

152114002: NESNE TABANLI PROGRAMLAMA I
2022-2023 GÜZ DÖNEMİ
DÖNEM PROJESİ (II.AŞAMA)
Son Teslim Tarihi: 06 Ocak 2023, 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 gerçekleştirme 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/veya kodlarınızın kısmen ya da tamamen başka gruplardan ve referans gösterilmemiş kaynaklardan alındığı belirlenirse, sıfır not alırsınız.**
10. **Proje gruplarının sunum yapması istenebilir.**

PROJE BAŞLIĞI: 3B Izgara Tabanlı Haritalama (3D Grid-Based Mapping)

Proje Açıklaması:

Kısım 1’de verilen projedeki tasarım üzerinde değişiklikler yapılması istenmektedir. Bu değişiklikler, bazı sınıfların çok yüzlü(polymorphic) yapıya dönüştürülmesi, vector ve list kullanımları, lineer cebir hesaplamalarında kullanılan açık kaynak kodlu bir kütüphanenin bu projede kullanımı, iki yeni sınıfın eklenmesini kapsamaktadır.

Mevcut sınıflarda yapılacak değişiklikler:

1. **PointCloud** sınıfında, nokta bulutunu oluşturan noktaları barındıran array yerine **list** kullanılacaktır.

2. **Transform** sınıfında lineer cebir işlemleri kapsamında, vektörler, matrisler, matris çarpımları kullanılmaktadır. İlk kısımda, bu kapsamda vektör ve matrisler array ile oluşturuldu ve matris çarpma işlemleri gerçekleştirildi. Bu kısımda ise, **eigen** kütüphanesi kullanılacaktır. Bu sınıfta kodlamasında ihtiyaç duyulacak 3x1 boyutunda vektör **Eigen::Vector3d** yapısı ile, 4x1 boyutunda vektör **Eigen::Vector4d** yapısında, 3x3 matris **Eigen::Matrix3d** ve 4x4 boyutunda matris **Eigen::Matrix4d** yapısında tanımlanacaktır. İşlemlerin kullanımına yönelik örnekler eigen kütüphanesinin web sayfasından ulaşılabilir. http://eigen.tuxfamily.org/index.php?title=Main_Page

Kütüphanenin kullanımı için faydalı olacak olan dokümanı ve bazı örnek kodlamalar sitesinden ulaşılabilir: http://eigen.tuxfamily.org/dox/group__TutorialMatrixArithmetic.html

(Amaç: Sizin geliştirmedığınız açık kaynak kodlu kütüphaneleri, projenizde kullanabilme kabiliyetinin geliştirilmesi)

3. Haritalar, **Map** abstract sınıfı altında toplanacaktır. **insertPointCloud**, **insertPoint**, **loadMap**, **saveMap** üye fonksiyonları **pure virtual** fonksiyon olacaktır.

4. **DepthCamera** sınıfı, **PointCloudGenerator** abstract sınıfı altında toplanacaktır. **capture** ve **captureFor** üye fonksiyonları **pure virtual** fonksiyon olacaktır. **capture** fonksiyonu Kısım 1 ‘de anlatıldığı gibi çalışacaktır. **captureFor** fonksiyonu ise, **capture** fonksiyonu ile öncelikle aynı işlemleri yapacak; ancak, nokta bulutu, üyesi olan **transform** nesnesi ile dönüştürüldükten sonra döndürülecektir. (Örneğin, bir kamera için nesne yaratırken, bu kameranın taban koordinat eksenine göre dönüşümünü yapacak şekilde transform üye nesnesi konfigüre edilecektir. Bu kameraya ait nesnenin captureFor fonksiyonu çağrıldığında elde edilen noktalar dönüşüm yapılacak ve döndürülecektir. Böylece, nesne noktaları kamera koordinat eksenine göre değil, direk taban koordinat sistemine göre döndürecektir.)

Yeni eklenen sınıflar:

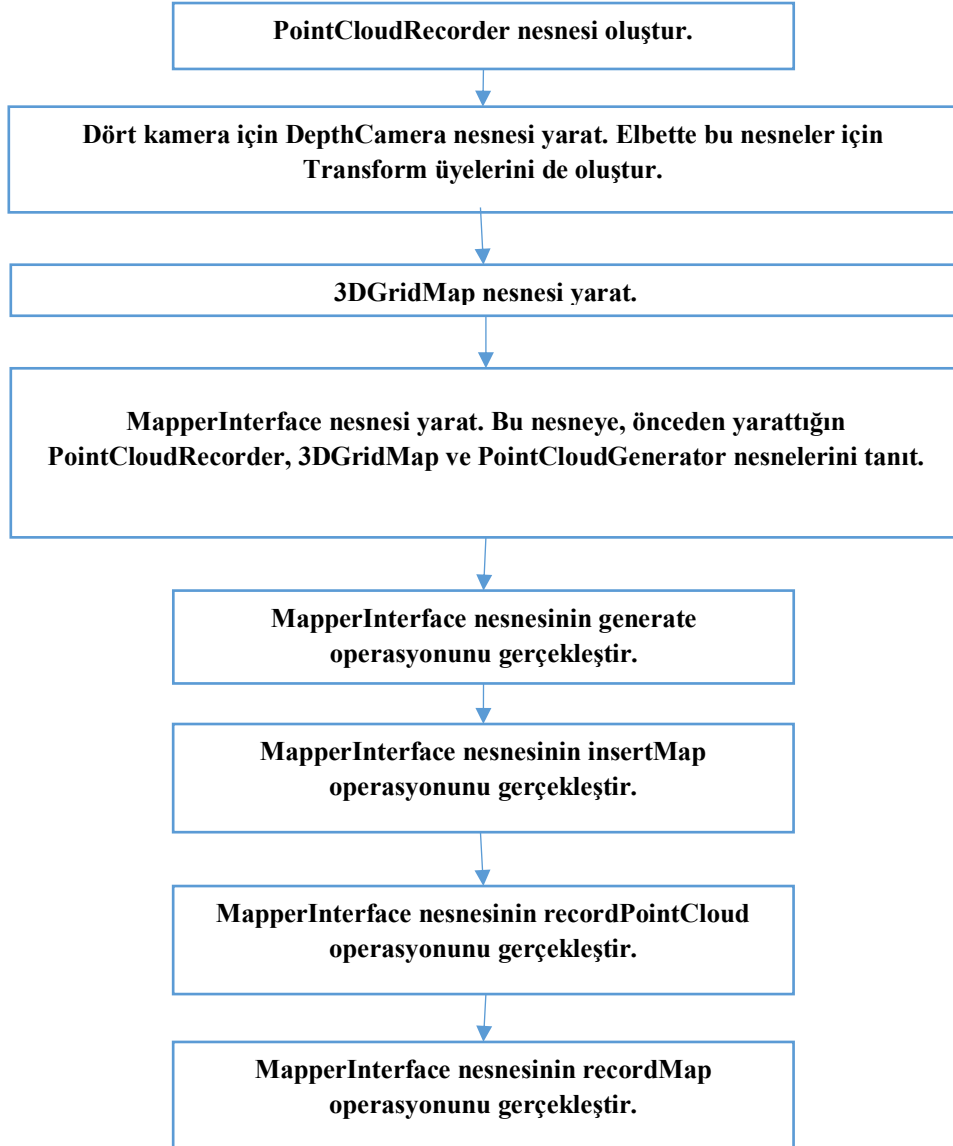
MapperInterface: Bu sınıf işlemleri basitleştirmek için kullanılan bir sınıftır. **generate()** fonksiyonu çağrıldığında, **generators** üyesinde bulunan tüm nesnelerden **captureFor** fonksiyonu çağrılarak nokta bulutları sağlanır. Daha sonra her bir nokta bulutu **pointCloud** üyesine eklenir. **insertMap()** nokta bulutlarını haritaya işler. **recordPointcloud()** ve **recordMap()** fonksiyonları, sırasıyla pointCloud ve haritayı dosyaya kaydeder.

Bölüm 1.

Şekil 3'te verilen tasarıma uygun olarak bir Haritalama Kütüphanesi oluşturun. Her sınıf için doğruluğunu test etmek için bir test programı yazınız. Testler sırasında, Bölüm 2'de verilen örnek değerlerden faydalanabilirsiniz.

Bölüm 2.

Bir test uygulama programı yazınız. Bu test programında, proje I. Aşaması ile verilen dört adet derinlik kamerasından alınan nokta bulutları ve dönüşüm matrisleri kullanılacaktır. **Sonuçlar, projenin daha önce yapılan I. aşaması ile aynı olacaktır.** Bunu dikkate alarak sağlamasını yapabilirsiniz.





Şekil 3. Geliştirilecek yazılımın taslak UML Sınıf Diyagramı

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](#)[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

fA an integer argument

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

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

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 çıktıların kısaca anlatılması.

Görev Atamaları (İçerilmeli)

Grup Üyesi	Görevler
<i>Grup üyesinin adı</i>	<i>Sınıflar, dokümantasyon, etc.</i>
<i>Grup üyesinin adı</i>	...
...	...

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