AGH

Technika Mikroprocesorowa

Gesture Processing Library

Autorzy:
Michał Janiec
Bartosz Polnik

Spis treści

1 Temat **2**

2 Cel **2**

3 Opis zagadnienia **2**

4 Lista wspieranych gestów **3**

5 Instrukcja użytkownika **4**

6 Realizacja **5**

7 Implementacja **8**
7.1 gp_Main.h 8
7.1.1 Gesture definitions 8
7.1.2 Data structures 8
7.1.3 Functions 9
7.2 gp_Alloc.h 10
7.2.1 Constants 10
7.2.2 Functions 11
7.3 gp.h 11
7.4 gp_bool.h 11
7.4.1 Constants 11
7.5 gp_point.h 11
7.5.1 Data structures 12
7.5.2 Functions 12
7.6 gp_printf.h 12
7.7 gp_types.h 12
7.7.1 Data types 13
7.7.2 Constants 13
7.8 gp_vector.h 13
7.8.1 Data structures 13
7.8.2 Functions 14
7.9 gp_gestures_parameters.h 15
7.9.1 Constants 15
7.10 gp_gestures_results.h 16
7.10.1 Constants 16
7.11 gp_MotionEvent.h 16
7.11.1 Constants 16
7.12 gp_OutputGesture.h 16

7.12.1 Data structures 17
7.13 gp_Math.h 18
7.13.1 Constants 19
7.13.2 Functions 19

8 Pliki **23**

9 Podsumowanie **25**

10 Bibliografia **25**

1 Temat

Stworzenie niskopoziomowej biblioteki do przetwarzania gestów, dedykowanej dla mikroprocesorów jedno-układowych.

2 Cel

Celem projektu było przede wszystkim stworzenie ww. biblioteki pozwalającej na wygodne korzystanie z technologi multi-touch na różnorodnych urządzeniach. Ponadto utworzona została aplikacja na platformę Android służąca zaprezentowaniu działania biblioteki. Jej zadaniem jest odczytywanie gestów wykonanych przez użytkownika i wyświetlanie ich nazw.

3 Opis zagadnienia

Zadaniem biblioteki jest rozpoznawanie gestów. Biblioteka periodycznie dostawać będzie informacje o czasie oraz współrzędnych dotknięcia ekranu. Na tej podstawie porównuje ruch z rozpoznawanymi gestami i ustala odpowiednie zmienne informujące o tym, który ruch został wykonany. Użytkownik biblioteki powinien periodycznie sprawdzać czy jakiś ruch został rozpoznany i samodzielnie przetwarzać zawartość zmiennych przechowujących o nim dodatkowe informacje. Biblioteka została napisana w języku C bez użycia zewnętrznych bibliotek.

4 Lista wspieranych gestów

W celu uniknięcia niejednoznaczności proponujemy anglojęzyczne nazwy gestów.

Nazwa	Rysunek	Opis	Parametry
Tap		Pojedyncze stuknięcie w multi-touch.	Pozycja (x,y)
Two Finger Tap		Stuknięcie w multi-touch dwoma palcami.	Pozycja (x,y)
Press		Stuknięcie i przytrzymanie palca przez dłuższy czas.	Pozycja(x,y)
Move		Przesunięcie palca w dowolnym kierunku.	Pozycja(x,y) Pozycja(x,y)

Nazwa	Rysunek	Opis	Parametry
Rotate		Obrót w lewo lub w prawo.	Left/Right Obrót, (kąt)
Flick		Przesunięcie palca w lewo lub prawo i puszczenie.	Left/Right, Pozycja(x,y)
Scroll		Przesunięcie palca w górę lub w dół i puszczenie.	Up/Down, Pozycja(x,y)
Zoom		Przybliżenie lub oddalenie palca wskazującego i kciuka do siebie.	In/Out, Przybliżenie (liczba)
Two Finger Scroll		Przesunięcie dwóch palców równolegle w górę lub w dół.	Up/Down, Pozycja(x,y)

5 Instrukcja użytkownika

Aby korzystać z biblioteki należy dla każdego otrzymanego z ekranu zdarzenia utworzyć strukturę **gpMotionEvent** i wypełnić jej odpowiednie pola - kod akcji, współrzędne, czas oraz ilość palców. Kolejnym krokiem jest wywołanie funkcji **gpRecognize** z utworzoną przed chwilą strukturą. Biblioteka ustawia wartość zmiennych **gp_is*** żyjących w *gp_Main.h*. Jeśli interesują nas dodatkowe informacje o wykonanych ruchu, to można je znaleźć w zmiennych **gp_*Data**, również z *gp_Main.h*.

6 Realizacja

Ze względu, iż biblioteka nie wykorzystuje nic ponad możliwości czystego języka c, należało stworzyć sobie niezbędne środowisko umożliwiające komfortową pracę.

Pierwszym krokiem było przyjęcie kilku założeń - ze względu na różny rozmiar typów dostępnych na maszynie zdefiniowano własne typy danych na podstawie tych, które zapewniał język. Wiadomym jest, że rozmiar typu ma znaczenie np. przy rozpoznawaniu współrzędnych, więc stworzenie abstrakcyjnych typów danych pozwala na pewną niezależność od środowiska wykorzystania naszego produktu. Typy są deklarowane w *gp_types.h*, można je zmieniać w razie potrzeby.

Potrzebowaliśmy również przechowywać dane o trwającym ruchu. Ponieważ nie wiadomo, czy dany system pozwala na alokację pamięci, wydzieliliśmy fragment - *gp_Alloc*, który odpowiada za zapewnianie nam niezbędnej przestrzeni. Jeśli maszyna wspiera dynamiczną alokację, to można podmienić implementację funkcji *gpAlloc_alloc* oraz *gpAlloc_free* na delegację do odpowiednich rozwiązań systemowych.

Ze względów estetycznych i efektywnościowych zdecydowaliśmy stworzyć sobie strukturę danych przechowującą elementy oraz posiadającą funkcje potrafiące nią zarządzać. Nazwaliśmy ją vector i powołaliśmy do życia w plikach *gp_vector*.

Podczas tworzenia testów napotkaliśmy problem z nakładaniem się aliasów na wartości logiczne, dlatego zostały one wydzielone do pliku *gp_bool.h*.

Ważnych założeniem podczas projektowania było umożliwienie wykorzystania biblioteki przy jak najmniejszej ingerencji w utworzony przez klienta kod, co było powodem utworzenia *gp_MotionEvent* - definiującego dane otrzymywane od klienta(reprezentacja zdarzenia) oraz *gp_OutputGestures* - precyzyjnie definiującego otrzymywane informacje od biblioteki o formacie rozpoznanych gestów, nie wspominając już o próbie zachowywania pseudo przestrzeni nazw w tworzonych rozwiązaniach. Ponieważ w gestach pojawiają się informacje dotyczące np. kierunku ruchu utworzono odpowiednie stałe w *ge_gesture_results.h*. Na potrzeby wewnętrzne zadeklarowano również strukturę ułatwiającą reprezentacje punktu - *gp_point*.

Uznaliśmy też za istotne danie klientowi możliwości dostosowania zaproponowanych algorytmów do własnych potrzeb - można decydować o takich parametrach jak czas pomiędzy tap'em, a pressem, czy maksymalne odchylenie odległości przy wykonywaniu rotacji. Wszystko w *gp_gestures_parameters.h*.

Ostatnim elementem, który był po prostu niezbędny i bez którego nie udałoby się zrelizować założonej funkcjonalności był fragment odpowiedzialny

za matematykę. Pozwolę sobie przypomnieć, iż w założeniach był brak wykorzystania koprocesora (w celu zwiększenia kręgu możliwych odbiorców) konieczne było stworzenie jakiegoś zamiennika. Zamiennik ten pozwala na wykonywanie podstawowych operacji arytmetycznych poprzez odpowiednie funkcje z *gp_Math*. Sam moduł matematyki pozwala np. na liczenie siunusa, tangensa, potęgowanie, pierwiastkowanie(drugiego stopnia), arkusatangensa, a nawet obliczanie kąta pomiędzy dwoma punktami w przestrzeni $\Re^2$.

Warto jeszcze wspomnieć o algorytmach wykorzystywanych do decydowania o tym, czy coś jest odpowiednim gestem, czy też nim nie jest.

Zoom Implementacja alogrytmu znajduje się w pliku *gp_zoom.c*. Na początku obliczany jest dystans pomiędzy pierwszym i ostatnim odczytanym punktem, jeśli jest on mniejszy niż minimalne odchylenie dla gestu zoom - *GP_ZOOM_MIN_CHANGE*, to kończymy działanie i stwierdzamy, że nie jest to to, czego oczekiwaliśmy - gest zoom'em na pewno nie jest. Obliczamy kierunek na podstawie różnicy odległości punktów początkowych i końcowych - zbliżenie oznacza, że punkty końcowe są bliżej niż początkowe. Dalej sprawdzamy, czy ktoś przypadkiem nie wykonał dwoma palcami ruchu w jednym kierunku - jeśli to miało miejsce, to nie uznajemy tego za zoom. Warto zauważyć, iż prawie zawsze wykonujemy ruch dwoma palcami, stąd za ruch uznajemy coś, co pokonało dystans co najmniej *GP_TAP_MAX_MOVE*. Jedyne co nam jeszcze zostało to sprawdzenie, czy kąt pomiędzy punktami ruchu oraz punktem początkowym jest mniejszy niż $\pi/4$ oraz czy przypadkiem ktoś podczas ruchu nie zmienił kierunku. Każde nie spełnienie warunku jest zliczane i jeśli suma takich wypadków jest większa niż 1/3 wszystkich punktów to uznajemy, to za błędny ruch. Jeśli ruch po tych wszystkich sprawdzeniach dalej nie jest dorzucony, to uzupełniamy kierunek - zbliżenie/oddalenie, wartość zoom'u oraz ustawiamy flagę sugerującą, iż gest został poprawie rozpoznany.

Rotation Implementacja alogrytmu znajduje się w pliku *gp_rotation.c*. Pierwszym elementem sprawdzanym podczas rotacji ilość palców biorących udział w ruchu. Później obliczamy dystans pomiędzy pierwszym punktem ruchu dla pierwszego palca oraz ostatnim, i w drugim obliczeniu pomiędzy pierwszym punktem ruchu dla drugiego palca i ostatnim. Jeśli okaże się, iż oba dystanse są większe niż *GP_ROTATION_MAX_MOVE*, wtedy informujemy iż nie jest to gest rotacji. Następnie ustalamy, który palec nie wykonywał ruchu. Sprawdzamy, czy dystans pomiędzy tym palcem, a każdym punktem ruchu drugiego palca jest w normie, a jest

nią iloczyn *GP_ROTATION_DIST_PARAM* oraz odległości pomiędzy pierwszym punktem nieruchomego palca, a pierwszym punktem ruchomego palca. Każde niespełnienie warunku powoduje odrzucenie ruchu jako rotation. Teraz pozostaje nam jedynie obliczenie wykonanego kąta obrotu. W tym celu sumujemy różnice kątów pomiędzy kątami dla kolejnych punktów, a punktem bazowym (jest nim pierwszy punkt nieruchomego palca). Ponieważ dostajemy zawsze kąt dodatni, to uznajemy, iż kąt powyżej π jest ujemny. Na koniec moduł takiej sumy porównujemy z *GP_ROTATION_MIN_ANGLE* - minimalnym kątem obrotu. Jest to zarazem ostatni test ruchu. Jeśli jest pomyślny, to uzupełniamy dane o kącie: rozpoznanie gestu - *gp_isRotation*, na podstawie znaku sumy - kierunek kąta oraz jako kąt uznajemy moduł otrzymanej poprzednio sumy.

7 Implementacja

7.1 gp_Main.h

Udostępnia podstawowe funkcje realizowane przez bibliotekę

7.1.1 Gesture definitions

Zmienne zawierają szczegółowe informacje o wykonanym ruchu

- *gpOutputGesture_tap* gp_TapData
- *gpOutputGesture_press* gp_PressData
- *gpOutputGesture_flick* gp_FlickData
- *gpOutputGesture_move* gp_MoveData
- *gpOutputGesture_rotation* gp_RotationData
- *gpOutputGesture_scroll* gp_ScrollData
- *gpOutputGesture_zoom* gp_ZoomData
- *gpOutputGesture_two_finger_scroll* gp_TwoFingerScrollData
- *gpOutputGesture_two_finger_tap* gp_TwoFingerTapData

7.1.2 Data structures

gpRecognizeContext
: Przechowuje informacje o aktualnie wykonywanym ruchu

- *gpVector* finger1* przechowuje zbiór współrzędnych wykonywanego ruchu dla pierwszego palca
- *gpVector* finger2* przechowuje zbiór współrzędnych wykonywanego ruchu dla drugiego palca
- *gpByte fingers* przechowuje ilość placów biorących udział w ruchu
- *gpInt firstTime* przechowuje czas pierwszego dotknięcia ekranu w aktualnym ruchu

7.1.3 Functions

gpVoid gpRecognize(gpMotionEvent* event)
Odpowiada za rozpoznanie gestu

- *gpMotionEvent* event* event do przetworzenia

gpBool gpTryTap(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to tap

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryPress(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to press

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryFlick(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to flick

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryRotation(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to rotation

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryScroll(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to scroll

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryZoom(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to zoom

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryTwoFingerScroll(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to two finger scroll

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

gpBool gpTryTwoFingerTap(gpMotionEvent* event, gpRecognizeContext* context)
Sprawdza, czy aktualnie skończony ruch to two finger tap

- *gpMotionEvent* event* ostatni otrzymany event związany z ruchem
- *gpRecognizeContext* context* kontekst związany z ruchem

Warto zauważyć brak funkcji sprawdzającej, czy aktualnie skończony ruch to move. Wynika to z faktu, iż rozpoznawanie takiego ruchu następuje na bieżąco i informacja o tym ruchu dostępna jest nie tylko po otrzymaniu akcji dotyczącej oderwania ostatniego palca.

7.2 gp_Alloc.h

Odpowiada za zarządzanie pamięcią

7.2.1 Constants

- *gpAlloc_MAX_MEM* *1000000* przechowuje rozmiar bufora pamięci tymczasowej

7.2.2 Functions

gpVoid* gpAlloc_alloc(gpInt size)
Przydziela pamięć z bufora

- *gpInt size* ilość jednostek pamięci do przydzielenia

gpVoid gpAlloc_free(gpVoid* ptr)
Zwalnia poprzednio przydzieloną z bufora pamięć

- *gpVoid* ptr* wskaźnik na początek zaalokowanej poprzednio pamięci

gpVoid gpAlloc_copy(gpVoid* from, gpVoid* to, gpInt size)
Przekopiowuje dane pomiędzy from do to. Nie przydziela pamięci.

- *gpVoid* from* źródło danych do przeniesienia
- *gpVoid* to* lokacja do której dane mają być przeniesione
- *gpInt size* ilość jednostek do przeniesienia

7.3 gp.h

Zawiera użyteczne include'y plików nagłówkowych z folderu base

7.4 gp_bool.h

Definiuje podstawowe aliasy na wartości typu logicznego

7.4.1 Constants

- *false* *0*
- *true* *1*

7.5 gp_point.h

Zawiera funkcje i struktury dotyczące punktu

7.5.1 Data structures

gpPoint
Reprezentacja punktu

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

7.5.2 Functions

gpFloat gpPoint_distance(gpPoint* a, gpPoint* b)
Oblicza dystans pomiędzy dwoma punktami w przestrzeni

- *gpPoint* a* wskaźnik na pierwszy punkt
- *gpPoint* b* wskaźnik na drugi punkt

gpFloat gpPoint_distance2(gpPoint* a, gpPoint* b)
Oblicza kwadrat odległości pomiędzy punktami

- *gpPoint* a* wskaźnik na pierwszy punkt
- *gpPoint* b* wskaźnik na drugi punkt

gpPoint gpPoint_init(gpFloat x, gpFloat y)
Tworzy punkt o zadanych współrzędnych

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

7.6 gp_printf.h

Zawiera funkcie pomocne przy debugowaniu na platformie android

7.7 gp_types.h

Definiuje abstrakcję na typy zależne od platformy

7.7.1 Data types

- *typedef void gpVoid*
- *typedef char gpBool*
- *typedef unsigned char gpUByte*
- *typedef signed char gpByte*
- *typedef unsigned short gpUWord*
- *typedef signed short gpWord*
- *typedef unsigned long gpUInt*
- *typedef signed long gpInt*
- *typedef char gpChar*
- *typedef char* gpString*
- *typedef long gpFloat*

7.7.2 Constants

- *null ((gpVoid*)(0))* pomocna reprezentacja wskaźnika o wartości nieokreślonej

7.8 gp_vector.h

Opisuje abstrakcyjną kolekcję i operacje na niej

7.8.1 Data structures

gpVector
: Kolekcja

- *gpVoid** data* wskaźnik na pierwszy element kolekcji
- *gpInt capacity* aktualna pojemność kolekcji
- *gpInt size* aktualny rozmiar kolekcji

7.8.2 Functions

gpVoid gpVector_init(gpVector* self)
Inicjalizuje kolekcję (alokuje pamięć na jej elementy)

- *gpVector* self* wskaźnik na zaalokowaną kolekcję

gpVoid gpVector_destroy(gpVector* self)
Zwalnia pamięć zajmowaną przez kolekcję

- *gpVector* self* wskaźnik na kolekcję

gpInt gpVector_getSize(gpVector* self)
Zwraca rozmiar kolekcji

- *gpVector* self* wskaźnik na kolekcję

gpVoid* gpVector_at(gpVector* self, gpInt index)
Zwraca wskaźnik na element kolekcji

- *gpVector* self* wskaźnik na kolekcję
- *gpInt index* numer wskaźnika do elementu do zwrócenia

gpVoid gpVector_clean(gpVector* sefl)
Czyści zawartość kolekcji

- *gpVector* sefl* wskaźnik na kolekcję

gpVoid gpVector_pushBack(gpVector* self, gpVoid* what, gpInt size)
Dokłada element na koniec kolekcji

- *gpVector* self* wskaźnik na kolekcję
- *gpVoid* what* wskaźnik na element do dołożenia
- *gpInt size* rozmiar elementu dokładanego

gpVoid gpVector_popBack(gpVector* self, gpVoid* where, gpInt size)
Usuwa ostatni element z kolekcji

- *gpVector* self* wskaźnik na kolekcję
- *gpVoid* where* wskaźnik na miejsce w pamięci, w które ma być przeniesiony ostatni element kolekcji
- *gpInt size* rozmiar elementu kolekcji

7.9 gp_gestures_parameters.h

Opisuje parametry dotyczące tolerancyjności w wykrywaniu ruchów

7.9.1 Constants

- *GP_TAP_MAX_TIME 40* maksymalny czas tap'a
- *GP_TAP_MAX_MOVE gpMkFloat("12")* maksymalne odchylenie ruchu dla tapa
- *GP_ROTATION_DIST_PARAM gpMkFloat("0.35")* maksymalna różnica odległości (proporcjonalnie do odległości początkowej)
- *GP_ROTATION_MIN_ANGLE gpMkFloat("9")* minimalny kąt obrotu (stopnie)
- *GP_ROTATION_MAX_MOVE gpMkFloat("35")* maksymalne odchylenie podczas wykonywania rotacji
- *GP_TAP_PRESS_MOVE gpMkFloat("10")* maksymalnie odchylenie dla długiego przyciśnięcia
- *GP_SCROLL_MIN_LEN gpMkFloat("20")* minimalna odległość dla ruchu scroll
- *GP_FLICK_MIN_LEN gpMkFloat("15")* minimalna odległość dla ruchu flick
- *GP_TWO_FINGER_TAP_MAX_DIST gpMkFloat("60")* maksymalna odległość dla two finger tap
- *GP_NOTATIME -1* Wartość oznaczająca że czas jest nie ustawiony.
- *GP_ZOOM_MIN_CHANGE gpMkFloat("10")* minimalna odległość dla zoom

7.10 gp_gestures_results.h

Definiuje stałe pomocne przy rozpoznawaniu kierunków ruchu

7.10.1 Constants

- *GP_SCROLL_DOWN* *true* definiuje kierunek w dół
- *GP_SCROLL_UP* *false* definiuje kierunek w górę
- *GP_FLICK_LEFT* *false* definiuje ruch w lewo
- *GP_FLICK_RIGHT* *true* definiuje ruch w prawo
- *GP_ZOOM_IN* *true* definiuje przybliżenie
- *GP_ZOOM_OUT* *false* definiuje oddalenie

7.11 gp_MotionEvent.h

Określa znacznie kodu akcji w evencie

7.11.1 Constants

- *GP_ME_ACTION_DOWN* *0* początek ruchu
- *GP_ME_ACTION_MOVE* *2* ruch po ekranie
- *GP_ME_ACTION_POINTER_1_DOWN* *5* opuszczenie pierwszego palca na ekran
- *GP_ME_ACTION_POINTER_1_UP* *6* podniesienie pierwszego palca
- *GP_ME_ACTION_POINTER_2_DOWN* *261* opuszczenie drugiego palca
- *GP_ME_ACTION_POINTER_2_UP* *262* podniesienie drugiego palca
- *GP_ME_ACTION_UP* *1* koniec ruchu

7.12 gp_OutputGesture.h

Zawiera struktury reprezentujące dodatkowe informacje o ruchu

7.12.1 Data structures

gpOutputGesture_two_finger_scroll
Reprezentuje two finger scroll

- *gpFloat x* przesunięcie poziome
- *gpFloat y* przesunięcie pionowe
- *gpBool direction* kierunek przewinięcia

gpOutputGesture_zoom
Reprezentuje zoom

- *gpBool direction* kierunek
- *gpFloat magnification* stosunek odległości

gpOutputGesture_scroll
Reprezentuje scroll

- *gpFloat x* przesunięcie poziome
- *gpFloat y* przesunięcie pionowe
- *gpBool direction* kierunek przewinięcia

gpOutputGesture_flick
Reprezentuje flick

- *gpFloat x* przemieszczenie poziome
- *gpFloat y* przemieszczenie pionowe
- *gpBool direction* kierunek przemieszczenia

gpOutputGesture_rotation
Reprezentuje rotation

- *gpBool direction* informacja, czy kąt jest dodatni, czy ujemny
- *gpFloat angle* kąt obrotu

gpOutputGesture_move
Reprezentuje przesunięcie

- *gpFloat x* początkowa współrzędna x
- *gpFloat y* początkowa współrzędna y
- *gpFloat begx* końcowa współrzędna x
- *gpFloat begy* końcowa współrzędna y

gpOutputGesture_press
Reprezentuje press

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

gpOutputGesture_tap
Reprezentuje tap

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

gpOutputGesture_two_finger_tap
Reprezentuje two finger tap

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

7.13 gp_Math.h

Zawiera deklarację zaimplementowanych funkcji oraz stałych

7.13.1 Constants

- *GP_FLOAT_BASE 10000* część liczby przeznaczona na miejsca dziesiętne
- *gpMath_EPSILION 10* epsilon
- *gpMath_PI 31416* PI
- *gpMath_2PI 62832* dwukrotność PI
- *gpMath_PI2 15708* połowa liczy PI
- *gpMath_PI4 7854* PI przez 4
- *gpMath_PI6 5236* PI przez 6
- *gpMath_E 27183* E
- *gpMath_1 10000* 1
- *gpMath_2 20000* 2
- *gpMath_3 30000* 3
- *gpMath_SINPI4 7071* sinus PI przez 4
- *gpMath_0 0* 0

7.13.2 Functions

gpFloat gpMul(gpFloat a, gpFloat b)
Mnoży dwie liczy stałopozycyjne

- *gpFloat a* mnożna
- *gpFloat b* mnożnik

gpFloat gpDiv(gpFloat a, gpFloat b)
Dzieli dwie liczy stałopozycyjne

- *gpFloat a* dzielna
- *gpFloat b* dzielnik

gpFloat gpSub(gpFloat a, gpFloat b)
odejmuje dwie liczy stałopozycyjne

- *gpFloat a* odjemna
- *gpFloat b* odjemnik

gpFloat gpAdd(gpFloat a, gpFloat b)
Dodaje dwie liczy stałopozycyjne

- *gpFloat a* składnik
- *gpFloat b* składnik

gpFloat gpNeg(gpFloat a)
Zwraca liczbę przeciwną do danej liczy stałopozycyjnej

- *gpFloat a* operand

gpInt gpMath_MinInt()
Zwraca mniejszą z liczb typu gpInt

- *gpFloat a* operand
- *gpFloat b* operand

gpByte gpMath_Sign(gpFloat x)
Zwraca signum liczby

- *gpFloat x* operand

gpFloat gpMath_Abs(gpFloat a)
Zwraca wartość bezwzględną liczby

- *gpFloat a* operand

gpFloat gpMath_Square(gpFloat a)
Zwraca kwadrat liczby

- *gpFloat a* operand

gpFloat gpMath_Sqrt(gpFloat a)
Zwraca pierwiastek liczy

- *gpFloat a* operand

gpFloat gpMath_Exp(gpFloat a)
Zwraca E podniesione do potęgi a

- *gpFloat a* operand

gpFloat gpMath_PowI(gpFloat base, gpInt exp)
Potęguje

- *gpFloat base* liczba do podniesienia
- *gpInt exp* wykładnik

gpFloat gpMath_Sin(gpFloat x)
Oblicza sinus

- *gpFloat x* kąt w radianach

gpFloat gpMath_Cos(gpFloat x)
Oblicza kosinus

- *gpFloat x* kąt podany w radianach

gpFloat gpMath_Tan(gpFloat x)
Oblicza tangens

- *gpFloat x* kąt podany w radianach

gpFloat gpMath_ATan2(gpFloat x, gpFloat y)
Oblicza wartość bliższego kąta pomiędzy punktem, a osią x

- *gpFloat x* współrzędna x
- *gpFloat y* współrzędna y

gpFloat gpMath_ASin(gpFloat x)
Oblicza arcus-sinus

- *gpFloat x* kąt w radianach

gpFloat gpMath_ACos(gpFloat x)
Oblicza arcus-kosinus

- *gpFloat x* kąt podany w radianach

gpFloat gpMath_ATan(gpFloat x)
Oblicza arcus-tangens

- *gpFloat x* kąt podany w radianach

gpFloat gpMath_MinFloat(gpFloat a, gpFloat b)
Zwraca mniejszą z liczb

- *gpFloat a* operand
- *gpFloat b* operand

gpFloat gpMath_MaxFloat(gpFloat a, gpFloat b)
Zwraca większą z liczb

- *gpFloat a* operand
- *gpFloat b* operand

gpBool gpMath_Equals(gpFloat a, gpFloat b)
Porównuje dwie liczby

- *gpFloat a* operand
- *gpFloat b* operand

gpInt gpMath_Int(gpFloat a)
Zamienia liczbę stałopozycyjną na całkowitą

- *gpFloat a* liczba stałopozycyjna

gpFloat gpMath_FloatI(gpInt a)
Zwraca liczbę całkowitą na stałopozycyjną

- *gpInt a* liczba całkowita

gpFloat gpMath_AngleToAzimut(gpPoint a, gpPoint b)
Oblicza kąt skierowany między punktami

- *gpPoint a* pierwszy punkt
- *gpPoint b* drugi punkt

gpFloat gpMkFloat(gpString x)
Tworzy liczbę stałopozycyjną z napisu

- *gpString x* napis

8 Pliki

- *gp_Main.h* project_c\BaseProject\Include
- *gp_Alloc.h* project_c\BaseProject\Include\Alloc
- *gp.h* project_c\BaseProject\Include\Base
- *gp_bool.h* project_c\BaseProject\Include\Base
- *gp_point.h* project_c\BaseProject\Include\Base
- *gp_printf.h* project_c\BaseProject\Include\Base
- *gp_types.h* project_c\BaseProject\Include\Base
- *gp_vector.h* project_c\BaseProject\Include\Base
- *gp_gestures_parameters.h* project_c\BaseProject\Include\Gestures
- *gp_gestures_results.h* project_c\BaseProject\Include\Gestures

- *gp_MotionEvent.h* project_c\BaseProject\Include\InOut
- *gp_OutputGesture.h* project_c\BaseProject\Include\InOut
- *gp_Math.h* project_c\BaseProject\Include\Math
- *gp_Main.c* project_c\BaseProject\Source
- *gp_Alloc.c* project_c\BaseProject\Source\Alloc
- *gp.c* project_c\BaseProject\Source\Base
- *gp_point.c* project_c\BaseProject\Source\Base
- *gp_printf.c* project_c\BaseProject\Source\Base
- *gp_vector.c* project_c\BaseProject\Source\Base
- *gp_flick.c* project_c\BaseProject\Source\Gestures
- *gp_gestures.c* project_c\BaseProject\Source\Gestures
- *gp_press.c* project_c\BaseProject\Source\Gestures
- *gp_rotation.c* project_c\BaseProject\Source\Gestures
- *gp_scroll.c* project_c\BaseProject\Source\Gestures
- *gp_tap.c* project_c\BaseProject\Source\Gestures
- *gp_zoom.c* project_c\BaseProject\Source\Gestures
- *gp_MotionEvent.c* project_c\BaseProject\Source\InOut
- *gp_Math.c* project_c\BaseProject\Source\Math

9 Podsumowanie

Biblioteka realizuje założoną funkcjonalność. Podczas weryfikacji na aplikacji testowej zaskoczyła nawet samych twórców swoją dokładnością i precyzją. Nie posiada dużych wymagań sprzętowych i pamięciowych oraz jest bardzo elastyczna, jeśli chodzi o kontrolę czułości wykrywania gestów.

10 Bibliografia

- Notatki z wykładów dotyczących Metod Obliczeniowych w Nauce i Technice - `http://www.icsr.agh.edu.pl/~mownit/mownit.html`
- Wikipedia - `http://en.wikipedia.org/`
- Tutorial tworzenia aplikacji w ndk - `http://mobile.tutsplus.com/tutorials/android/ndk-tutorial/`
- Opis algorytmów podstawowych funkcji matematycznych - `http://mathonweb.com/help_ebook/html/algorithms.htm`
- Ściągawka z typami z jni - `http://dev.kanngard.net/Permalinks/ID_20050509144235.html`