalowana i jest gotowa do użycia.



Rysunek 55. Zrzut ekranu zakończenia instalacji

Źródło: opracowanie własne

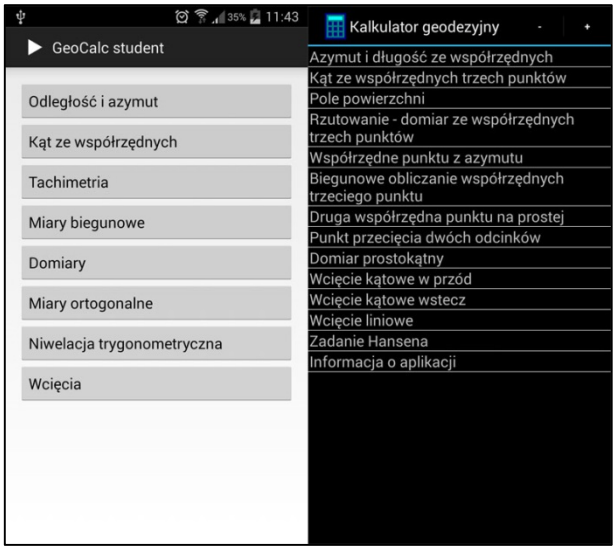
3. PODSUMOWANIA I WNIOSKI

3.1. PORÓWNANIE APLIKACJI DO INNYCH DOSTĘPNYCH NA RYNKU

W Google Play możemy znaleźć tylko jedna aplikację do obliczeń geodezyjnych, jest nią „Kalkulator Geodezyjny” firmy Geobid. Niestety, w opinii autora aplikacja ta posiada wiele wad, które czynią ją mało użyteczną podczas pracy w terenie.

3.1.1. LAYOUT

Layout aplikacji „Kalkulator Geodezyjny” jest utrzymany w ciemnych kolorach. Powoduje to, że nawet w dość lekkim nasłonecznieniu aplikacja jest całkowicie nieczytelna. Podczas działania aplikacji pasek statusu urządzenia jest ukryty, co uniemożliwia kontrolowanie stanu urządzenia.



Rysunek 56. Porównanie layoutu aplikacji

Źródło: opracowanie własne

3.1.2. FUNKCJONALNOŚĆ (MODUŁY OBLICZENIOWE)

Aplikacja „Kalkulator Geodezyjny” składa się z 13 modułów, „GeoCalcStudent” składa się z 11 modułów.

Tabela 3. Porównanie możliwości obliczeniowych aplikacji

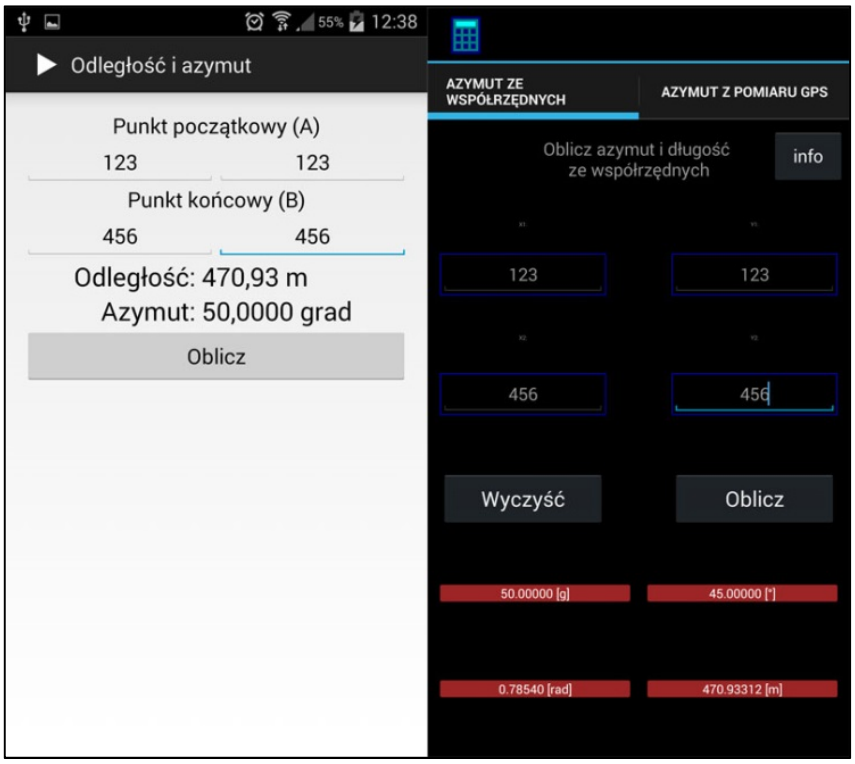
Zadanie obliczeniowe	Kalkulator Geodezyjny	GeoCalcStudent
Obliczenie odległości i azymutu	+	+
Obliczenie kata ze współrzędnych	+	+
Obliczenie współrzędnych metodą biegunową	+	+
Obliczenie danych do wyniesienia punktu metodą biegunową	-	+
Obliczenie współrzędnych metoda ortogonalną	-	+
Obliczenie danych do wyniesienia punktu metodą ortogonalną	+	+
Obliczenie wysokości punktów metodą niwelacji trygonometrycznej	-	+
Obliczenie wcięcia liniowego	+	+
Obliczenie wcięcia kątowego	+	+
Obliczenie wcięcia wstecz	+	+
Obliczenie wcięcia przestrzennego w przód	-	+
Obliczenie pola powierzchni	+	-
Obliczenie współrzędnych punktu z azymutu	+	-
Obliczanie drugiej współrzędnej na prostej	+	(możliwe)
Obliczanie współrzędnych przecięcia dwóch odcinków	+	-
Obliczenie domiaru prostokątnego	+	-
Obliczenie zadania Hansena	+	-

Źródło: opracowanie własne

Aplikacja „Kalkulator Geodezyjny„ posiada więcej modułów jednak wiele z nich jest nieużytecznych w pracy terenowej (np. zadanie Hansena) a dodatkowo brakuje funkcji bardzo użytecznych w terenie (np. obliczanie danych biegunowych do wyniesienia punktu).

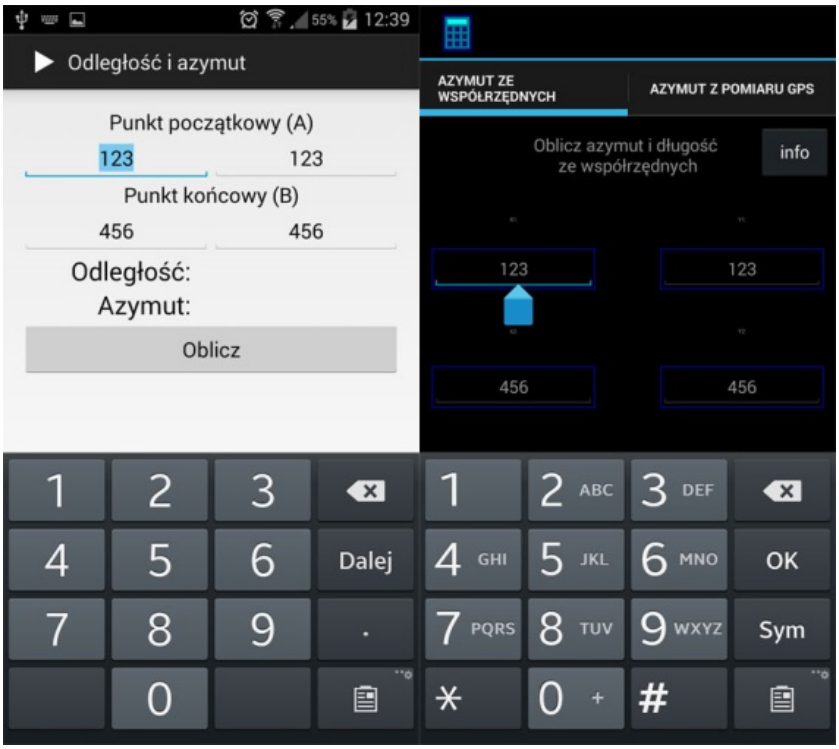
3.1.3. INTERFEJS UŻYTKOWNIKA

Aplikacja „Kalkulator Geodezyjny” jest bardzo niewygodna w użytkowaniu. Aplikacja działa w dwóch położeniach urządzenia przez, powoduje to problemy z przypadkową zmianą orientacji ekranu. Czcionka użyta w aplikacji jest bardzo mała i nieczytelna, w połączeniu z ciemnym layoutem jest praktycznie nieużyteczna w terenie. Elementy layoutu jak editText-y czy butony są małe, co utrudnia ich obsługę. Autorzy nie zadbali o wygodny sposób wprowadzania danych. Po wpisaniu danych do pierwszego editText należy nacisnąć klawisz „OK.” na klawiaturze ekranowej a następnie ręcznie wskazać następny editText. Przy próbie edycji wprowadzonych danych nie zaznacza się cały ciąg znaków, co powoduje dodatkowe trudności w zmianie we wprowadzonych danych. Pomoc do każdego modułu to mało czytelny obrazek o bardzo małym rozmiarze. Autor aplikacji „GeoCalcStudent” już na etapie projektowania aplikacji zadbał o to, aby jego aplikacja była pozbawiona takich niedogonień.



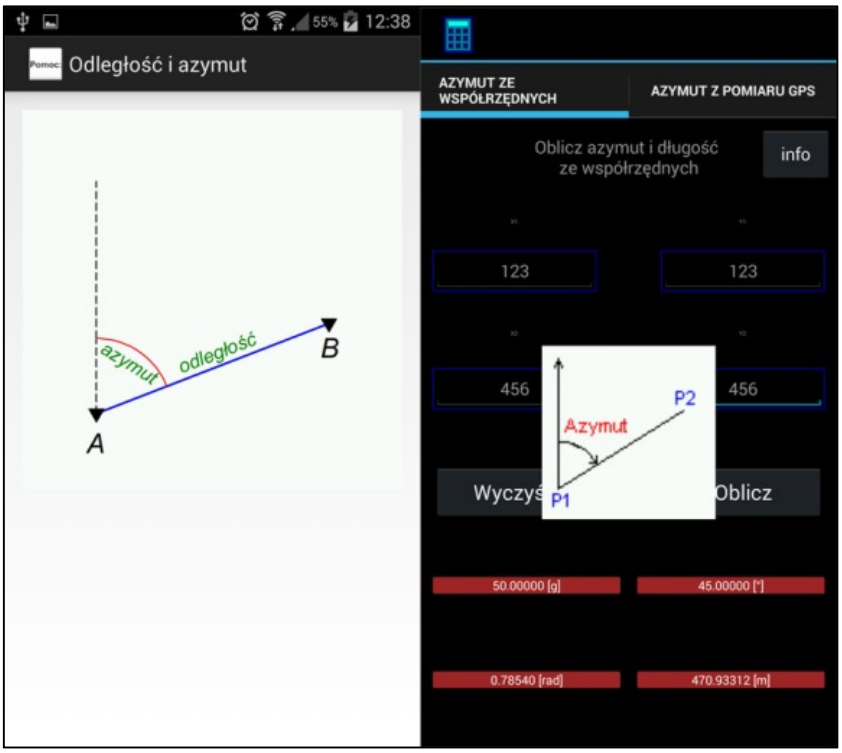
Rysunek 57. Porównanie layoutu modułu liczącego odległość i azymut

Źródło: opracowanie własne



Rysunek 58. Porównanie metody wprowadzania danych do obliczeń

Źródło: opracowanie własne



Rysunek 59. Porównanie pliku pomocy dołączonego do modułu obliczeniowego

Źródło: opracowanie własne

3.2. WNIOSKI

Po stworzeniu aplikacji autor udostępnił ją do testów studentom, geodetom oraz wykładowcom. Opinie wszystkich użytkowników były bardzo dobre. Przede wszystkim chwalono użyteczność aplikacji w terenie oraz jej intuicyjność i prostotę.

Stworzenie tej aplikacji było dla autora o tyle trudniejsze, że była to jego pierwsza aplikacja dla tego systemu. Wiązało się to z koniecznością przyswojenia wiedzy na temat środowiska programistycznego i nauki programowania w języku Java.

Dzięki doświadczeniu w pracy w terenie oraz wiedzy zdobytej podczas studiów powstała aplikacja, która pomoże w pracy geodety a także studentom geodezji.

Współczesne urządzenia z systemem Android mają bardzo dużą moc obliczeniową, często porównywalną ze stacjonarnymi komputerami. Idealnie nadają się do wspomagania pracy geodety. Doświadczenie i wiedza zdobyte podczas tworzenia tej aplikacji a także rozmowy z geodetami, studentami i wykładowcami pozwalają wyciągnąć wnioski, że można stworzyć profesjonalną aplikację do obliczeń geodezyjnych na urządzenia z systemem Android, na którą będzie duże zapotrzebowanie wśród wykonawców i studentów geodezji.

4. LITERATURA

Darwin I. *Android receptury.* HELION, Gliwice 2012, s. 315

Horstmann G. Cornell G. *JAVA™ Podstawy*, HELION, Gliwice 2014, s. 32

Jagielski A. *Geodezja I*. GEODPIS, Kraków 2002, s. 183

Jagielski A. *Geodezja II*. GEODPIS, Kraków 2003, s. 250; 255-257

Lehtimäki J. *Podręcznik dla projektantów Android UI*, HELION, Gliwice 2013, s. 19

STRONY INTERNETOWE:

http://pl.wikipedia.org/wiki/Wersje_systemu_Android [dostęp [28.12.2014]

http://mobirank.pl/2014/08/01/android-osiagnal-85-udzial-rynku-smartfonow/
[dostęp 28.12.2014]

5. SPIS TABEL, RYSUNKÓW I WZORÓW

RYSUNKI:

Rysunek 1. Udział mobilnych systemów operacyjnych na świecie w drugim kwartale 2014 roku.

Rysunek 2. Klawiatura pionowa (Android 4.4.2)

Rysunek 3. Klawiatura pozioma (Android 4.4.2)

Rysunek 4. Struktura aplikacji

Rysunek 5. Layout modułu „Odległość i azymut”

Rysunek 6. Diagram modułu „Odległość i azymut”

Rysunek 7. Odległość i azymut

Rysunek 8. Layout modułu „Kąt ze współrzędnych”

Rysunek 9. Diagram modułu „Kąt ze współrzędnych”

Rysunek 10. Kąt ze współrzędnych

Rysunek 11. Layout modułu „Tachimetria”

Rysunek 12. Diagram modułu „Tachimetria”

Rysunek 13. Tachimetria

Rysunek 14. Layout modułu „Miary biegunowe”

Rysunek 15. Diagram modułu „Miary biegunowe”

Rysunek 16. Miary biegunowe

Rysunek 17. Layout modułu „Domiary”

Rysunek 18. Diagram modułu „Domiary”
Rysunek 19. Domiary
Rysunek 20. Layout modułu „Miary ortogonalne”
Rysunek 21. Diagram modułu „Miary ortognalne”
Rysunek 22. Miary ortogonalne
Rysunek 23. Layout modułu „Niwelacja trygonometryczna”
Rysunek 24. Diagram modułu „Niwelacja trygonometryczna”
Rysunek 25. Niwelacja trygonometryczna
Rysunek 26. Layout modułu „Wcięcie liniowe”
Rysunek 27. Diagram modułu „Wcięcie liniowe”
Rysunek 28. Wcięcie liniowe
Rysunek 29. Layout modułu „Wcięcie kątowe”
Rysunek 30. Diagram modułu „Wcięcie kątowe”
Rysunek 31. Wcięcie kątowe
Rysunek 32. Layout modułu „Wcięcie wstecz”
Rysunek 33. Diagram modułu „Wcięcie wstecz”
Rysunek 34. Wcięcie wstecz
Rysunek 35. Kąty pomocnicze φ i ψ
Rysunek 36. Layout modułu „Wcięcie przestrzenne w przód”
Rysunek 37. Layout modułu „Wcięcie przestrzenne w przód”
Rysunek 38. Wcięcie przestrzenne w przód
Rysunek 39. Okno wyboru „Workspace”
Rysunek 40. Okno „SDK Menager”
Rysunek 41. Okno kreatora
Rysunek 42. Okno konfiguracji projektu
Rysunek 43. Okno konfiguracji ikony
Rysunek 44. Okno tworzenia nowej aktywności
Rysunek 45. Okno wyboru nazwy nowej aktywności
Rysunek 46. Drzewo plików projektu
Rysunek 47. Layout modułu „Miary ortogonalne”
Rysunek 48. Okno AndroidManifest
Rysunek 49. Okno eksportu aplikacji
Rysunek 50. Okno tworzenia klucza
Rysunek 51. Okno tworzenia aliasu klucza

Rysunek 52. Okno eksportu aplikacji
Rysunek 53. Zrzut ekranu alertu nieznanych źródeł
Rysunek 54. Zrzut ekranu okna instalacji aplikacji
Rysunek 55. Zrzut ekranu zakończenia instalacji
Rysunek 56. Porównanie layoutu aplikacji
Rysunek 57. Porównanie layoutu modułu liczącego odległość i azymut
Rysunek 58. Porównanie metody wprowadzania danych do obliczeń
Rysunek 59. Porównanie pliku pomocy dołączonego do modułu obliczeniowego

TABELE:

Tabela 1. Ewolucja języka JAVA™
Tabela 2. Zależności pomiędzy azymutem a czwartakiem
Tabela 3. Porównanie możliwości obliczeniowych aplikacji

WZORY:

Wzór 1. Wzór na obliczenie czawrtaka
Wzór 2. Wzór na obliczenie długości
Wzór 3. Wzór na obliczenie kąta płaskiego
Wzór 4. Wzór na obliczenie współrzędnej X
Wzór 5. Wzór na obliczenie współrzędnej Y
Wzór 6. Wzór na obliczenie azymutu boku SP
Wzór 7. Wzór na obliczenie kąta płaskiego
Wzór 8. Wzór na obliczenie długości
Wzór 9. Wzór na obliczenie współrzędnej X
Wzór 10. Wzór na obliczenie współrzędnej Y
Wzór 11. Wzór na obliczenie długości rzędnej
Wzór 12. Wzór na obliczenie długości odciętej
Wzór 13. Wzór na obliczenie wysokości stanowiska
Wzór 14. Wzór na obliczenie wysokości punktu
Wzór 15. Wzór na obliczenie współrzędnej X wcinanego punktu
Wzór 16. Wzór na obliczenie współrzędnej Y wcinanego punktu
Wzór 17. Wzór na obliczenie azymutu boku LW
Wzór 18. Wzór na obliczenie kąta pomiędzy bokami LP i LW
Wzór 19. Wzór na obliczenie współrzędnej X wcinanego punktu

Wzór 20. Wzór na obliczenie współrzędnej Y wcinanego punktu
Wzór 21. Wzór na obliczenie długości boku LW
Wzór 22. Wzór na obliczenie azymutu boku LW