152114002: NESNE TABANLI PROGRAMLAMA I
2019-2020 GÜZ DÖNEMİ
DÖNEM PROJESİ
Son Teslim Tarihi: 13 Aralık 2019, Cuma, 17:00

Açıklama ve Kurallar:

1. Grup çalışması içerisinde,
   - **i.** Bütün grup üyeleri, tasarım sürecinde bulunmalıdır.
   - **ii.** Görevler (sınıf kodlarının ve uygulamanın yazılması) tüm grup üyelerine dağıtılmalı ve raporda bu dağılım açıkça belirtilmelidir.
   - **iii.** Her öğrenci, kod sürüm takip sistemi olarak kullanılan bitbucket üzerinde hesap açarak (laboratuvarda nasıl kullanılacağı anlatılacaktır), her grupta bir takım lideri seçilecek, lider bir proje oluşturacak, takım üyeleri kendilerine atanan kaynak dosyalar üzerinde aynı projede çalışacaktır. Değerlendirmede, her bir takım üyesinin buradaki hareketi dikkate alınacaktır.
   - **iv.** Her öğrenci, kendisine atanan sınıf kodlarını ve test programlarını yazmalıdır. Her sınıf ayrı header (.h) ve ayrı source (.cpp) dosyasına sahip olmalıdır. Her dosyada kodu yazan kişinin ismi ve tarih bulunmalıdır.
2. Kodların yanında ilgili kod parçası ile ilgili açıklamaların yazılması gerekmektedir. (Bakınız EK A.1).
3. Proje için, tasarım ve gerçekleme aşamasının aktarıldığı bir rapor hazırlanması gerekmektedir. (Bakınız EK A.2)
4. Nesne tabanlı yapıların kullanıldığı (abstraction, data hiding, inheritance, operator overloading, polymorphism, templates, exception handling, etc) iyi bir tasarım ve kodlama yapmalısınız.
5. Proje için, yazılan kodları ve proje raporunu sıkıştırarak. zip ya da .rar olarak, ve dosyayı "Grup_Uyelerinden_birinin_ismi.zip/rar" şeklinde isimlendirip, DYS yüklemelisiniz.
6. Notlar, ekte verilen tabloya göre verilecektir. (Bakınız EK A.3)
7. Uygulama programı konsol tabanlı olmalıdır.
8. Sadece standart C++ kütüphaneleri kullanılmalıdır.
9. **Tasarım ve/ve ya kodlarınızın kısmen ya da tamamen başka gruplardan ve referans gösterilmemiş kaynaklardan alındığı belirlenirse, sıfır not alırsınız.**

10. **Proje gruplarının sunum yapması istenebilir.**

PROJE BAŞLIĞI: Nokta Bulutu İşleme

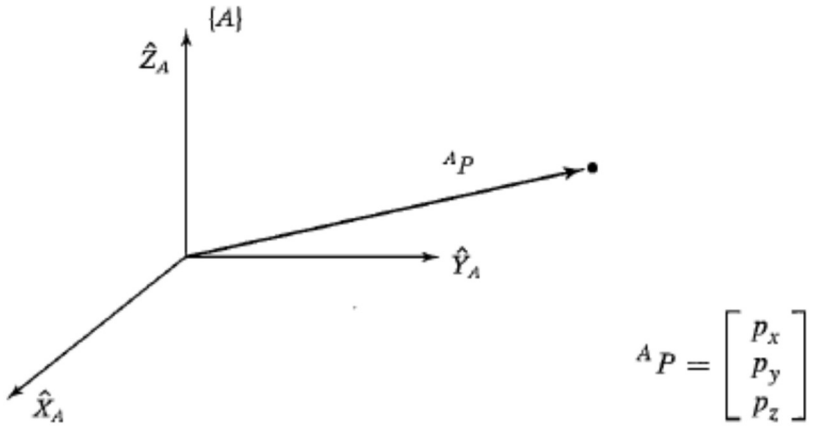
Proje Açıklaması:

Projenin kapsamı, nokta bulutları (Point Cloud) ve nokta bulutları üzerinde işlemleri kapsamaktadır. Şekil 1'de resmi verilen derinlik kamerası, görüş alanı içerisinde renkli piksel değerleri yanına derinlik bilgisi de sunmaktadır. Elde edilen 3B derinlik bilgisi kameraya göre tanımlı 3B noktalarla ifade edilmektedir. Bu noktaların oluşturduğu noktalar kümesine nokta bulutu adı verilir. Şekil 1'de bir tavşan yüzeyinden elde edilen nokta bulutunun gösterimi verilmektedir.

Şekil 1. Derinlik kamerası ve nokta bulutu

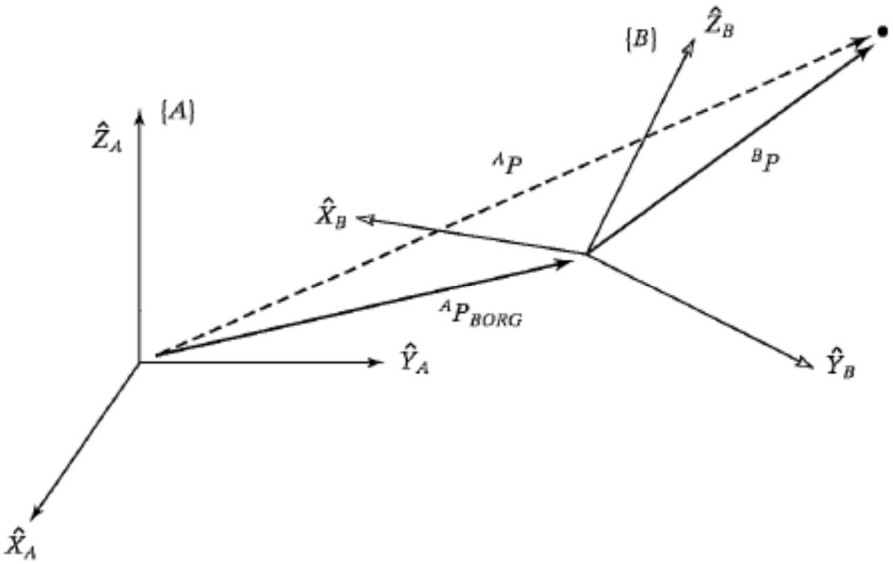
Bir kameradan alınan nokta bulutundaki her bir nokta, o kamera üzerinde tanımlı koordinat sistemine göre tanımlıdır. Örneğin, Şekil 2'de mavi renkli elips içinde gösterilen nokta bulutu O_{C1} koordinat sistemine göre tanımlıdır. Yeşil renkli elips içinde gösterilen nokta bulutu O_{C2} koordinat sistemine göre tanımlıdır. Bu iki nokta bulutunu birleştirip tek bir nokta bulutu olarak elde etmek için, her iki buluttaki noktaları ortak bir koordinat sistemine (O_B) göre tanımlı olacak şekilde dönüştürmek gerekir.

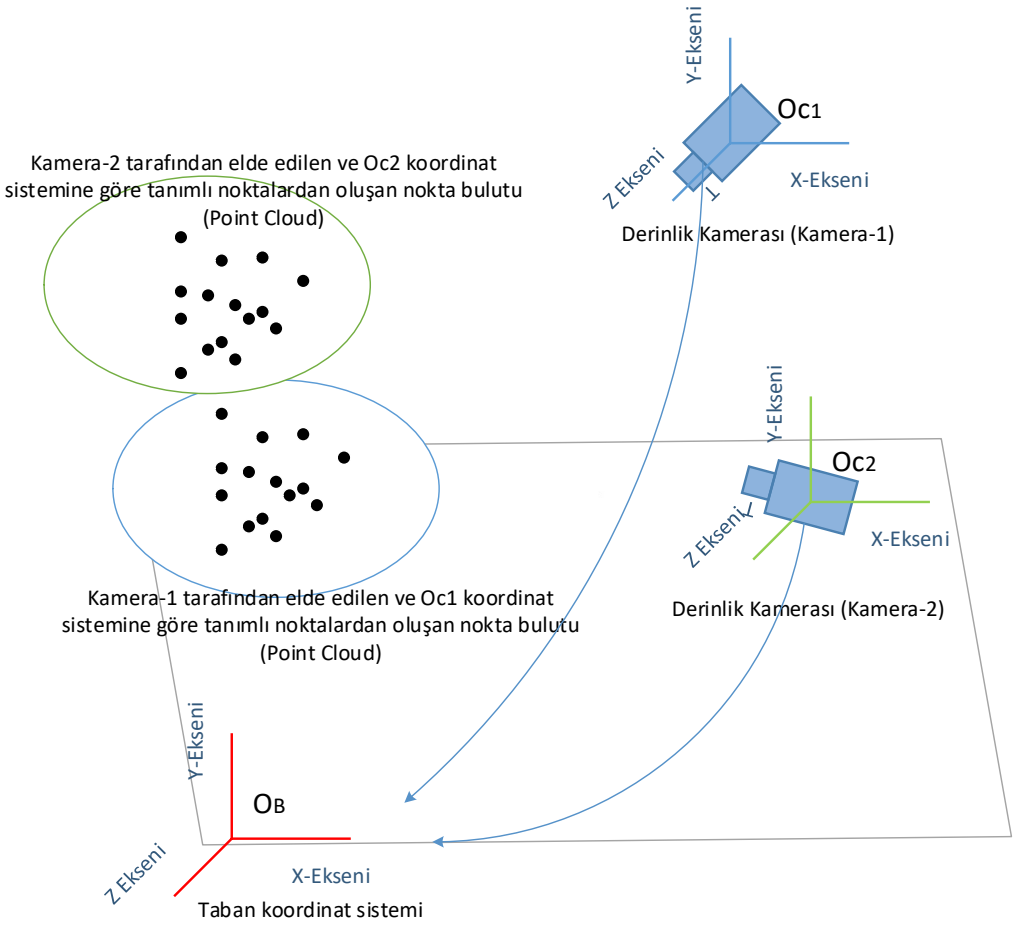
P noktasının, {A} koordinat eksenine göre gösterimi aşağıda verilmektedir.



$$ {}^AP = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} $$

Bir P noktasının, {A} ve {B} koordinat eksenlerine göre, gösterimi sırasıyla ${}^AP$ ve ${}^BP$ ile gösterilmektedir.



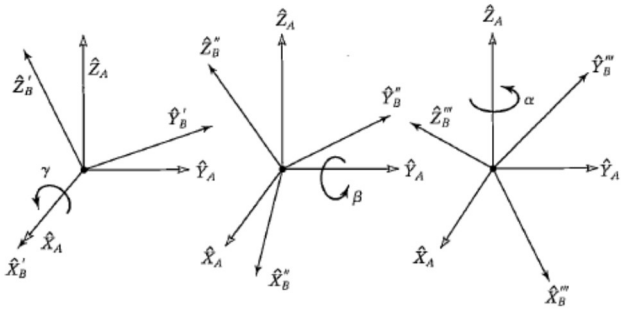


Şekil 2. İki farklı kameradan sağlanan nokta bulutları ve referans koordinat sistemlerinin sembolik gösterimi

${}^{B}P$ verildiğinde ${}^{A}P$ vektörünü bulmak için {B} 'den {A}'ya dönüşüm matrisi gerekir. Dönüşüm matrisi ${}^{A}T_{B}$, aşağıda gösterilmektedir. ${}^{A}_{B}R$, {B} 'den {A}'ya 3x3 rotasyon matrisidir.

$$\begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}.$$

Dönüşüm matrisinin ${}^{A}T_{B}$ tersi, {A} 'dan {B}'ye dönüşüm matrisini verir. $\left({}^{A}T_{B}\right)^{-1} = {}^{B}T_{A}$. Koordinat eksenleri arasındaki rotasyon matrisi 3 açı değeri ile belirtilebilir. X-Y-Z dönüşümüne göre



γ, β, α açıları ile belirtilen rotasyon için rotaston matrisi

$${}^{A}_{B}R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}.$$

olarak hesaplanabilir.

Proje İstekleri:

Proje kapsamında, iki farklı kameradan elde edilen nokta bulutlarının taban koordinat sistemine dönüştürülmesi ve tek bir nokta bulutu olarak elde edilmesi istenmektedir. Ayrıca nokta bulutu üzerinde bazı filtreleme işlemlerinin yapılabilmesi isteniyor. Aşağıda geliştirilecek sınıflar UML olarak verilmektedir. Üye fonksiyonlar için zorunlu olanlar verilmiştir. Bunun dışında gereken ya da sizin eklemek istediğiniz üye fonksiyon ve üye verileri sınıflara dahil edebilirsiniz.

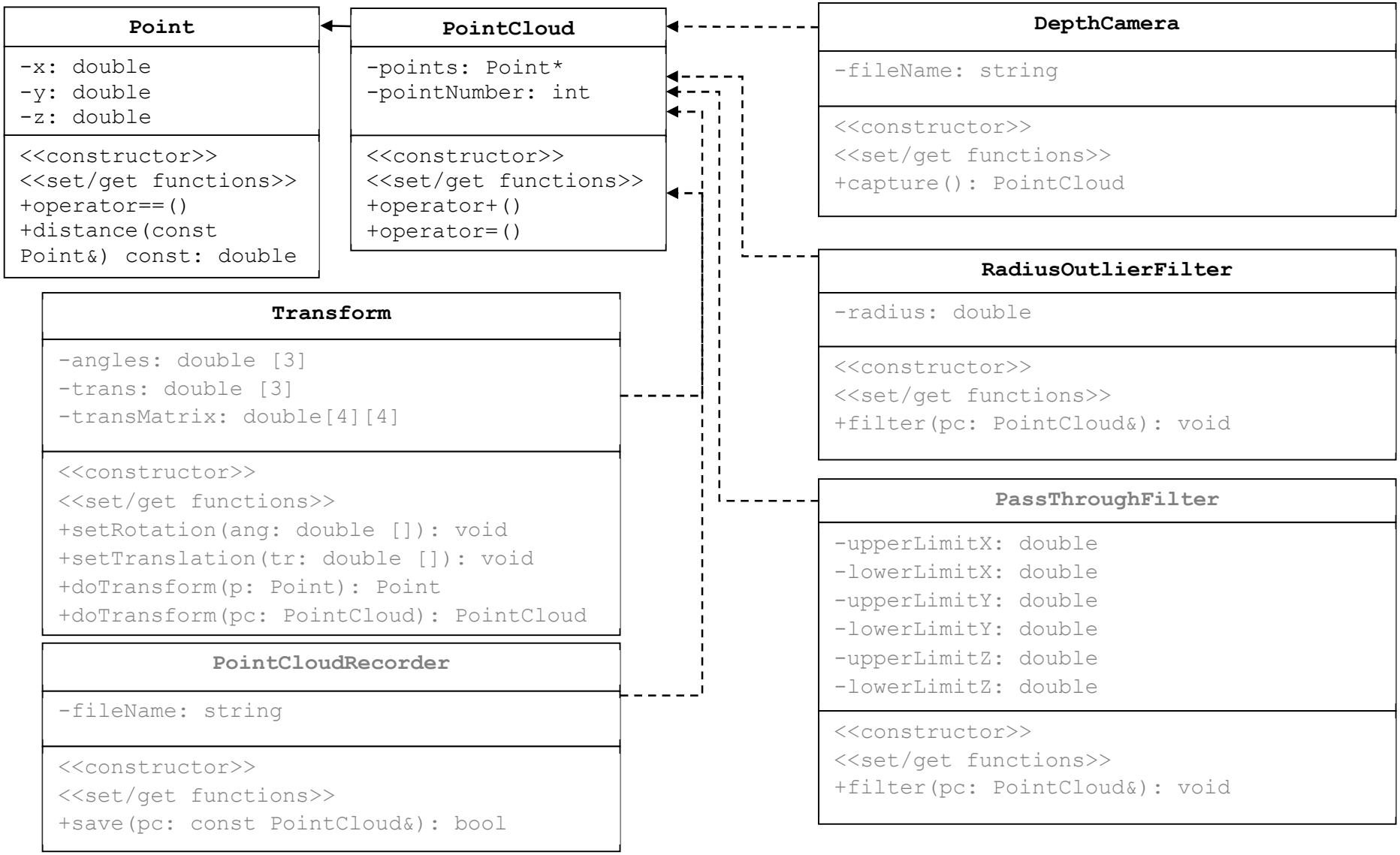
Şekil 3'de geliştirilmesi beklenen taslak UML sınıf diyagramı verilmektedir. Bu tasarımda, genel olarak sadık kalınmalı ancak üzerinde gerekli değişiklikler yapılabilir. Tasarımdaki sınıflar ve üye fonksiyonların açıklamaları kısaca şu şekildedir:

Point Sınıfı: Bu sınıf nokta bulutundaki 3B noktaların koordinatlarını tutar. Eşit operatörü (= =), iki noktanın eşit olup olmadığını denetler.

PointCloud Sınıfı: Sahip olduğu noktaları, dinamik olarak yaratılan bir Point dizisinde tutar. Dizinin boyutu, nesne yaratılırken constructor fonksiyonunda bir parametre olarak alınır. + operatörü, her iki nokta bulutunun sahip olduğu noktalara sahip tek bir nokta bulutunu döndürür. = operatörü, bir nokta bulutunun başka bir nokta bulutuna kopyalanmasını sağlar.

Transform Sınıfı: Açıklamlar kısmında anlatıldığı gibi, iki koordinat ekseni orijinleri arasındaki uzaklık (trans) ve rotasyon açılarını (angles) alır. Dönüşüm matrisini oluşturur (transMatrix). Daha sonrasında doTrans fonksiyonu ile alınan nokta ya da nokta bulutunu bu dönüşüme tabi tutarak dönüştürülmüş nokta ya da nokta bulutunu döndürür.

DepthCamera Sınıfı: İsmi (fileName) verilen dosyadan, capture fonksiyonu çağrıldığında noktaları okur ve yaratılan nokta bulutu nesnesine bu noktaları atar. Nokta bulutunu döndürür. Bu işlem kameranın bir benzetimidir. Noktalar bir kamera yerine dosyadan alınır.



Şekil 3. Geliştirilecek yazılımın taslak UML Sınıf Diyagramı

RadiusOutlierFilter Sınıfı: Bu nokta bulutunda filtreleme yapar. Filter fonksiyonu ile nokta bulutunu alır ve filtrelenmiş halini döndürür. Bu filtreleme işleminin algoritması şu şekildedir. Nokta bulutundaki her bir nokta için tek tek işlem yapılır. Noktaya, radius değerinden daha yakın başka bir nokta yok ise, bu nokta nokta bulutundan çıkarılır.

PassThroughFilter Sınıfı: Bu nokta bulutunda filtreleme yapar. Filter fonksiyonu ile nokta bulutunu alır ve filtrelenmiş halini döndürür. Bu filtreleme işleminde, noktanın x, y ve z değerlerinden en az birisi limitlerin dışında ise, bu nokta nokta bulutundan çıkarılır.

PointCloudRecorder Sınıfı: Bu nokta bulutlarının dosyaya kaydedilmesi için kullanılmaktadır. save fonksiyonu çağrıldığında, fileName ile ismi verilen dosya açılır, parametre olarak verilen nokta bulutundaki noktalar bu dosyaya kaydedilir.

Hem DepthCamera hemde PointCloudRecorder tarafından kullanılan metin tabanlı dosyada noktalar şu şekilde bulunmalıdır. Her satırda bir nokta olmak üzere, noktanın x, y, z bileşen değerleri aralarında boşlukla bulunacaktır.

```
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
.
.
.
Xn Yn Zn
```

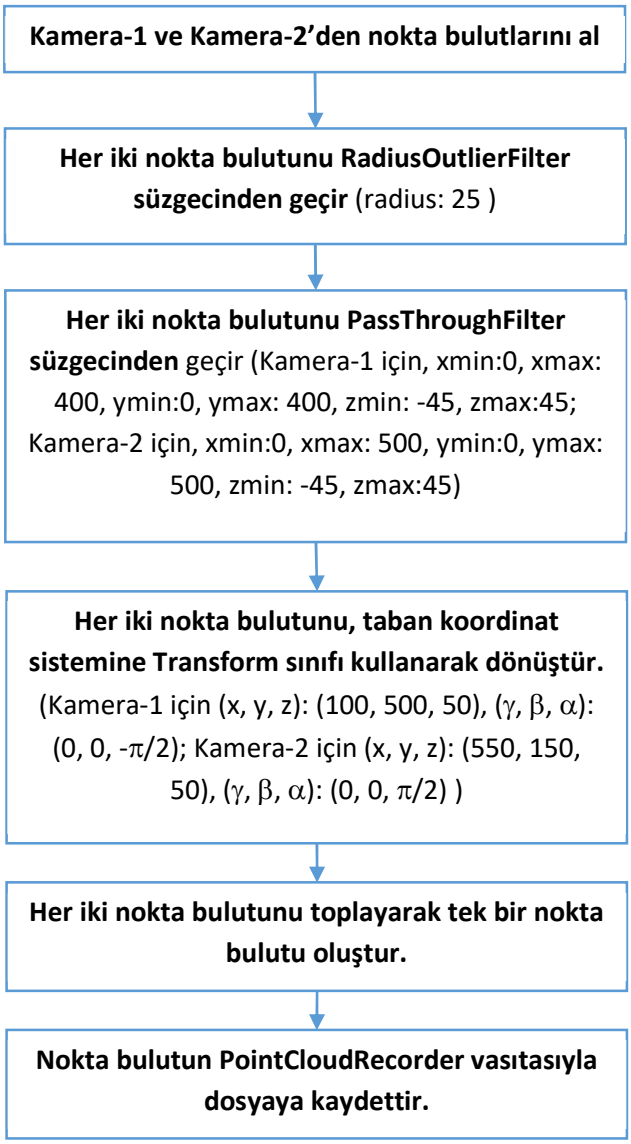
Nokta bulutunu 3B görselleştirmek için, CloudCompare (https://www.danielgm.net/cc/) programını bilgisayarınıza kurup, yukarıdaki formattaki dosyalarınızı bu programda açabilirsiniz. Nokta bulutunu 3B olarak görebilirsiniz.

Kısım 1.

Yukarıda verilen tasarıma uygun olarak bir Nokta Bulutu Kütüphanesi oluşturun. Her sınıf için doğruluğunu test etmek için bir test programı yazınız. Testler sırasında, Kısım 2'de verilen örnek değerlerden faydalanabilirsiniz.

Kısım 2.

Bir test uygulama programı yazınız. Bu test programında iki adet derinlik kamerası vardır. Derinlik kameralarından birincisinin (Kamera-1) koordinat ekseninin orijini, taban koordinat sistemine göre (x, y, z): (100, 500, 50) -translation- ve (γ, β, α): (0, 0, -π/2) -rotation- olarak tanımlıdır. Derinlik kameralarından ikincisinin (Kamera-2) koordinat ekseninin orijini, taban koordinat sistemine göre (x, y, z): (550, 150, 50) -translation- ve (γ, β, α): (0, 0, π/2) -rotation- olarak tanımlıdır. Kamera-1'den alınan nokta bulutu "camera1.txt" ve Kamera-2'den alınan nokta bulutu "camera2.txt" olacaktır.



DOSYA EKLERİ:

EK 1. "camera1.txt" dosyası

EK 2. "camera2.txt" dosyası.

Appendix:

A.1 Doxygen HowTo

Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#, and to some extent D.

It can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. You can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can also use doxygen for creating normal documentation (as I did for this manual).

Doxygen is developed under Linux and Mac OS X, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

For more detail, http://www.stack.nl/~dimitri/doxygen/manual.html

Sample :

The Header File (TestA.h) is:

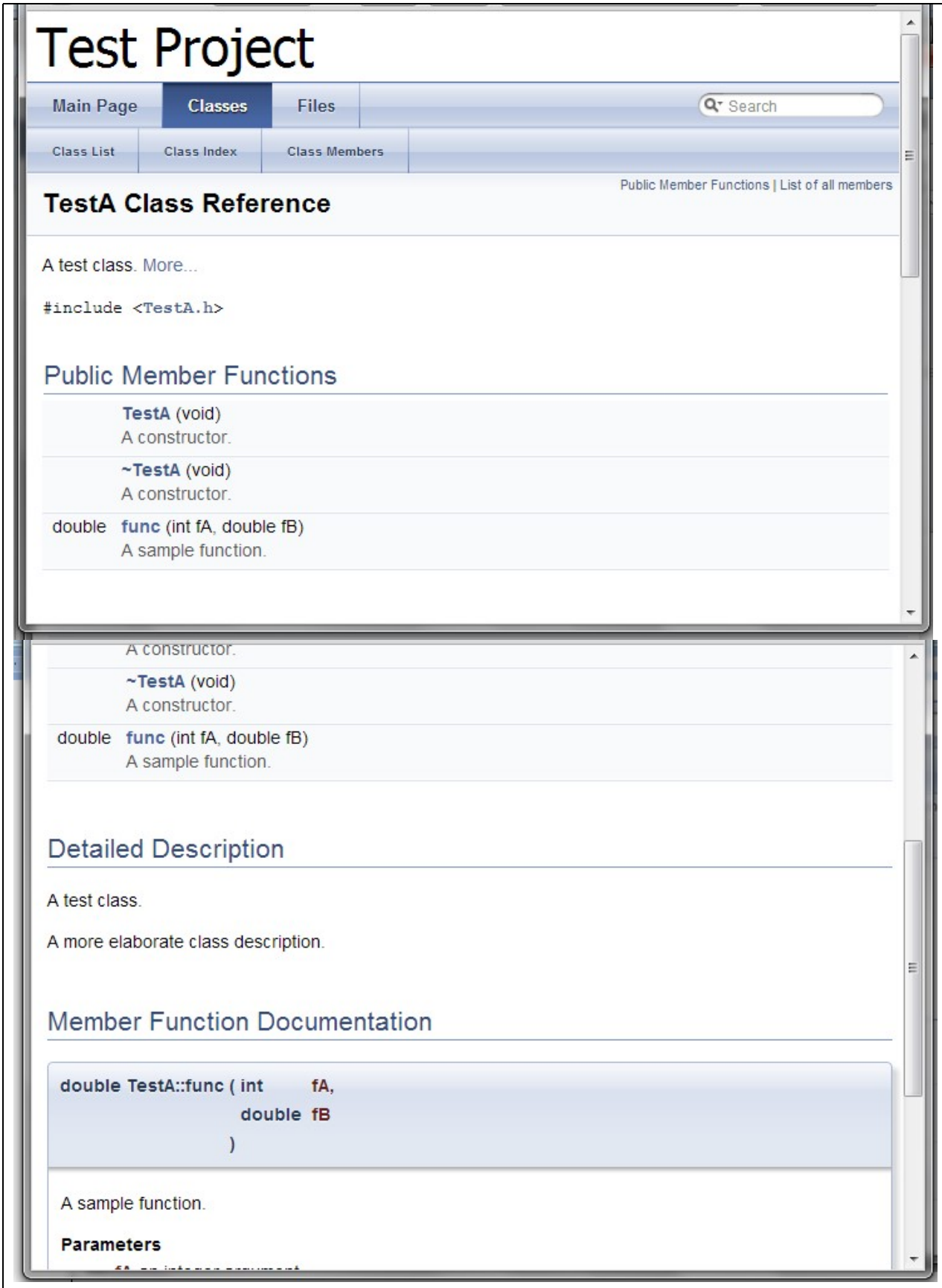
```
/**
 * @file   TestA.h
 * @Author Me (me@example.com)
 * @date   September, 2008
 * @brief  Brief description of file.
 *
 * Detailed description of file.
 */

//! An enum.
/*! More detailed enum description. */
enum TestENUM{
     ENUM1,  /*!< Definition of ENUM1 */
     ENUM2,    /*!< Definition of ENUM2 */
     ENUM3     /*!< Definition of ENUM3 */
};

//!  A test class.
/*!
  A more elaborate class description.
*/
class TestA{
public:
     //! A constructor.
     TestA(void);
     //! A constructor.
```

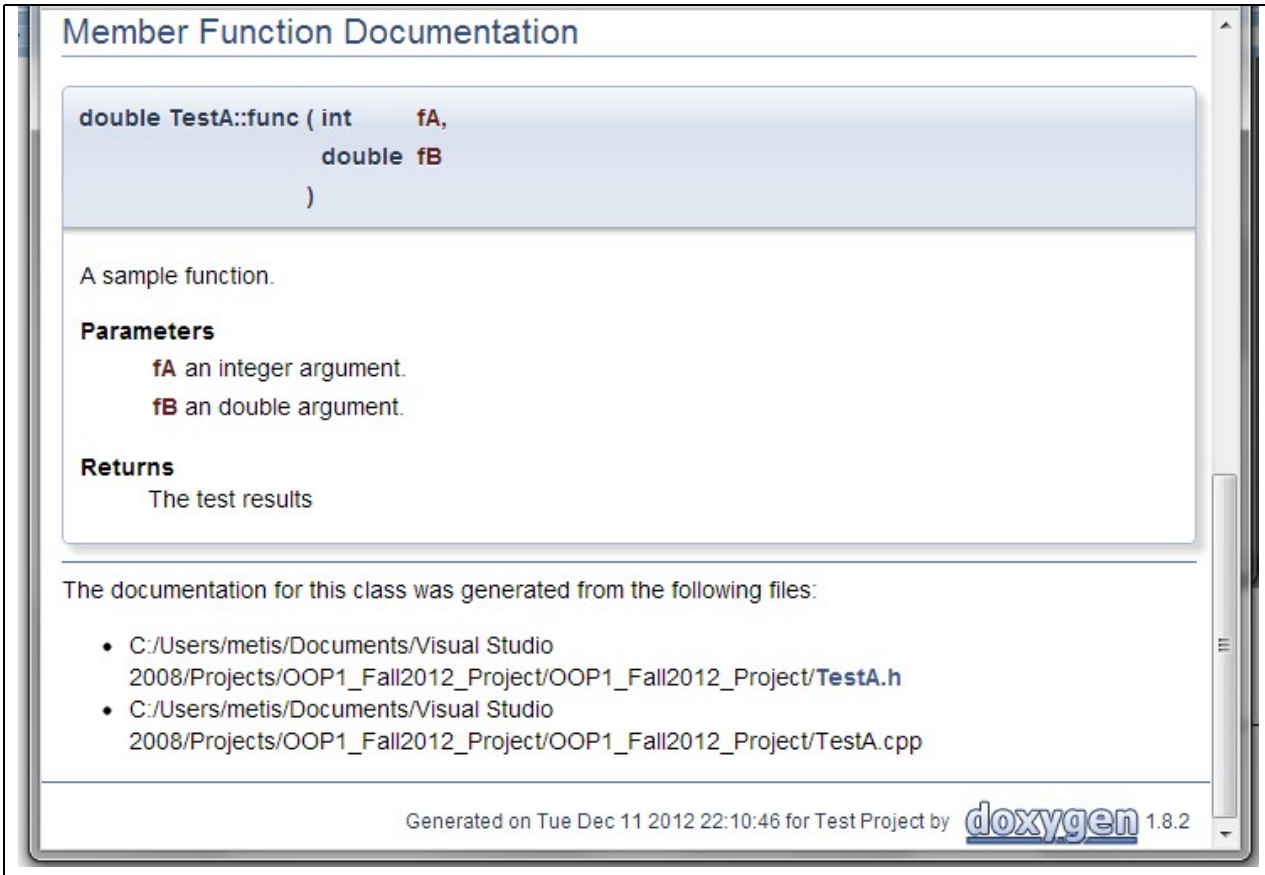
```
     ~TestA(void);
     //! A sample function.
     double func(int fA, double fB);
};
```

The Source File (TestA.cpp) is:

```
#include "TestA.h"

TestA::TestA(void)
{}

TestA::~TestA(void)
{}

/*!
     \param fA an integer argument.
    \param fB an double argument.
    \return The test results
*/
double TestA::func(int fA, double fB)
{
     return fA*fB;

}
```

Snapshots from html type documentation pages after generating by Doxygen


Test Project
Main Page
Classes
Files
Search
Class List
Class Index
Class Members
Public Member Functions | List of all members
TestA Class Reference
A test class. More...
#include <TestA.h>
Public Member Functions
TestA (void)
A constructor.
~TestA (void)
A constructor.
double func (int fA, double fB)
A sample function.
A constructor.
~TestA (void)
A constructor.
double func (int fA, double fB)
A sample function.
Detailed Description
A test class.
A more elaborate class description.
Member Function Documentation
double TestA::func (int fA,
double fB
)
A sample function.
Parameters



Member Function Documentation

double TestA::func (int fA,
double fB
)

A sample function.

Parameters

fA an integer argument.

fB an double argument.

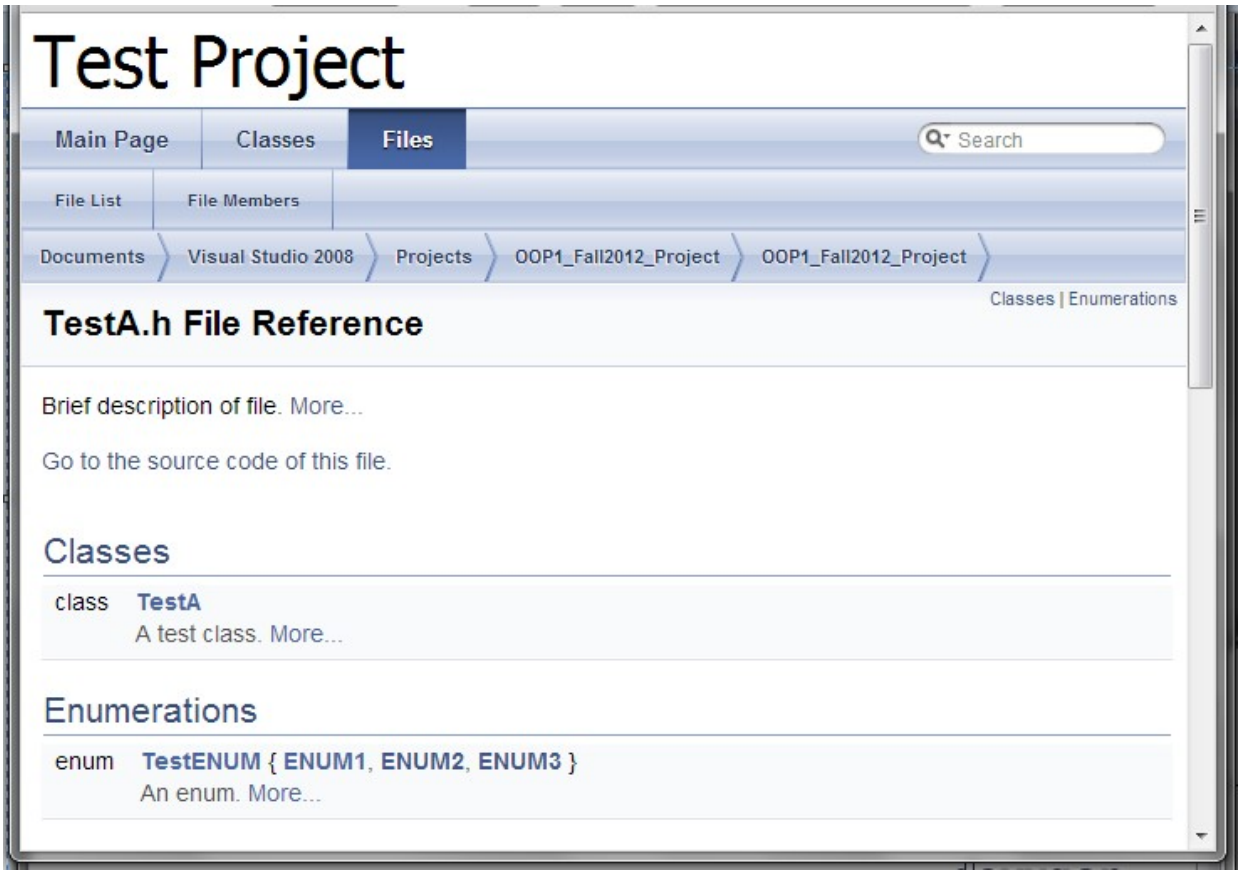
Returns

The test results

The documentation for this class was generated from the following files:

- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/**TestA.h**
- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/TestA.cpp

Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2



Test Project

Main Page | Classes | Files | Search

File List | File Members

Documents > Visual Studio 2008 > Projects > OOP1_Fall2012_Project > OOP1_Fall2012_Project

Classes | Enumerations

TestA.h File Reference

Brief description of file. More...

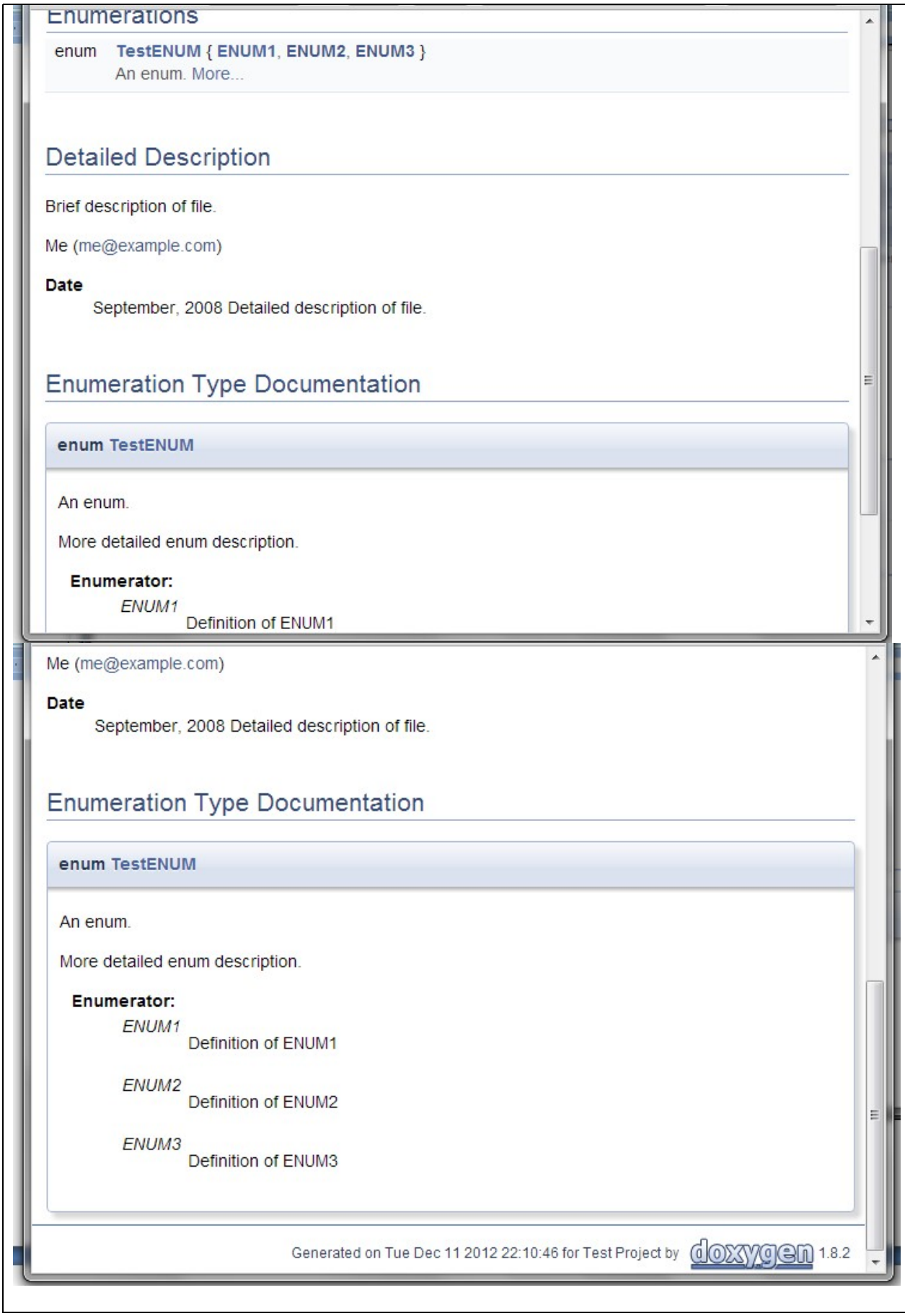
Go to the source code of this file.

Classes

class **TestA**
A test class. More...

Enumerations

enum **TestENUM** { **ENUM1**, **ENUM2**, **ENUM3** }
An enum. More...


Enumerations
enum TestENUM { ENUM1, ENUM2, ENUM3 }
An enum. More...
Detailed Description
Brief description of file.
Me (me@example.com)
Date
September, 2008 Detailed description of file.
Enumeration Type Documentation
enum TestENUM
An enum.
More detailed enum description.
Enumerator:
ENUM1
Definition of ENUM1
Me (me@example.com)
Date
September, 2008 Detailed description of file.
Enumeration Type Documentation
enum TestENUM
An enum.
More detailed enum description.
Enumerator:
ENUM1
Definition of ENUM1
ENUM2
Definition of ENUM2
ENUM3
Definition of ENUM3
Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2

A.2 Report Format

Kapak Sayfası

ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE TABANLI PROGRAMLAMA I
PROJE RAPORU -1

Proje Başlığı

Proje Grup Üyesi Numarası ve Adı
Proje Grup Üyesi Numarası ve Adı
Proje Grup Üyesi Numarası ve Adı
...

Aralık 2019

Rapor İçeriği

1. **Giriş**
   Genel proje bilgisi

2. **Tasarım**
   Alt başlıklar eklenebilir. Tasarım, UML diyagramları, görev atamaları, örnek program girdi ve çıkatılarının kısaca anlatılması.

Görev Atamaları (İçerilmeli)

Grup Üyesi	**Görevler**
Grup üyesinin adı	*Sınıflar, dokümantasyon, etc.*
Grup üyesinin adı	*. . .*
. . .	**. . .**

3. **Sonuçlar**
   Proje sonuçlarının değerlendirilmesi, takım çalışması ile ilgili yorumlar, artı ve eksiler, sonraki çalışmalar için öneriler, vb.

A.3 Check list for the materials which should be submitted and grading

CHECK LIST

Materials	**Included**
Header and source files of each classes and application (.h, .cpp)	Yes / No
Each class has separate header and source files?	Yes / No
Codes are well formatted?	Yes / No
Codes are commented by using Doxygen tags?	Yes / No
There are test programs for each class?	Yes / No
Internal documentation generated by Doxygen (.htm)	Yes / No
Report	Yes / No

GRADING (TENTATIVE)

Items	**Grade (%)**
Individual Studies	
Internal Documentation	5
Code quality (well formatted)	5
Completeness	5
Overall (including all classes and application)	
Internal documentation (using doxygen)	10
Overall code quality	10
Group work	15
Functionality (implemented and correctly running functionalities).	30
Report	20