QT ARAYÜZÜ VE İÇERİK BAĞLANTIMIZ

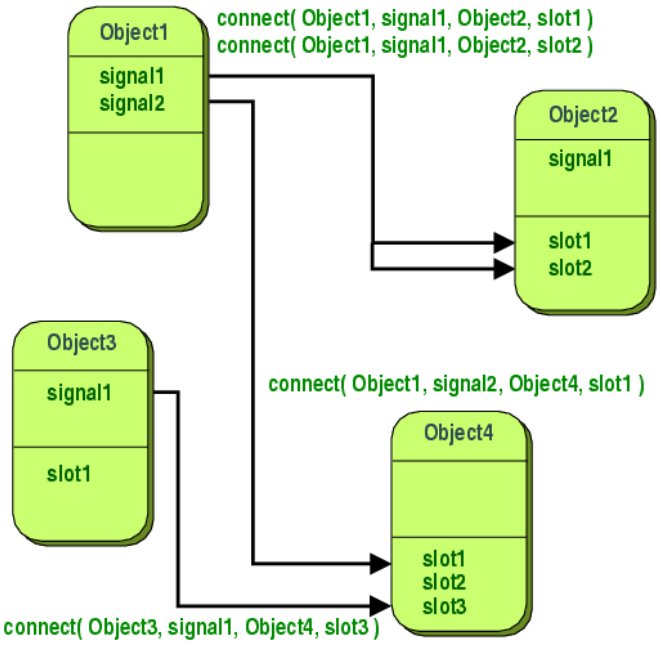
SİNYALLER VE YUVALAR

Nesneler arasındaki iletişim için sinyaller ve yuvalar kullanılır. Sinyaller ve yuvalar mekanizması, Qt'nin merkezi bir özelliğidir ve diğer çerçeveler tarafından sağlanan özelliklerden en farklı olan kısımdır. Sinyaller ve yuvalar, Qt'nin meta-nesne sistemi tarafından mümkün kılınmıştır.

GİRİŞ

GUI programlamada, bir parçacığı değiştirdiğimizde, genellikle başka bir parçacığın bilgilendirilmesini isteriz. Daha genel olarak, her tür nesnenin birbiriyle iletişim kurmasını istiyoruz. Örneğin, bir kullanıcı bir Kapat düğmesini tıklarsa, pencerenin close() işlevinin çağrılmasını ister.

Diğer araç setleri, geri aramaları kullanarak bu tür iletişimi sağlar. Geri arama, bir işleve yönelik bir işaretçidir, bu nedenle, bir işleme işlevinin sizi bazı olaylar hakkında bilgilendirmesini istiyorsanız, bir işaretçiyi başka bir işleve (geri arama) işleme işlevine iletirsiniz. İşleme işlevi daha sonra uygun olduğunda geri aramayı çağırır. Bu yöntemi kullanan başarılı çerçeveler mevcut olsa da geri aramalar sezgisel olmayabilir ve geri arama argümanlarının tür doğruluğunu sağlamada sorun yaşayabilir.



SİNYALLER VE YUVALAR

Qt'de geri arama tekniğine bir alternatifimiz var: Sinyaller ve yuvalar kullanıyoruz. Belirli bir olay meydana geldiğinde bir sinyal yayınlanır. Qt widget'larının önceden tanımlanmış birçok sinyali vardır, ancak kendi sinyallerimizi onlara eklemek için her zaman widget'ları alt sınıflara ayırabiliriz. Bir yuva, belirli bir sinyale yanıt olarak çağrılan bir işlevdir. Qt'nin widget'ları önceden tanımlanmış birçok yuvaya sahiptir, ancak widget'ları alt sınıflara ayırmak ve ilgilendiğiniz sinyalleri işleyebilmeniz için kendi yuvalarınızı eklemek yaygın bir uygulamadır.

Sinyaller ve yuvalar mekanizması tip güvenlidir: Bir sinyalin imzası, alıcı yuvanın imzasıyla eşleşmelidir. (Aslında bir yuvanın aldığı sinyalden daha kısa bir imzası olabilir, çünkü fazladan argümanları yok sayabilir.) İmzalar uyumlu olduğundan, derleyici, işlev işaretçisine dayalı sözdizimini kullanırken tür uyumsuzluklarını tespit etmemize yardımcı olabilir. Dize tabanlı SIGNAL ve SLOT sözdizimi, çalışma zamanında tür uyumsuzluklarını algılar. Sinyaller ve yuvalar gevşek bir şekilde bağlanmıştır: Bir sinyal yayan bir sınıf, sinyali hangi yuvaların aldığını ne bilir ne de önemser. Qt'nin sinyalleri ve yuva mekanizması, bir sinyali bir yuvaya bağlarsanız, yuvanın sinyal parametreleriyle doğru zamanda çağrılmasını sağlar. Sinyaller ve yuvalar, herhangi bir türden herhangi bir sayıda argüman alabilir. Tamamen tip güvenlidirler.

QObject'ten veya alt sınıflarından birinden (örn. QWidget) miras alan tüm sınıflar, sinyaller ve yuvalar içerebilir. Sinyaller, durumlarını diğer nesneler için ilginç olabilecek bir şekilde değiştirdiklerinde nesneler tarafından yayılır. Nesnenin iletişim kurmak için yaptığı tek şey bu. Yayınladığı sinyalleri herhangi bir şeyin alıp almadığını bilmiyor ya da umursamıyor. Bu gerçek bilgi kapsüllemedir ve nesnenin bir yazılım bileşeni olarak kullanılabilmesini sağlar.

```
#include <QObject>

class Counter : public QObject
{
    Q_OBJECT

public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }

public slots:
    void setValue(int value);

signals:
    void valueChanged(int newValue);

private:
    int m_value;
};
```

Yuvalar sinyal almak için kullanılabilir, ancak bunlar aynı zamanda normal üye işlevleridir. Bir nesnenin herhangi bir şeyin kendi sinyallerini alıp almadığını bilmemesi gibi, bir yuva da kendisine bağlı herhangi bir sinyal olup olmadığını bilmez. Bu, Qt ile gerçekten bağımsız bileşenlerin oluşturulabilmesini sağlar.

Tek bir yuvaya istediğiniz kadar sinyal bağlayabilir ve ihtiyacınız olduğu kadar yuvaya bir sinyal bağlanabilir. Bir sinyali doğrudan başka bir sinyale bağlamak bile mümkündür. (Bu, ilk sinyal gönderildiğinde hemen ikinci sinyali yayar.)

Sinyaller ve yuvalar birlikte güçlü bir bileşen programlama mekanizması oluşturur.

İŞARETLER

Sinyaller, iç durumu nesnenin müşterisi veya sahibi için ilginç olabilecek bir şekilde değiştiğinde bir nesne tarafından yayılır. Sinyaller genel erişim işlevleridir ve herhangi bir yerden gönderilebilir, ancak bunları yalnızca sinyali ve alt sınıflarını tanımlayan sınıftan yayınlamanızı öneririz.

Bir sinyal gönderildiğinde, ona bağlı slotlar genellikle normal bir fonksiyon çağrısı gibi hemen yürütülür. Bu olduğunda, sinyaller ve yuvalar mekanizması herhangi bir GUI olay döngüsünden tamamen bağımsızdır. İfadeyi izleyen kodun yürütülmesi, emittüm yuvalar geri döndüğünde gerçekleşir. Sıraya alınmış bağlantılar kullanıldığında durum biraz farklıdır ; böyle bir durumda, emitanahtar kelimeyi takip eden kod hemen devam edecek ve slotlar daha sonra çalıştırılacaktır.

Bir sinyale birkaç yuva bağlanırsa, yuvalar, sinyal verildiğinde, bağlanma sırasına göre birbiri ardına yürütülür.

Sinyaller moc tarafından otomatik olarak oluşturulur ve .cpp dosyaya uygulanmamalıdır. Asla dönüş türlerine sahip olamazlar (yani use void).

YUVALAR

Bir yuva, kendisine bağlı bir sinyal yayıldığında çağrılır. Yuvalar normal C ++ işlevleridir ve normal olarak çağrılabilir; Onların tek özelliği, sinyallerin bunlara bağlanabilmesidir. Yuvalar normal üye işlevler olduğundan, doğrudan çağrıldıklarında normal C ++ kurallarını izlerler. Bununla birlikte, yuvalar olarak, erişim düzeyine bakılmaksızın herhangi bir bileşen tarafından bir sinyal yuvası bağlantısı aracılığıyla çağrılabilirler. Bu, rastgele bir sınıfın örneğinden yayılan bir sinyalin, ilişkisiz bir sınıfın örneğinde özel bir yuvanın çağrılmasına neden olabileceği anlamına gelir. Geri çağırmalarla karşılaştırıldığında sinyaller ve yuvalar, sağladıkları artan esneklik nedeniyle biraz daha yavaştır, ancak gerçek uygulamalar için fark önemsizdir. Genel olarak, bazı slotlara bağlı bir sinyal yaymak, alıcıları sanal olmayan işlev çağrılarıyla doğrudan aramaktan yaklaşık on kat daha yavaştır. Bu, bağlantı nesnesini bulmak, tüm bağlantılar üzerinde güvenli bir şekilde yinelemek (yani, emisyon sırasında sonraki alıcıların imha edilmediğini kontrol etmek) ve herhangi bir parametreyi genel bir şekilde sıralamak için gereken ek yüktür. Sanal olmayan on işlev çağrısı kulağa çok benziyor olsa da new, delete, örneğin herhangi bir işlemden çok daha az ek yük. Sahne arkasında gerektiren bir dizgi, vektör veya liste işlemi gerçekleştirir gerçekleştirmez new veya delete, sinyaller ve yuva ek yükü, tam işlev çağrısı maliyetlerinin yalnızca çok küçük bir kısmından sorumludur. Aynı şey, bir yuvada bir sistem çağrısı yaptığınızda da geçerlidir veya dolaylı olarak ondan fazla işlevi çağırın. Sinyallerin ve yuva mekanizmasının basitliği ve esnekliği, kullanıcılarınızın fark etmeyeceği ek yüke değer.

Qt tabanlı bir uygulamayla birlikte derlendiğinde, çağrılan signals veya slots derleyici uyarılarına ve hatalarına neden olabilecek değişkenleri tanımlayan diğer kitaplıkların olduğunu unutmayın. Bu sorunu çözmek için #undef, rahatsız edici önişlemci sembolü.

```
void MainWindow::connectSignalSlots() {
    //connect signals/slots
    //load/save/open export folder
    connect(ui->pushButton_load,             SIGNAL(clicked()), this, SLOT(loadUserFilePath()));
    connect(ui->pushButton_save,             SIGNAL(clicked()), this, SLOT(saveUserFilePath()));
    connect(ui->pushButton_openExportFolder, SIGNAL(clicked()), this, SLOT(openExportFolder()));
    //zoom
    connect(ui->pushButton_zoomIn,    SIGNAL(clicked()), this, SLOT(zoomIn()));
    connect(ui->pushButton_zoomOut,   SIGNAL(clicked()), this, SLOT(zoomOut()));
    connect(ui->pushButton_resetZoom, SIGNAL(clicked()), this, SLOT(resetZoom()));
    connect(ui->pushButton_fitInView, SIGNAL(clicked()), this, SLOT(fitInView()));
    //calculate
    connect(ui->pushButton_calcNormal,   SIGNAL(clicked()), this, SLOT(calcNormalAndPreview()));
    connect(ui->pushButton_calcSpec,     SIGNAL(clicked()), this, SLOT(calcSpecAndPreview()));
    connect(ui->pushButton_calcDisplace, SIGNAL(clicked()), this, SLOT(calcDisplaceAndPreview()));
    connect(ui->pushButton_calcSsao,     SIGNAL(clicked()), this, SLOT(calcSsaoAndPreview()));
    //switch between tabs
    connect(ui->tabWidget, SIGNAL(tabBarClicked(int)),  this, SLOT(preview(int)));
    connect(ui->tabWidget, SIGNAL(currentChanged(int)), this, SLOT(preview(int)));
```

3 BOYUTLU MODELLERİ DOKULANDIRMA

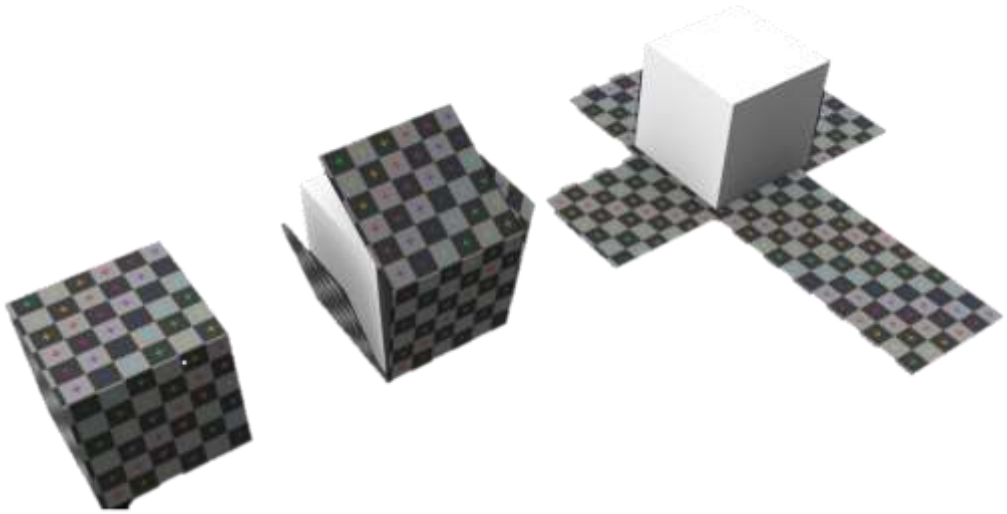
DOKU (TEXTURE) NEDİR?

Bilgisayar grafiklerinde görüntüler kendiliğinden oluşturulamaz. Bilgisayar grafiklerinde gerçek dünya ile referans oluşturmak amacıyla çeşitli görüntü dosyaları ile kurulumlar yapılır. Yapılan bu kurulumlar daha sonra render (oluşturucu) programları vasıtasıyla ekranlarımızda gerçek gibi görünen görseller oluşturabilir. Doku eşleme bilgisayar tarafından oluşturulan bir grafik veya 3D modelde yüksek frekanslı ayrıntı, yüzey dokusu veya renk bilgilerini tanımlamak için bir yöntemdir. Doku eşleme başlangıçta, pikselleri bir dokudan bir 3B yüzeye basitçe eşler. Son birkaç on yılda, çok geçişli işleme gelişi, çoklu doku , Eşleşme gibi, ve daha karmaşık eşlemeler yüksekliği haritalama, normal bir haritalama, yer değiştirme haritalama, yansıma haritalama, speküler haritalama, oklüzyon haritalama tekniğine ve başka bir takım varyasyonlar gelişti. Doku eşlem, bir şeklin veya çokgenin yüzeyine uygulanan (eşlenen) bir görüntüdür. Bu bir bitmap görüntüsü veya prosedürel bir doku olabilir. Bunlar, ortak görüntü dosyası formatlarında depolanabilir, 3B model formatları veya malzeme tanımları ile referans gösterilebilir ve kaynak demetleri halinde birleştirilebilirler.

Görünür yüzeyler için en yaygın 2 boyut olmasına rağmen, 1-3 boyuta sahip olabilirler. Modern donanımla kullanım için, doku eşlemi verileri, önbellek tutarlılığını iyileştirmek için sıralı veya döşemeli sıralarda saklanabilir. Oluşturma API'leri tipik olarak doku eşleme kaynaklarını (aygıt belleğinde bulunabilen) tamponlar veya yüzeyler olarak yönetir ve sonradan işleme veya ortam haritalama gibi ek etkiler için ' dokuya dönüştürmeye ' izin verebilir.

Genellikle RGB renk verilerini (doğrudan renk, sıkıştırılmış formatlar veya indekslenmiş renk olarak depolanır) ve bazen özellikle reklam panoları ve çıkartma kaplama dokuları için alfa karıştırma (RGBA) için ek bir kanal içerirler. Spekülerlik gibi diğer kullanımlar için alfa kanalını (donanım tarafından ayrıştırılan formatlarda saklamak uygun olabilir) kullanmak mümkündür.

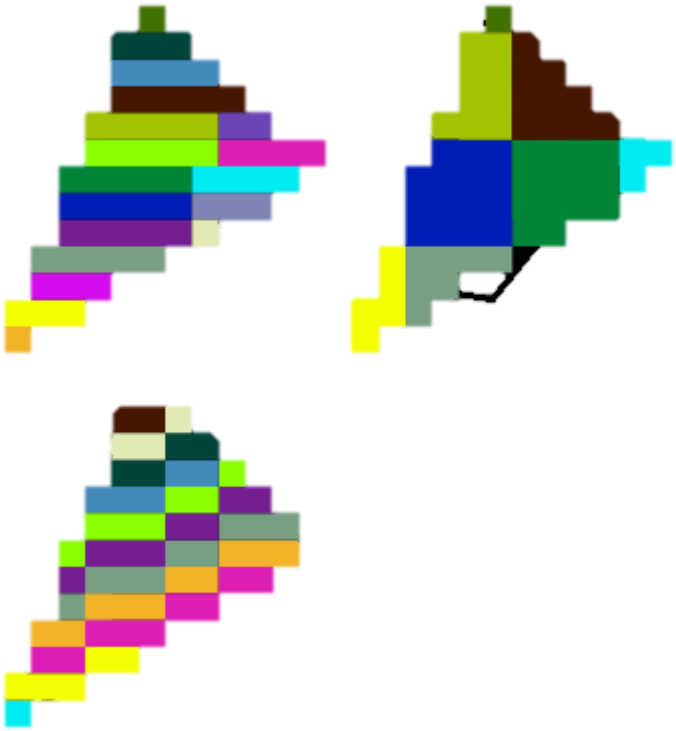
Spekülerlik, normaller, yer değiştirme veya yüzey altı saçılması üzerinde kontrol için çoklu doku haritaları (veya kanallar) birleştirilebilir. Modern donanım için durum değişikliklerini azaltmak için doku atlaslarında veya dizi dokularında birden fazla doku görüntüsü birleştirilebilir. Modern donanım genellikle ortam haritalaması için birden çok yüze sahip küp harita dokularını destekler. Bu işlem, desenli kâğıdı düz beyaz bir kutuya uygulamaya benzer. Bir çokgendeki her köşe noktasına bir doku koordinatı atanır. (2d durumunda UV koordinatları olarak da bilinir).



DOKU ALANI

Doku eşleme, model yüzeyini (rasterleştirme sırasında ekran alanını) doku uzayıyla eşler; bu alanda doku eşlemi bozulmamış haliyle görülebilir. UV sarmalama araçları tipik olarak doku koordinatlarının manuel olarak düzenlenmesi için doku alanında bir görünüm sağlar. Yüzey altı saçılması gibi bazı işleme teknikleri, yaklaşık olarak doku-uzay işlemleri ile gerçekleştirilebilir.

DOKU FİLTRELEME



Doku koordinatının dokunun dışında olması durumunda ya sabitlenir ya da sarılır. Anizotropik filtreleme dokuları eğik bakış açılarından görüntülerken yönsel bozuklukları daha iyi ortadan kaldırır.

DOKU AKIŞI

Doku akışı, hangi dokunun hafızaya yüklenmesi ve izleyiciden çekme mesafesine ve ne kadar hafıza için kullanılabileceğine bağlı olarak, her dokunun iki veya daha fazla farklı çözünürlükte mevcut olduğu dokular için veri akışlarını kullanmanın bir yoludur. Doku akışı, görüntü oluşturma motorunun, izleyicinin kamerasından uzaktaki nesneler için düşük çözünürlüklü dokuları kullanmasına ve bunları daha ayrıntılı dokulara dönüştürmesine, bakış açısı nesnelere yaklaştıkça bir veri kaynağından okunmasına olanak tanır.

PİŞİRME (TEXTURE BAKİNG)

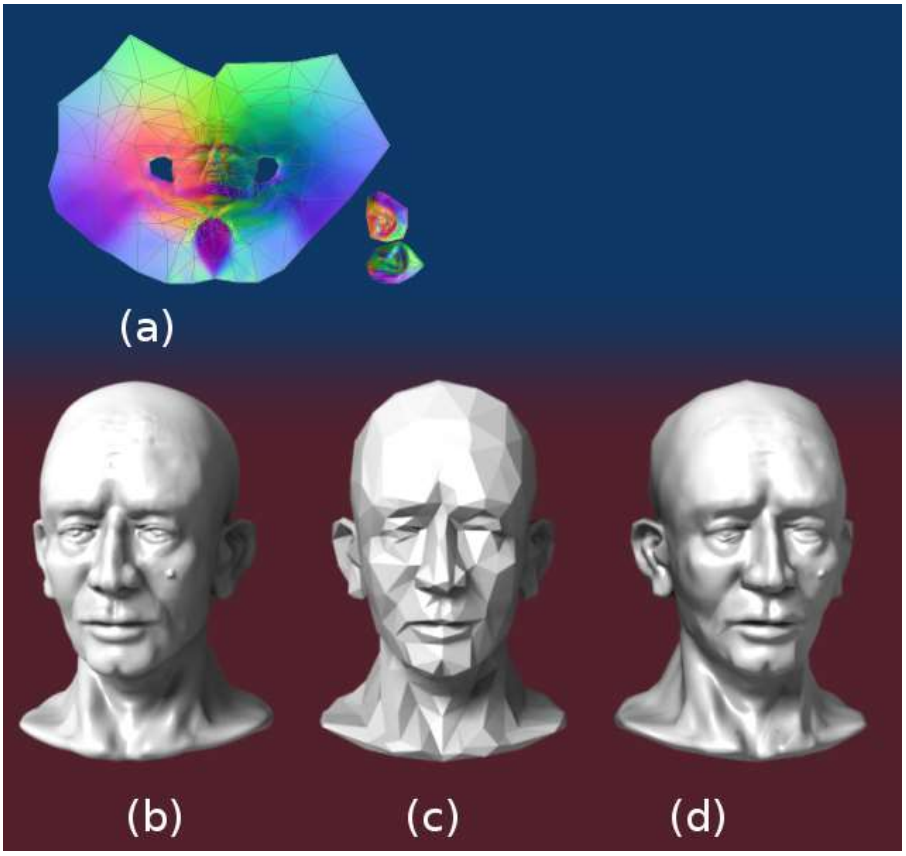
Bir optimizasyon olarak, karmaşık, yüksek çözünürlüklü bir modelden veya pahalı bir işlemden (küresel aydınlatma gibi) bir yüzey dokusuna (muhtemelen düşük çözünürlüklü bir modelde) ayrıntı vermek mümkündür. Baking render eşleme olarak da bilinir. Bu teknik en yaygın olarak ışıklı haritalar için kullanılır, ancak normal haritalar ve yer değiştirme haritaları oluşturmak için de kullanılabilir. Bazı bilgisayar oyunları bu tekniği kullanmıştır. Orijinal Quake yazılım motoru, ışık haritalarını ve renk haritalarını (" yüzey önbelleğe alma ") birleştirmek için anında baking kullandı.

Pişirme, birçok farklı öğe ve malzemeye sahip karmaşık bir sahnenin tek bir dokuya sahip tek bir öğe tarafından yaklaştırılabildiği ve daha sonra daha düşük işleme maliyeti ve daha az çekiliş için algoritmik olarak azaltıldığı bir ayrıntı düzeyi oluşturma biçimi olarak kullanılabilir . Ayrıca, 3B şekillendirme yazılımından ve nokta bulutu taramasından yüksek detaylı modeller almak ve bunları gerçek zamanlı işleme için daha uygun ağlarla yaklaştırmak için kullanılır.

NORMAL HARİTASI

GİRİŞ

3D Bilgisayar grafiklerinde, normal bir haritalama, yükseklik ve alçaklık aydınlatmasını taklit etmek için kullanılan bir tekniktir. Daha fazla poligon kullanmadan detay eklemek için kullanılır. Bu tekniğin yaygın bir kullanımı, yüksek çokgen modelden veya yükseklik haritasından normal bir harita oluşturarak düşük çokgen modelinin görünümünü ve ayrıntılarını büyük ölçüde geliştirmektir.



Normal haritalar, RGB bileşenlerinin sırasıyla yüzey normalinin X, Y ve Z koordinatlarına karşılık geldiği normal RGB görüntüleri olarak depolanır.

YÖNTEM

Birim vektör ışık kaynağına gölgeleme noktasından olan noktalı bu yüzeye normal birim vektör ile ve sonuç yüzey üzerinde ışığın yoğunluğudur. Model boyunca dokulandırılmış 3 kanallı bir bitmap kullanılarak, daha ayrıntılı normal vektör bilgileri kodlanabilir. Bitmap'teki her kanal bir uzaysal boyuta (X, Y ve Z) karşılık gelir. Bu, özellikle gelişmiş aydınlatma teknikleriyle bağlantılı olarak bir modelin yüzeyine çok daha fazla ayrıntı ekler.

NORMAL HARİTASI OLUŞTURMA

Yaygın dokudan normal haritayı hesaplamak için sobel filtresi ya da prewitt filtresi gri ölçekli dağınık görüntüye uygulanır.

SOBEL METODU

Kenar Algılama, basitçe, yoğunlukta keskin bir değişikliğin veya renkte keskin bir değişikliğin olduğu, yüksek bir değerin keskin bir değişikliği ve düşük bir değerin sığ bir değişikliği ifade ettiği bir görüntüdeki bölgeleri bulmaya çalışmanın bir örneğidir.

Sobel Operatörü:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

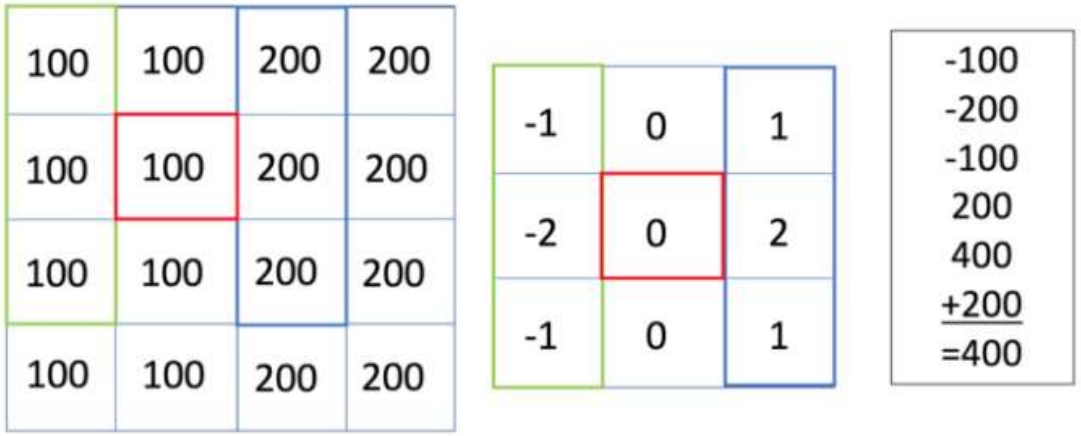
+1	+2	+1
0	0	0
-1	-2	-1

Gy

Bunu yapmak için çok yaygın bir operatör, bir görüntünün türevine bir yaklaşım olan bir Sobel Operatörüdür. Y ve X yönlerinde ayrıdır. X yönüne bakarsak, bir görüntünün x yönündeki eğimi buradaki operatöre eşittir. Her x ve y yönü için 3'e 3'lük bir çekirdek kullanıyoruz. X yönü için gradyan sol tarafta eksi sayılara ve sağ tarafta pozitif sayılara sahiptir ve merkez piksellerin birazını koruyoruz. Benzer şekilde, y yönü için gradyan altta eksi sayılara ve üstte pozitif sayılara sahiptir ve burada orta satır piksellerinde biraz korunuyoruz.

Sobel Operator, gradyan matrisini görselimizin her pikselinin üzerine yerleştirerek farkın miktarını bulmaya çalışır.

Biri X-Yönü ve diğeri Y-Yönü için çıktı olarak iki görüntü elde ederiz. Kernel Convolution kullanarak, aşağıdaki örnek görüntüde 100 ve 200 değerlik sütun arasında bir kenar olduğunu görebiliriz.



Yukarıdaki örnek, Gradyan matrisi X'i kırmızı işaretli 100 görüntünün üzerine yerleştirerek evrişim yapmanın sonucunu göstermektedir. Hesaplama, sıfır olmayan 400'e kadar olan toplamı sağda gösterilir, dolayısıyla bir kenar vardır. Görüntülerin tüm pikselleri aynı değerde olsaydı, bu durumda evrişim sonuçta sıfır toplamı ile sonuçlanırdı. Dolayısıyla gradyan matrisi, bir taraf daha parlak olduğunda büyük bir yanıt verecektir. Burada dikkat edilmesi gereken bir diğer nokta, sonuç çıktısının işaretinin önemli olmadığıdır.

Özetle Sobel Operatör yöntemi, herhangi bir insan müdahalesi olmadan bir görüntüyü bölümlere ayırmak için başarıyla kullanılabilir. Sobel Operatörünün de uygulaması çok hızlıdır. Bir görüntü üzerinde her çalıştırdığınızda aynı çıktıyı ürettiğinden, Sobel Operator'ü görüntü bölümleme için kararlı bir kenar algılama tekniği yapar.

Tuğla duvar ve bisiklet askısının gri tonlamalı test görüntüsü

Sobel – Feldman operatöründen normalleştirilmiş gradyan büyüklüğü

Sobel – Feldman operatöründen normalleştirilmiş *x*- gradyanı

Sobel – Feldman operatöründen normalleştirilmiş *y*- gradyanı

PREWİTT METODU

Prewitt operatörü, bir görüntüde kenar algılama için kullanılır. İki tür kenarı algılar

- Yatay kenarlar
- Dikey Kenarlar
-

Kenarlar, bir görüntünün karşılık gelen piksel yoğunlukları arasındaki fark kullanılarak hesaplanır. Kenar algılama için kullanılan tüm maskeler, türev maskeler olarak da bilinir. Çünkü bu eğitim dizisinde daha önce birçok kez belirttiğimiz gibi, bu görüntü de bir sinyaldir, bu nedenle bir sinyaldeki değişiklikler yalnızca farklılaşma kullanılarak hesaplanabilir. Bu nedenle bu operatörler, türev operatörler veya türev maskeler olarak da adlandırılır.

Tüm türev maskeler aşağıdaki özelliklere sahip olmalıdır:

- Maskede tam tersi işaret bulunmalıdır.
- Maske toplamı sıfıra eşit olmalıdır.
- Daha fazla ağırlık, daha fazla kenar algılama anlamına gelir.

Prewitt operatörü bize, biri yatay yönde kenarları, diğeri dikey yönde kenarları algılamak için iki maske sağlar.

Dikey yön

-1 0 1

-1 0 1

-1 0 1

Maskenin üstündeki kenarlar dikey yöndeki kenarları bulacaktır ve bunun nedeni dikey yöndeki sıfırlar sütunudur. Bu maskeyi bir görüntüye bağladığınızda, size bir görüntüdeki dikey kenarları verecektir.

Nasıl Çalışır?

Bu maskeyi görüntüye uyguladığımızda, göze çarpan dikey kenarlar ortaya çıkıyor. Basitçe birinci dereceden türev gibi çalışır ve bir kenar bölgesindeki piksel yoğunluklarının farkını hesaplar. Merkez sütun sıfır olduğundan, bir görüntünün orijinal değerlerini içermez, bunun yerine o kenarın etrafındaki sağ ve sol piksel değerlerinin farkını hesaplar. Bu, kenar yoğunluğunu artırır ve orijinal görüntüye kıyasla daha iyi hale gelir.

Yatay Yön

-1 -1 -1

0 0 0

1 1 1

Maskenin üstündeki kenarlar yatay yönde kenarları bulacaktır ve bunun nedeni bu sıfırlar sütununun yatay yönde olmasıdır. Bu maskeyi bir görüntüye dönüştürdüğünüzde, görüntüde yatay kenarlar belirginleşecektir.

Nasıl Çalışır?

Bu maske, bir görüntüdeki yatay kenarları öne çıkaracaktır. Ayrıca yukarıdaki maske prensibine göre çalışır ve belirli bir kenarın piksel yoğunlukları arasındaki farkı hesaplar. Maskenin merkez sırası sıfırlardan oluştuğu için görüntüdeki orijinal kenar değerlerini içermez, bunun yerine belirli kenarın üstündeki ve altındaki piksel

yoğunluklarının farkını hesaplar. Böylece ani yoğunluk değişimini arttırır ve kenarı daha görünür kılar. Yukarıdaki maskelerin her ikisi de türev maskesi ilkesini izler. Her iki maskede de zıt işaret vardır ve her iki maskenin toplamı sıfıra eşittir. Yukarıdaki her iki maske de standartlaştırıldığı ve içlerindeki değeri değiştiremeyeceğimiz için üçüncü koşul bu operatörde geçerli olmayacaktır.

Tuğla duvar ve bisiklet askısının gri tonlamalı görüntüsü

Bir tuğla duvar ve bir bisiklet rafının gri tonlamalı görüntüsünün Prewitt operatörüyle gradyan

NORMAL HARİTASI ÇALIŞMASI

(Soldan) hesaplandığı sahne ile normal bir harita (merkez) ve düz bir yüzeye uygulandığında elde edilen sonuç (sağda) örneği. Bu harita teğet uzayda kodlanmıştır.

Bir yüzeyin Lambertian (dağınık) aydınlatmasını hesaplamak için, gölgeleme noktasından ışık kaynağına kadar olan birim vektör noktalı birim vektör o yüzeye normaldir ve sonucu, o yüzeydeki ışığın yoğunluğudur. Bir kürenin poligonal bir modelini hayal edelim, yalnızca yüzeyin şeklini tahmin edebilirsiniz. Model genelinde dokulu 3 kanallı bir bit eşlem kullanılarak, daha ayrıntılı normal vektör bilgileri kodlanabilir. Bitmap'teki her kanal bir uzamsal boyuta (X, Y ve Z) karşılık gelir. Bu uzamsal boyutlar, nesne-uzay normal haritaları için sabit bir koordinat sistemine veya teğet uzay normal haritaları durumunda düzgün şekilde değişen bir koordinat sistemine (doku koordinatlarına göre konumun türevlerine dayalı olarak) görelidir. Bu, özellikle gelişmiş aydınlatma teknikleriyle bağlantılı olarak, bir modelin yüzeyine çok daha fazla ayrıntı ekler.

U, V doku koordinatına karşılık gelen birim normal vektörler, normal eşlemlere eşlenir. İzleyiciden uzaklaşan geometriler üzerindeki vektörler asla gösterilmediğinden, yalnızca izleyiciye işaret eden vektörler (Sol El Yönlendirme için z: 0 ila -1) mevcuttur.

Eşleştirme aşağıdaki gibidir:

X: -1 ila +1: Kırmızı: 0 ila 255

Y: -1 ila +1 : Yeşil: 0 ila 255

Z: 0 ila -1 : Mavi: 128 ila 255

- Doğrudan izleyiciye (0,0, -1) dönük normal bir işaret (128,128,255) ile eşleştirilir. Bu nedenle, nesnenin doğrudan bakana bakan kısımları açık mavidir. Normal bir haritada en yaygın renk.
- Dokunun (1,1,0) sağ üst köşesini gösteren normal bir işaret (255,255,128) ile eşlenir. Bu nedenle, bir nesnenin sağ üst köşesi genellikle açık sarıdır. Bir renk haritasının en parlak kısmı.
- Dokunun (1,0,0) sağına normal bir işaret (255,128,128) ile eşlenir. Bu nedenle, bir nesnenin sağ kenarı genellikle açık kırmızıdır.
- Dokunun (0,1,0) üstüne normal bir işaret (128,255,128) ile eşlenir. Bu nedenle, bir nesnenin üst kenarı genellikle açık yeşildir.
- Dokunun (-1,0,0) soluna normal bir işaret (0,128,128) ile eşlenir. Bu nedenle, bir nesnenin sol kenarı genellikle koyu camgöbeğidir.
- Dokunun (0, -1,0) altını gösteren normal bir işaret, (128,0,128) ile eşlenir. Bu nedenle, bir nesnenin alt kenarı genellikle koyu magentadır.

- Dokunun (-1,-1,0) sol alt köşesine normal bir işaret (0,0,128) ile eşlenir. Dolayısıyla, bir nesnenin sol alt köşesi genellikle koyu mavidir. Bir renkli haritanın en karanlık kısmı.

Yaygın aydınlatma hesaplaması için nokta çarpım hesaplamasında bir normal kullanılacağından, {0, 0, –1} değerlerinin {128, 128, 255} değerlerine yeniden eşlenerek bu tür bir gök mavisi verileceğini görebiliriz. Normal haritalarda görülen renk (mavi (z) koordinatı perspektif (derinlik) koordinatı ve ekranda RG-xy düz koordinatlarıdır). {0.3, 0.4, –0.866}, ({0.3, 0.4, –0.866} / 2 + {0.5, 0.5, 0.5}) * 255 = {0.15 + 0.5, 0.2 + 0.5 - 0.433 + 0.5} olarak yeniden eşlenecektir. *255 = {0.65, 0.7, 0.067} * 255 = {166, 179, 17} değer (0,3 ^ {2} + 0,4 ^ {2} + (-0,866) ^ {2} = 1). Z koordinatının (mavi kanal) işareti, normal haritanın normal vektörü ile gözünki (bakış açısı veya kamera) veya ışık vektörünü eşleştirmek için çevrilmelidir. Negatif z değerleri, tepe noktasının kameranın önünde (kameranın arkasında değil) olduğu anlamına geldiğinden, bu kural, ışık vektörü ve normal vektör çakıştığı zaman yüzeyin tam olarak maksimum güçle parlamasını garanti eder.

SPEKULAR HARİTASI

Bilgisayar grafiklerinde, bir yüzeyin sahip olduğu yansıtma miktarını temsil eden üç boyutlu (3B) görüntülemede kullanılan miktar anlamına gelir. Bu belirlemede önemli bir bileşeni olan parlaklık ve speküler olayları ile birlikte parıldayan kısımları belirlemek için önemli. Sıklıkla gerçek zamanlı bilgisayar grafikleri ve ışın izlemede kullanılır, burada diğer yüzeylerden gelen ışığın ayna benzeri aynasal yansıması genellikle göz ardı edilir (bunu hesaplamak için gereken daha yoğun hesaplamalar nedeniyle) ve ışığın doğrudan noktasal ışık kaynakları aynasal vurgular olarak modellenmiştir. Speküler haritalama bir malzeme sisteminde, ek doku haritası katmanları tarafından kontrol edilen bir yüzey boyunca spekülaritenin değişmesine izin verebilir.

SPEKULAR HARİTASI OLUŞTURMA

Yaygın dokudan spekülerliği hesaplamak için gauss (DOG) filtrelerinin farkı uygulanır.

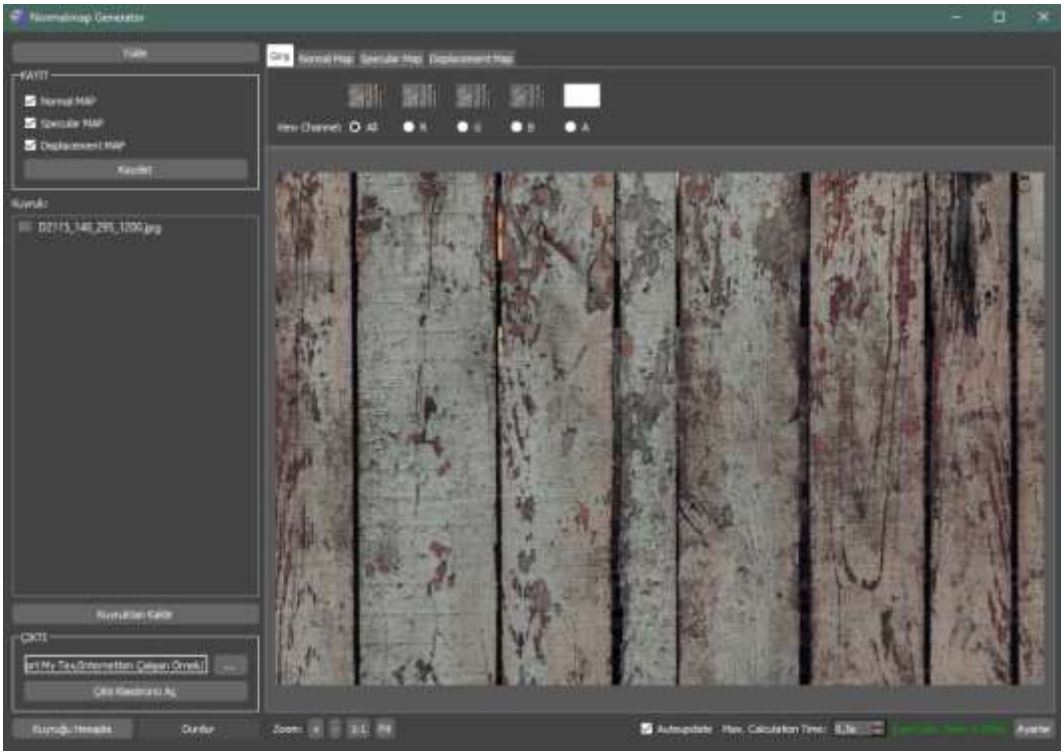
DISPLACEMENT HARİTASI

DISPLACEMENT HARİTASI OLUŞTURMA

NORMAL MAP GENERATOR ÇALIŞMASI

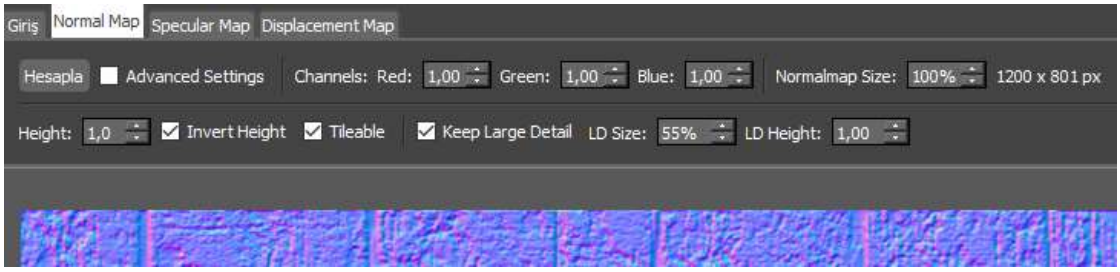
ANA PENCERE

Ana pencere aşağıda açıklanacak kısımlardan oluşmaktadır. Temel olarak en sade görünümde kalınması planlanmıştır.



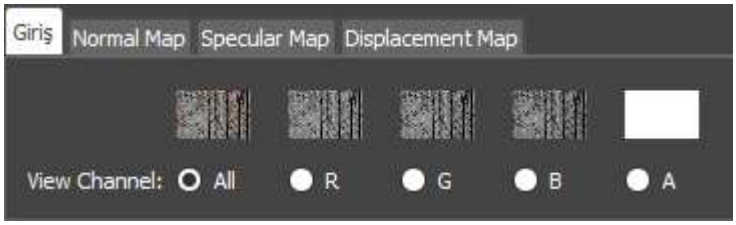
TAB MENU

Programımızın işlevselliğini kazandıran grafik görünümünün farklı harita türleri için değiştirelibilir özelliklerin konumlandırıldığı yönetim kısmıdır. Tablar birbirlerinin üstüne binerek ilgili harita ayarları için çok daha fazla alan sağlamaktadır.



GİRİŞ TAB

Kullanıcının dosya iletişim kutusundan ya da işletim sistemi sürükle bırak yöntemiyle kuyruk kısmına bıraktığı görüntü dosyasını belleğe yükler ve ilgili renk kanalları ile birlikte grafik görünümünde kullanıcıya gösterir.

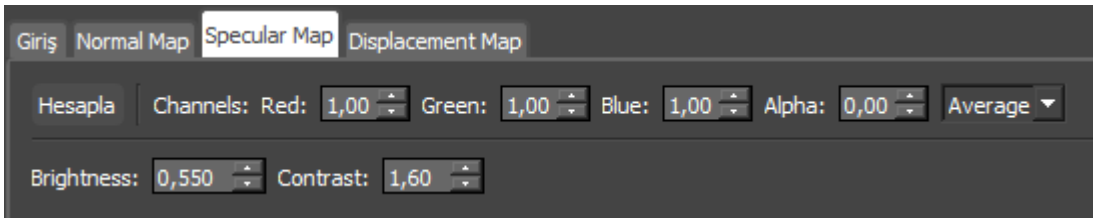


NORMAL TAB

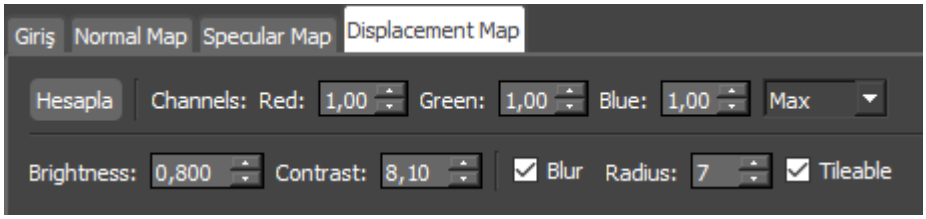
Giriş tarafından alınan görüntü dosyasını kullanıcının ayarları ile birlikte detaylandırarak normal haritasına çevrilip grafik görünümde gösterecek paneldir.

SPECULAR TAB

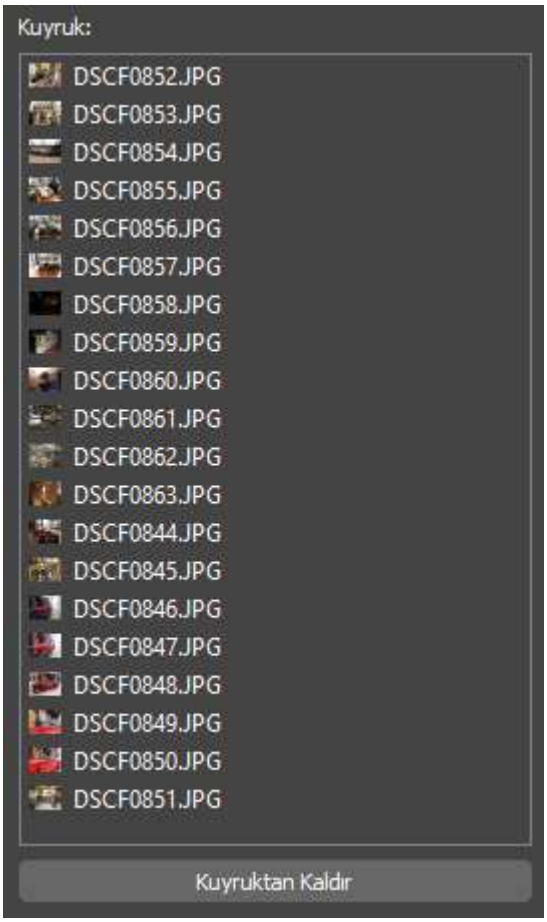
Giriş haritasından alınan görüntünün yansıma haritasına hesaplandığı kısımdır. Giriş tarafından alınan görüntü dosyasını kullanıcının ayarları ile birlikte detaylandırarak grafik görünümde gösterecek paneldir.



DISPLACEMENT TAB



KUYRUK



Birden fazla hesaplama yapmak adına oluşturulan bölümdür. Kuyruktaki her resim bir iş parçasıdır. Kuyruktaki her görüntü çift tıklanılarak düzenlenebilir. Kuyruktaki her görüntü yapılan ayarları kendisinde kaydetmektedir. Kuyruk kayıt işlemi başlatılana kadar bellekte beklemektedirler. Çift tıklama eyleminde grafik görünümüne aktarılırlar ve ilgili tab sekmesinde hesaplamaları yapılır.

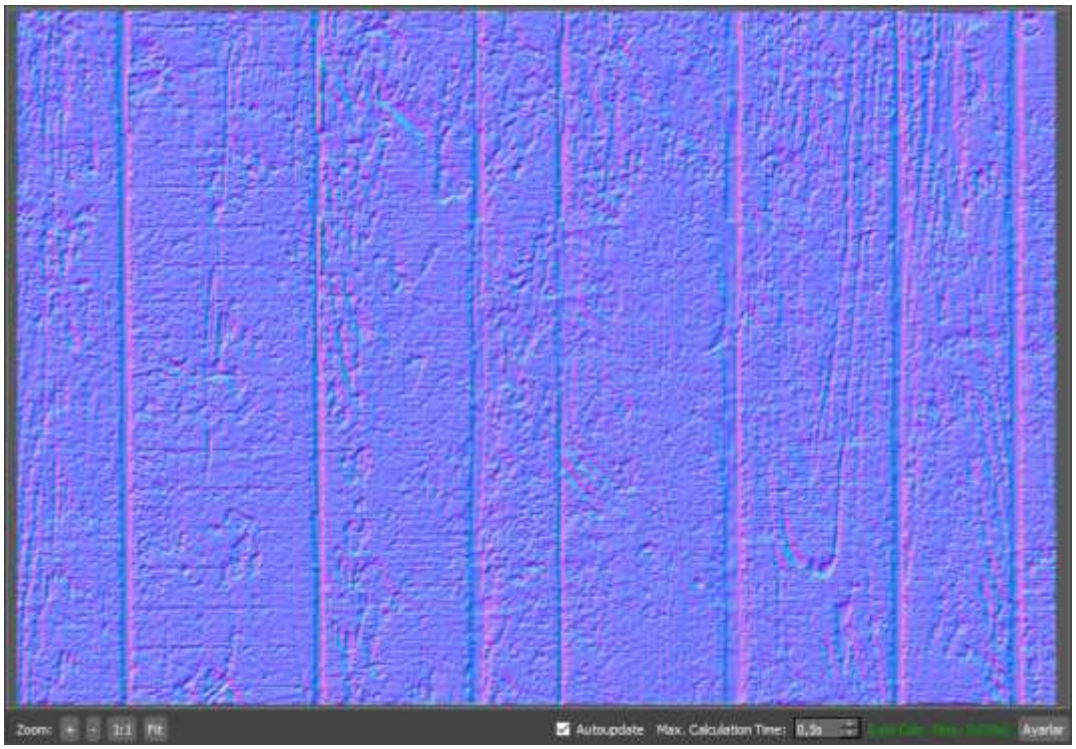
GRAFİK GÖRÜNÜMÜ

Yüklenilen görüntülerin kullanıcı manipülasyonların çıkan sonuçları görmesi için bu kısım oluşturulmuştur. Grafik görünümü Qt frameworkün QGraphicsScene klasından gelmektedir. İlgili kontrollerin sağlanması için çeşitli sürükle bırak ve Mouse eklentileri eklendi.

```
void GraphicsView::mouseReleaseEvent(QMouseEvent *event)
{
    QGraphicsView::mouseReleaseEvent(event);

    if(event->button() == Qt::RightButton)
    {
        //rightclick -> reset image scale to 1:1
        emit rightClick();
    }
    else if(event->button() == Qt::MiddleButton)
    {
        //middle click -> fit image to view
        emit middleClick();
    }
}
```

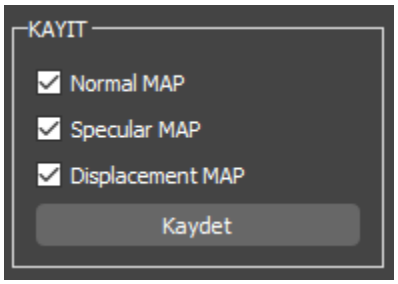
Bu kod parçasında QMouseEvent'ten aldığımız olayı mouseumuzun hangi tuşuna denk geliyorsa ona bir özellik atamış oluyoruz. Programımızın bu sürümünde orta tuş görüntüyü merkezi ortalayıp sığacak en büyük hale, sağ tıklama ise resmin kendi görüntü boyutuna getirmektedir.



GRAFİK GÖRÜNÜMÜ KONTROLLERI

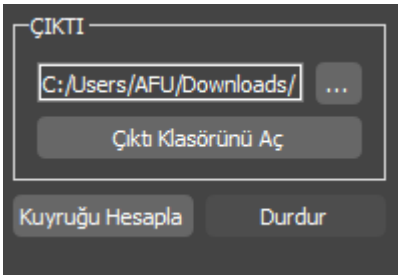
Grafik görünümünde daha ayrıntılı çalışmak için görüntüleri daha ayrıntılı incelemek isteyeceğiz.

KAYIT TÜRÜ



Kayıt edilecek harita türlerini işaretlediğimiz kısımdır. Kendisi ilgili yerlerin true işaretlenmesi durumunda kayıt butonuna bağlanır. Çıktılar ilgili harita türlerinin adlarının sonlarına ek alarak giriş klasörü olarak belirlenmiş konuma kaydedecektir.

ÇIKTI



KAYNAKLAR

SRC GUI

SRC OLUŞTURUCU