



AGH

TECHNIKA MIKROPROCESOROWA

Gesture Processing Library

Autorzy:

Michał JANIEC

Bartosz POLNIK

Spis treści

1	Temat	2
2	Cel	2
3	Opis zagadnienia	3
4	Lista wspieranych gestów	3
5	Instrukcja użytkownika	5
6	Realizacja	6
6.1	gp.types - Typy danych	6
6.2	gp_Alloc - Dynamiczna alokacja pamięci	6
6.3	gp_Vector - Kontener danych	6
6.4	gp_Bool - Typy logiczny	6
6.5	Wejście, wyjście	6
6.6	Parametryzacja Algorytmów	7
6.7	gp_Math - matematyka	7
7	Algorytmy rozpoznające gesty	7
7.1	Move	8
7.2	Tap	8
7.3	Press	8
7.4	Two Finger Tap	9
7.5	Scroll	9
7.6	Two Finger Scroll	10
7.7	Flick	10
7.8	Zoom	10
7.9	Rotation	12
8	Implementacja	14
8.1	gp_Main.h	14
8.1.1	Gesture definitions	14
8.1.2	Data structures	14
8.1.3	Functions	15
8.2	gp_Alloc.h	16
8.2.1	Constants	16
8.2.2	Functions	17
8.3	gp.h	17
8.4	gp_bool.h	17

8.4.1	Constants	17
8.5	gp_point.h	17
8.5.1	Data structures	18
8.5.2	Functions	18
8.6	gp_printf.h	18
8.7	gp_types.h	18
8.7.1	Data types	19
8.7.2	Constants	19
8.8	gp_vector.h	19
8.8.1	Data structures	19
8.8.2	Functions	20
8.9	gp_gestures_parameters.h	21
8.9.1	Constants	21
8.10	gp_gestures_results.h	22
8.10.1	Constants	22
8.11	gp_MotionEvent.h	22
8.11.1	Constants	22
8.12	gp_OutputGesture.h	22
8.12.1	Data structures	23
8.13	gp_Math.h	24
8.13.1	Constants	25
8.13.2	Functions	25
9	Pliki	29
10	Podsumowanie	31
11	Bibliografia	31

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 technologii 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.





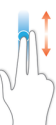
3 Opis zagadnienia

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 równoległe 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*. Triggerem do rozpoznawania gestów jest puszczenie obu palców (nie dotyczy gestu **Move** przetwarzanego w czasie rzeczywistym). Można to jednak zmienić wywołując funkcję rozpoznającą konkretny gest **gpTry*** w innej sytuacji. (Wewnątrz funkcji *gpRecognize*).

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.

6.1 `gp_types` - Typy danych

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.

6.2 `gp_Alloc` - Dynamiczna alokacja pamięci

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.

6.3 `gp_Vector` - Kontener danych

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*.

6.4 `gp_Bool` - Typy logiczny

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

6.5 Wejście, wyjście

Ważnym 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 reprezentację punktu - *gp_point*.

6.6 Parametryzacja Algorytmów

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*.

6.7 gp_Math - matematyka

Ostatnim elementem, który był po prostu niezbędny i bez którego nie udałooby się zrealizować założonej funkcjonalności był fragment odpowiedzialny za matematykę. Pozwolę sobie przypomnieć, iż w założeniach był brak wykorzystania koprocatora (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 sinususa, tangensa, potęgowanie, pierwiastkowanie (drugiego stopnia), arkusatangensa, a nawet obliczanie kąta pomiędzy dwoma punktami w przestrzeni \mathbb{R}^2 . Potencjalny brak koprocatora indukował brak typów zmiennoprzecinkowych. Dlatego wprowadziliśmy własny typ (*gpFloat* z *gp_Types*) realizujący arytmetykę stałoprzecinkową.

7 Algorytmy rozpoznające gesty

Algorytmy sprowadzają się do sprawdzenia serii założeń dotyczących konkretnych gestów. Jeśli wszystkie założenia są spełnione uznajemy że gest został wykonany.

7.1 Move

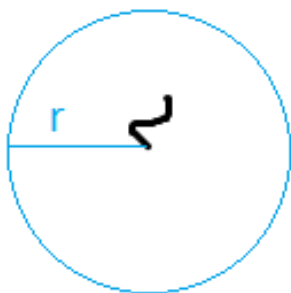
Implementacja znajduje się w pliku *gp_Main.c*. Gest nieco różni się od pozostałych: jest przetwarzany w czasie rzeczywistym.

- Warunkiem jego rozpoznania jest wykonanie dowolnego ruchu **jednym** palcem.
- Po rozpoznaniu zmienna **gp_isMove** ustawiana jest na true, struktura **gp_MoveData** wypełniana jest początkową, poprzednią i bieżącą pozycją palca.

7.2 Tap

Implementacja znajduje się w pliku *gp_tap.c*. Najprostszy gest, algorytm:

- Różnica między czasem rozpoczęcia gestu a jego zakończeniem musi być mniejsza niż *GP_TAP_MAX_TIME*.
- Podczas całego gestu palec nie może się oddalić o więcej niż $r = GP_TAP_MAX_MOVE$ od swojego początkowego położenia.
- Po rozpoznaniu flaga **gp_isTap** ustawiana jest na true, struktura **gp_TapData** wypełniana jest współrzędnymi tap'a.



7.3 Press

Implementacja w pliku *gp_press.c*.

Algorytm niemal identyczny jak **Tap**.

- Czas wykonania gestu musi być nie krótszy niż *GP_TAP_MAX_TIME*,
- Odsunięcie od środka nie większe niż *GP_PRESS_MAX_MOVE*.
- Po rozpoznaniu flaga **gp_isPress** ustawiana jest na true, struktura **gp_PressData** wypełniana jest współrzędnymi press'a.

7.4 Two Finger Tap

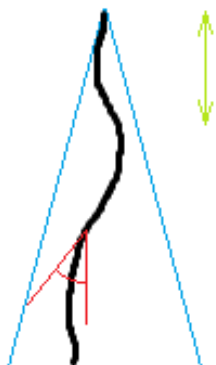
Implementacja w pliku *gp_tap.c*. Algorytm zbliżony do **tap**.

- Czas trwania gestu (licząc od położenia pierwszego palca do zdjęcia obu), musi być mniejszy niż *GP_TAP_MAX_TIME*,
- Każdy z palców z osobna musi spełnić warunek nie wyjścia z poza okręgu o promieniu *GP_TAP_MAX_MOVE*
- Po rozpoznaniu flaga **gp_isTwoFingerTap** ustawiana jest na true, struktura **gp_TwoFingerTapData** wypełniana jest współrzędnym tap'a (środek po między palcami).

7.5 Scroll

Implementacja znajduje się w pliku *gp_scroll.c*. Do rozpoznania następujące warunki muszą być zachowane:

- Pierwszy i ostatni punkt ruchu muszą różnić się wzdłuż osi *y* o co najmniej *GP_SCROLL_MIN_LEN*
- ostatni punkt ruchu może wychylić się w poziomie o maksymalnie 1/3 całego ruchu w pionie.
- styczna do ruchu w każdym punkcie musi tworzyć z osią *y* kąt mniejszy niż $\arctan(\frac{3}{2})$
- ponieważ przy opuszczaniu/podnoszeniu palca mogą powstać "zanieczyszczenia" w ruchu, z powyższego warunku zwolnione są pierwsze i ostatnie dwa punkty gestu.
- Po rozpoznaniu flaga **gp_isScroll** ustawiana jest na true, struktura **gp_ScrollData** wypełniana jest kierunkiem (up/down) i współrzędnymi palca w chwili zakończenia ruchu.



7.6 Two Finger Scroll

Implementacja znajduje się w pliku *gp_scroll.c*.

- Sprawdzeniu podlegają warunki jak dla **Scroll**, dla każdego palca z osobna. Jeśli ruchy obu palców zostaną rozpoznane jako **Scroll**, i będą to ruchy w tym samym kierunku, to biblioteka rozpozna **Two Finger Scroll** w tym właśnie kierunku.
- Po rozpoznaniu flaga **gp_isTwoFingerScroll** ustawiana jest na true, struktura **gp_TwoFingerScrollData** wypełniana jest kierunkiem (up/down) i współrzędnymi palców w chwili zakończenia ruchu (środek).

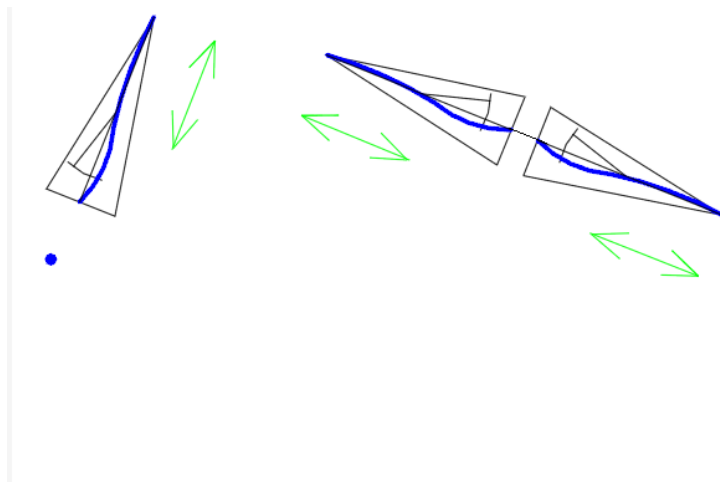
7.7 Flick

Implementacja znajduje się w pliku *gp_flick.c*.

- Implementacja identyczna jak dla **Scroll**'a jedynie osie X, Y są ze sobą zamienione.
- Wykorzystane jest stała **GP_FLICK_MIN_LEN**.
- Po rozpoznaniu flaga **gp_isFlick** ustawiana jest na true, struktura **gp_FlickData** wypełniana jest kierunkiem (left/right) i współrzędnymi palca w chwili zakończenia ruchu.

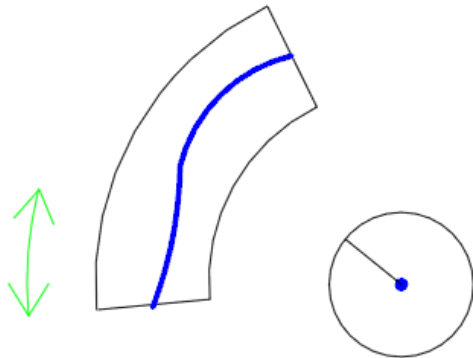
7.8 Zoom

Implementacja algorytmu znajduje się w pliku *gp_zoom.c*.



- Obliczane są dwie wartości: początkowa odległość palców, końcowa odległość palców. Jeśli różnią się od siebie o mniej niż `GP_ZOOM_MIN_CHANGE` to gest jest odrzucany.
- Obliczamy kierunek (in/out) na podstawie obliczonych wcześniej wartości - 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 odrzucamy.
- Aby ruch został zaliczony jako zoom sprawdzamy czy co najmniej jeden palec pokonał dystans nie mniejszy niż `GP_TAP_MAX_MOVE`
- Na podstawie punktów krańcowych - tych bardziej odległych, wyznaczana jest oś gestu.
- Aby sprawdzić czy ktoś przypadkiem nie zmienił kierunku sprawdzamy czy kąt między styczną a osią jest mniejszy niż $\pi/4$. 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.

7.9 Rotation



w pliku *gp_rotation.c*.

Implementacja algorytmu znajduje się

- Pierwszym elementem sprawdzanym podczas rotacji ilość palców biorących udział w ruchu.
- Później obliczamy dystans pomiędzy pierwszym oraz ostatnim punktem ruchu dla pierwszego jak i drugiego palca. Jeśli okaże się, iż oba dystanse są większe niż *GP_ROTATION_MAX_MOVE*, wtedy informujemy iż nie jest to gest rotacji (brak osi obrotu).
- Następnie ustalamy, który palec nie wykonywał ruchu (oś obrotu).
- Sprawdzamy, czy dystans pomiędzy osią obrotu, a każdym punktem ruchu drugiego palca jest w normie:
Wyznaczamy odległość bazową: odległości pomiędzy pierwszym punktem palca uznanego za oś, a pierwszym punktem ruchomego palca uznanego za łuk. Obliczamy iloczyn *GP_ROTATION_DIST_PARAM* oraz odległości bazowej, oznaczamy go jako *maxDistortion*. Uznajemy że punkt jest w normie jeśli odległość tego punktu od osi obrotu różni się co najwyżej o *maxDistortion* od odległości bazowej. Każde niespełnienie warunku powoduje odrzucenie ruchu jako rotation.
- 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 osi obrotu). Ponieważ dostajemy zawsze kąt dodatni, to uznajemy, iż kąt powyżej π jest ujemny.
- Moduł wyznaczonego kąta porównujemy z *GP_ROTATION_MIN_ANGLE* - minimalnym kątem obrotu. Jeśli jest mniejszy to odrzucamy.

- 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.

8 Implementacja

8.1 gp_Main.h

Udostępnia podstawowe funkcje realizowane przez bibliotekę

8.1.1 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*

8.1.2 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 palców biorących udział w ruchu
- *gpInt firstTime* przechowuje czas pierwszego dotknięcia ekranu w aktualnym ruchu

8.1.3 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.

8.2 gp_Alloc.h

Odpowiada za zarządzanie pamięcią

8.2.1 Constants

- *gpAlloc_MAX_MEM* 1000000 przechowuje rozmiar bufora pamięci tymczasowej

8.2.2 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)`

Przekopiuje 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

8.3 gp.h

Zawiera użyteczne include'y plików nagłówkowych z folderu base

8.4 gp_bool.h

Definiuje podstawowe aliasy na wartości typu logicznego

8.4.1 Constants

- *false* 0
- *true* 1

8.5 gp_point.h

Zawiera funkcje i struktury dotyczące punktu

8.5.1 Data structures

`gpPoint`

Reprezentacja punktu

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

8.5.2 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

8.6 gp_printf.h

Zawiera funkcje pomocne przy debugowaniu na platformie android

8.7 gp_types.h

Definiuje abstrakcję na typy zależne od platformy

8.7.1 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*

8.7.2 Constants

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

8.8 gp_vector.h

Opisuje abstrakcyjną kolekcję i operacje na niej

8.8.1 Data structures

gpVector

Kolekcja

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

8.8.2 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

`gplnt gpVector_getSize(gpVector* self)`

Zwraca rozmiar kolekcji

- *gpVector* self* wskaźnik na kolekcję

`gpVoid* gpVector_at(gpVector* self, gplnt index)`

Zwraca wskaźnik na element kolekcji

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

`gpVoid gpVector_clean(gpVector* self)`

Czyści zawartość kolekcji

- *gpVector* self* wskaźnik na kolekcję

`gpVoid gpVector_pushBack(gpVector* self, gpVoid* what, gplnt size)`

Dokłada element na koniec kolekcji

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

`gpVoid gpVector_popBack(gpVector* self, gpVoid* where, gplnt 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

8.9 gp_gestures_parameters.h

Opisuje parametry dotyczące tolerancyjności w wykrywaniu ruchów

8.9.1 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")* maksymalne 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

8.10 `gp_gestures_results.h`

Definiuje stałe pomocne przy rozpoznawaniu kierunków ruchu

8.10.1 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

8.11 `gp_MotionEvent.h`

Określa znaczenie kodu akcji w ewencie

8.11.1 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

8.12 `gp_OutputGesture.h`

Zawiera struktury reprezentujące dodatkowe informacje o ruchu

8.12.1 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

8.13 `gp_Math.h`

Zawiera deklarację zaimplementowanych funkcji oraz stałych

8.13.1 Constants

- *GP_FLOAT_BASE* 10000 część liczby przeznaczona na miejsca dziesiętne
- *gpMath_EPSILON* 10 epsilon
- *gpMath_PI* 31416 PI
- *gpMath_2PI* 62832 dwukrotność PI
- *gpMath_PI2* 15708 połowa liczby 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

8.13.2 Functions

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

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

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

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

`gpFloat gpSub(gpFloat a, gpFloat b)`
odejmuje dwie liczby stałopozycyjne

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

`gpFloat gpAdd(gpFloat a, gpFloat b)`
Dodaje dwie liczby stałopozycyjne

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

`gpFloat gpNeg(gpFloat a)`
Zwraca liczbę przeciwną do danej liczby 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 liczby

- *gpFloat a* operand

`gpFloat gpMath.Exp(gpFloat a)`

Zwraca E podniesione do potęgi a

- *gpFloat a* operand

`gpFloat gpMath.Pow(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

9 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

10 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.

11 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