Touch Library

MJaniec BPolnik

24 października 2012

1 Nazwa

Touch Library

2 Ciekawostka

http://www.imore.com/apple-multitouch-google

3 Nazewnictwo

Nie ma namespace. Proponuje zasymulowac poprzez uzycie prefiksu tl. Proponuje tez pisac pseudo-obiektowo.

- Pliki: **tl_file_name.ext**
- Funkcje: **tlFunctionName**
- Zmienne: **tl_var_name**
- Struktury: **tlStructName**
- "Metody": **tlStructName_MethodName(tlStructName* this...)**
- Stale: **tl_CONSTANT_NAME**

4 Makefile

Uzywamy makefile? mnie sie podoba. Wtedy moznaby podawac flage komplilacji DEBUG lub UNCHECKED. DEBUG wlaczalby biblioteke stdio.h oraz funkcje obslugi bledow.

5 Wersja C

Proponuje C99. Fajne petle, slowko const, bool

6 Typy danych

Trzeba ostroznie z pamiecia. W szczegulnosci proponuje nie floating pointow. W ogole wzorem OpenGL proponuje zdefiniowac wlasne typy. Moje propozycje:

- tlBool - bool
- tlUByte - unsigned char
- tlByte - signed char
- tlWord - sigend short
- tlUWord - unsigned short
- tlInt - tlWord
- tlUInt - tlUWord
- tlFloat - tlWord -¿ fixed point
- tlChar - char
- tlVoid - void
- tlString char*

Do debugowania wlasny printf jakis tlPrintf trzaby obslugiwac

- %s - tlString
- %c - tlChar
- %f - tlFloat
- %d %i - tlInt, tlWord
- %ud %ui - tlUInt, tlUWord
- %b - tlBool

7 Obsluga bledow

- tl_errno -numer bledu. 0=brak bledu
- tl_ERRSTR - stringi z opisami bledow
- tl_ERRSTR[tl_errno] - Opis ostatniego bledu
- tl_ERRSTR[0]= 'OK'

Wszystkie funkcje obsluguja bledy w TEN SAM sposob.
Informacja a blendzie tylko w tl_errno
Kazda funkcja na poczatku zeruje tl_errno
jesli po wykonaniu funkcji tl_errno!=0 to blad i wartosc zwrocona z funkcji moze byc invalid

Proponuje makra

- $fun
- $if
- $c
- $e

Definicja funkcji

```
tlType tlFunctionName(tlType1 var1,...){
  $fun;
  code...
  $if (condition) error_code; //if condition rise error error_code;
  code...
  return retval;
}
```

Wywolanie funkcji

```
tlFunctionName(var1,...)$c; //continues
tlFunctionName2(var2,...)$e; //exit
```

Dzialanie makr

```
$fun <=> tl_errno=0
$if condition error_code <=> if(condition){tl_errno=error_code; return;} //zwraca smiecia
//wypisuje informacje o bledzie i kontynuje wykonanie
$c <=> ; {if (tl_errno) print(Filename, linenumber tl_ERRSTR[tl_errno], line of code);}
//wypisuje informacje o bledzie i zamyka program
$e <=> ; {if (jw) print(jw); exit(tl_errno); }
```

Takie podejscie ma 2 zalety.
1. Zaimplementowalem juz cos z grubsza podobnego na potrzeby sysopow, wiec jest praktycznie gotowe.
2. Makra mozna uzaleznic od innego makra. Dzieki temu mozna zamienic wszystkie instrukcje obslugi bledow na blanki przy pomocy parametru kompilacji. W spomniane wyzej DEBUG, UNCHECKED

8 edytor

Nie wiem jescze czego uzywac bedziemy do edytowania, ale wiele edytorow posiada mozliwosc modyfikowania syntax highlightingu. Mozna by ódkeywordowacśtandardowe typy. Dodac typy uzywane w bibliotece. Dodac wyzej wymienione makra. I ponaddto zmienna tl_errno. oraz warto by bylo zrobic jakas stala tl_null=(tlVoid*)(0)

9 input

Na wejscie tablica dwu wymiarowa + jej wymiary jaks funkcja
tlVoid tlInit(tlInt width, tlHeight, tlByte** (getData*)());
Kod klienta by sobie refreshowal
tlVoid tlRefresh();

10 output

Na wyjsciu informacja o wykonywanym gescie , lub jego braku + dodatkowe parametry. Opisane w pliku external.

Kolejka zdazen. Klient moze zarejstrowac pewne zdazenia takie jak scrolldown. wtedy do kolejki beda wrzucane takie gesty. Bedzie to jedynie informacja ze uzytkownik cos zrobil. Rozni sie to tym ze to pojawia sie tylko po wykonaniu gestu i pociaga za soba konkretna akcje.