K-Da Library

Andrey Pérez Salazar
Andrés Sánchez López
David Pérez Bolaños

University of Costa Rica

June, 2nd, 2014

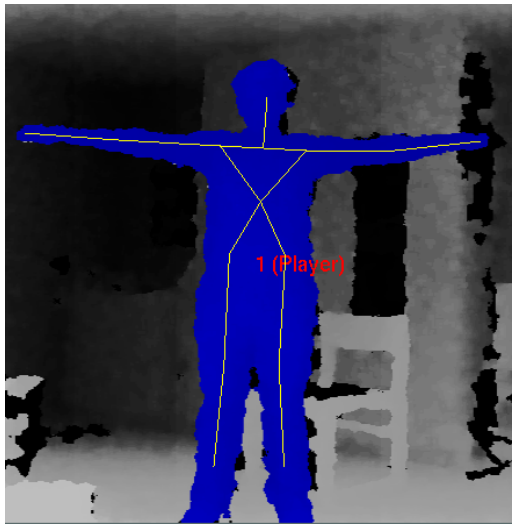
Resume

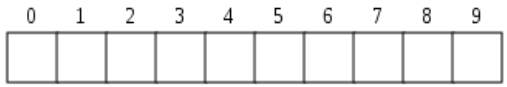
K-Da Library

Programming K-Da Library
- Class archivos
- Class conversion
- Class compara
- main

Functions to implement

K-Da Library





Class archivos

.hh

```
#ifndef ARCHIVOS_HH
#define ARCHIVOS_HH

class archivos {
public:
    archivos();
    archivos(const archivos& orig);
    virtual ~archivos();
    string getDirArchivo();
    int getCantLineas();
    void guardarEnArreglo();
    void setCantLineas();
    string * getDatosArreglo();
    void setDirArchivo(string pDato);

private:
    int _CantLineas;
    string _DirArchivo;
    string *_DatosArreglo;
};
```

Class archivos

.cpp

```
int archivos::getCantLineas() {
    return _CantLineas;
}

void archivos::setCantLineas() {
    string line;
    ifstream myfile(_DirArchivo.c_str());
    _CantLineas = 0;
    if (myfile.is_open()) {
        while (getline(myfile, line)) {
            _CantLineas = _CantLineas + 1;
        }
        myfile.close();
    }
}

string archivos::getDirArchivo() {
    return _DirArchivo;
}
```

Class archivos

.cpp

```
void archivos::setDirArchivo(string pDato) {
    this->_DirArchivo = pDato;
    setCantLineas();
    guardarEnArreglo();
}

void archivos::guardarEnArreglo() {
    _DatosArreglo = new string[_CantLineas];
    string line;
    ifstream myfile(_DirArchivo.c_str());
    int _Contador = 0;
    if (myfile.is_open()) {
        while (getline(myfile, line)) {

            _DatosArreglo [_Contador] = line;
            _Contador = _Contador + 1;
        }
        myfile.close();
    }
}

string * archivos::getDatosArreglo() {
    return _DatosArreglo;
}
```

Class conversion

.hh

```
class conversion {
public:
    conversion(string  ptxt1, string ptxt2);
    conversion(const conversion& orig);
    virtual ~conversion();
    double convertir(string* pjoint1, string* pjoint2, int  n);
    string * split(string pDato);
    void llenarArregloAngulos();
    double * getArregloAngulos();

private:
    archivos _Joint1_Txt;
    archivos _Joint2_Txt;
    double  * _Arreglo_angulos;
};
```

Class conversion

.cpp

```
double conversion::convertir(string* linea1, string* linea2, int n) { //Recibe arreglos ["1235,123,213"]
    string * arrXYZ = split(linea1[n]);
    string * arrXYZ_2 = split(linea2[n]);
    Vector3d a(atof(arrXYZ[0].c_str()), atof(arrXYZ[1].c_str()), atof(arrXYZ[2].c_str()));
    Vector3d b(atof(arrXYZ_2[0].c_str()), atof(arrXYZ_2[1].c_str()), atof(arrXYZ_2[2].c_str()));
    double dotProduct = a.dot(b);
    double aNormal = a.norm();
    double bNormal = b.norm();
    double cosangle = (dotProduct) / ((a.norm())*(b.norm()));
    double angleRad = acos(cosangle);
    double angleGrad = (acos(cosangle))*((180.0) / (PI));
    return angleGrad;
}
```

Class conversion

.cpp

```
string * conversion::split(string pDato) {
    string delimiters = " ,";
    size_t current;
    size_t next = -1;
    string * arregloXYZ = new string [3];
    int contador = 0;
    do {
        current = next + 1;
        next = pDato.find_first_of(delimiters, current);
        arregloXYZ[contador] = pDato.substr(current, next - current);
        contador++;
    } while (next != string::npos);
    return arregloXYZ;
}

void conversion::llenarArregloAngulos() {
    for (int cont = 0; cont < _Joint1_Txt.getCantLineas(); cont++) {
        double angulo = convertir(_Joint1_Txt.getDatosArreglo(), _Joint2_Txt.getDatosArreglo(), cont);
        _Arreglo_angulos[cont] = angulo;
    }
}

double *conversion::getArregloAngulos() {
    return _Arreglo_angulos;
}
```

Class compara

.hh

```
class compara {
public:
    compara();
    compara(const compara& orig);
    virtual ~compara();
    double * sacapromedios(double * arreglo);
    int * arreglo_promedio(double *arreglo_prom1, double *arreglo_prom2);
    int * getArregloComparativo();
private:
     double * _ArregloMov1;
     double  * _ArregloMov2;
     int * _ArregloComparativo;


};
```

Class compara

.cpp

```
compara::compara() {
    conversion persona1("jointp1", "jointp2");
    conversion persona2("joint1", "joint2");
    this->_ArregloMov1 = persona1.getArregloAngulos();
    this->_ArregloMov2 = persona2.getArregloAngulos();
    double * arregloP1 = sacapromedios(_ArregloMov1);
    double * arregloP2 = sacapromedios(_ArregloMov2);
    /*for (int i = 0; i < 10; i++) {
        cout << arregloP1[i] << endl;
    }
    for (int i = 0; i < 10; i++) {
        cout << arregloP2[i] << endl;
    }
    for (int i = 0; i < 10; i++) {
        cout << _ArregloMov2[i] << endl;
    }*/
     _ArregloComparativo = arreglo_promedio(arregloP1, arregloP2);

}
```

Class compara

.cpp

```
double * compara::sacapromedios(double* arreglo) {
    double * arreglo_prom = new double [10]; //revisar
    for (int k = 0; k < 10; k++) {
        int sumatoria = 0;
        for (int i = int(k * 10 * 0.1); i < int(10 * 0.1 * (1 + k)); i++) {
            sumatoria = sumatoria + arreglo[i];
        }
        arreglo_prom[k] = double(sumatoria) / double(int(10 * 0.1));
    }
    return arreglo_prom;
}
```

Class compara

.cpp

```
int * compara::arreglo_promedio(double *arreglo_prom1, double *arreglo_prom2) {
    int * selecciona = new int [10]; //revisar
    for (int i = 0; i < 10; i++) {
        if (arreglo_prom1[i] > arreglo_prom2[i] + 5.0) {
            selecciona[i] = 0;
        }
        else if (arreglo_prom1[i] < arreglo_prom2[i] - 5.0) {
            selecciona[i] = 0;
        }
        else{
            selecciona[i] = 1;
        }
    }
    for (int j = 0; j < 10; j++) {
    }
    return selecciona;
}

int * compara::getArregloComparativo() {
    return _ArregloComparativo;
}
```

main

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include "conversion.hh"
#include "compara.hh"
using namespace std;

int main() {
    compara movimiento;
    int * comp = movimiento.getArregloComparativo();

    for (int i = 0; i < 10; i++) {
        cout << comp[i] << endl;
    }
    return 0;
}
```

Comparison of the movements

Que tan bien se movió:

- Que tan parecido
- Que tan rápido

Comparison of the movements

```
void comparar_angulos(int promedio[]){
    int contador =0;
    for(int l=0;l<10;l++){
        contador+=promedio[l];
    }

    //cout<<"Contador es : "<<contador<<" \n";
    if (contador>=9){
        cout<<"El movimiento fue excelente"<<"\n";
        }
    else if(contador>=7){
        cout<<"El movimiento fue bueno pero podrias mejorar"<<"\n";
    }
    else if(contador>=5){
        cout<<"El movimiento fue regular"<<"\n";
    }
    else if(contador>=3){
        cout<<"El movimiento fue deficiente"<<"\n";
    }
    else {
        cout<<"El movimiento fue muy deficiente"<<"\n";
    }
}
```

Comparison of the movements

```
int comparar_velocidad(int lista_falsa[], int n, int lista_falsa1[], int k){

    float rizado;
    rizado = n*0.1;
    if (k<n-rizado){
        cout<<"Hiciste el movimiento demasiado rapido\n";
        return 1;
        }
    else if (k>n+rizado){
        cout<<"Hiciste el movimiento demasiado lento\n";
        return 1;
        }
    else {
        cout<<"Hiciste el movimiento a la velocidad adecuada, !Muy bien!\n";
        return 0;
        }

    }
```