PROJE BAŞLIĞI: Nokta Bulutu İşleme

Proje Açıklaması:

Kısım 1'de verilen projedeki tasarım üzerinde değişiklikler yapılması istenmektedir. Bu değişiklikler, bazı sınıfların polymorphic yapıya dönüştürülmesi, dizi kullanımı yerine 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 bulutunun 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 sınıfında 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 geliştirmediğiniz açık kaynak kodlu kütüphaneleri, projenizde kullanabilme kabiliyetinin geliştirilmesi)

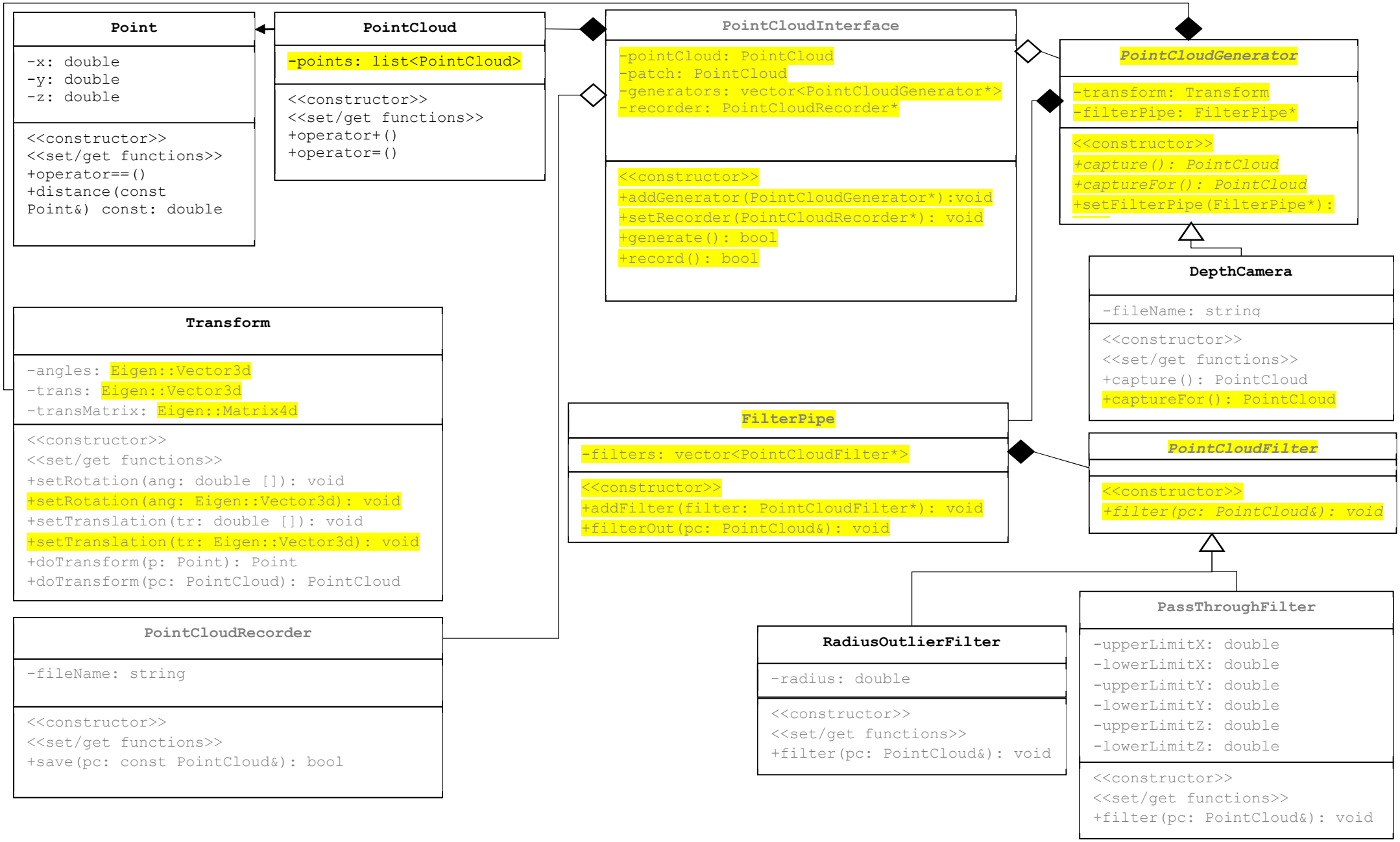
3. Filtreler, **`PointCloudFilter`** abstract sınıfı altında toplanacaktır. **`filter`** üye fonksiyonu pure virtual fonksiyon olacaktır.

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 **`filterpipe`** dan geçirdikten **`ve 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. Filtreler eklenecektir. Bu kameraya ait nesnenin captureFor fonksiyonu çağrıldığında elde edilen noktalar **`filterpipe`** dan geçirilecek ve sonra dönüşüm yapılacak ve döndürülecektir. Böylece, nesne noktaları kamera koordinat eksenine göre değil, direk taban koordinat sistemine göre filtrelenmiş olarak döndürecektir.)

Yeni eklenen sınıflar:

`FilterPipe:` Birden fazla filtreden geçmesi gereken nokta bulutunun filtreleme işlemini yapacak sınıftır. Bu nesneye, faklı tip filtreler ya da farklı parametrelere sahip aynı tip filtreler **`addFilter`** fonksiyonu ile eklenecektir. Daha sonra **`filterOut`** fonksiyonuna verilen nokta bulutu, ekleme sırasında göre birer birer filtrelerden geçirilip, sonuç nokta bulutunu döndürecektir.

`PointCloudInterface:` Bu sınıf işlemleri basitleştirmek için kullanılan bir sınıftır. İki üye fonksiyona sahiptir. **`generate()`** fonksiyonu çağrıldığında, **`generators`** üyesinde bulunan tüm nesnelerden **`captureFor`** fonksiyonu çağrılarak nokta bulutları sağlanır. Daha sonra her bir nokta bulutu **`pointCloud`** üyesine eklenir. Kısaca, proje Kısım 1 'de yaptığını uygulamadaki aşamalar bu tek fonksiyon ile gerçekleştirilir. **`record()`** fonksiyonu ise, pointCloud içindeki noktaları dosyaya kaydeder.



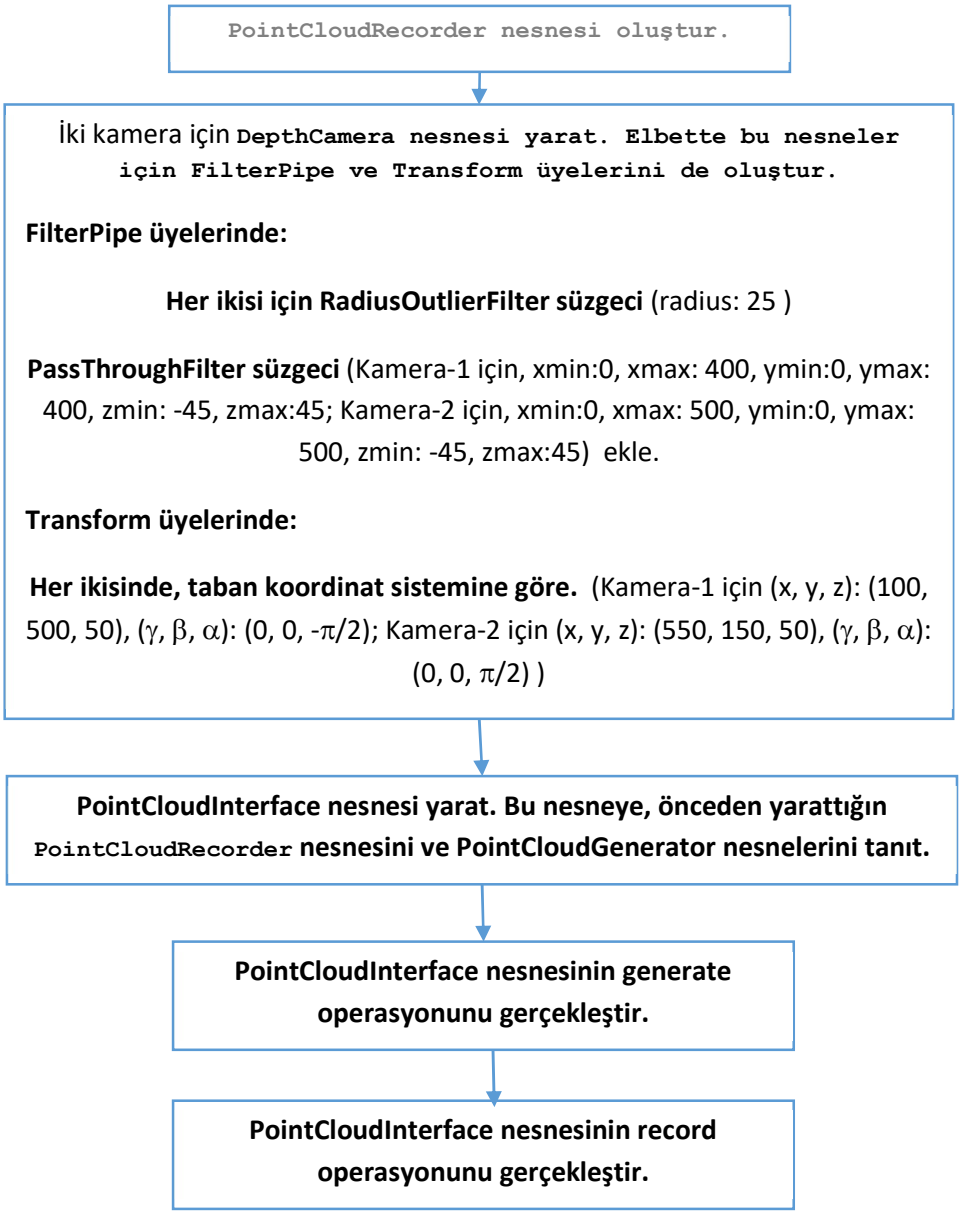
Şekil 1. Geliştirilecek yazılımın taslak UML Sınıf Diyagramı

Bölüm 1.

Yukarıda verilen tasarıma uygun olarak bir Nokta Bulutu 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 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. Aşağıdaki akış diyagramını uygulayınız. **Sonuçlar, projenin daha önce yapılan kısmı ile aynı olmalıdır.** Bunu dikkate alarak sağlamasını yapabilirsiniz.



DOSYA EKLERİ:

EK 1. “camera1.txt” dosyası

EK 2. “camera2.txt” dosyası.