KARADENİZ TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

RUBİK KÜPÜN CÖZÜMÜ VE SİMİLASYONU

TASARIM PROJESİ

MOHAMMAD ASIF AALIMY

2015-2016 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

RUBİK KÜPÜN CÖZÜMÜ VE SİMİLASYONU

TASARIM PROJESİ

MOHAMMAD ASIF AALIMY

Bu projenin teslim edilmesi ve sunulması tarafımca uygundur.

Danışman : Yrd. Doç. Dr. Bekir Dizdaroğlu .....

2015-2016 GÜZ DÖNEMİ

ÖNSÖZ

Rubik Küpün Çözümü ve Simülasyonu adlı bu çalışma 2015-2016 Güz döneminde Yrd. Doç. Dr. Bekir Dizdaroğlu danışmanlığında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde " Tasarım Projesi " olarak yapılmıştır. Bu süre zarfında yardımını ve bilgilerini bizden eksik etmeyen tüm değerli hocalarımıza ve danışmanım Yrd. Doç. Dr. Bekir Dizdaroğlu hocamıza, maddi ve manevi anlam da bu çalışma sürecinde yanımda olan tüm arkadaşlarıma ve aileme teşekkürlerimi sunuyorum.

MOHAMMAD ASIF AALIMY

Trabzon 2015

İÇİNDEKİLER

ÖZET

Rubik Küpün Çözümü ve Simülasyonu projesi C++ ve OpenGL ile tasarlanmıştır. Proje de C++ rubik küpü çözen algoritmayı geliştirmede , OpenGL ise görsel programlamada simülasyonu yapmak için kullanılmıştır. Rubik Küpün Çözümü ve Simülasyonu projesi anahatları ile üç kısımdan meydana gelir. Bunlar; küpü çözen algoritma, algoritmanın yaptığı çözüm sonunda döndürülen hareketler dizisi ve bu hareketler dizisini görsel ortamda bir küp modeli üzerinde hareketlerin simülasyonu için kullanmadır. Program, klavyeden alınan girdilerle karıştırılan rubik küpün programın kullandığı algoritmaya göre belirlediği hareketler doğrultusunda simülasyon ortamında çözümünü gerçekleştirmektedir.

1. GENEL BİLGİLER

1.1. Giriş

Rubik Küpü 1974 yılında Ernõ Rubik adında bir Macar mimar tarafından icat edilen bir mekanik bulmacadır. Hareketli yüzeylerden oluşan ve çoğunlukla plastikten yapılmış bir küp olan Rubik Küp başlıca dört şekilde piyasaya sürülmüştür: 2×2×2'lik, 3×3×3'lük, 4×4×4'lük ve 5×5×5'lik. 6×6×6 ve 7×7×7'lik küpler de halihazırda üretilmektedir. Biz 3x3x3'lük küpleri ele alacağız.

"Rubik Küpü" diye bilinen standart 3×3×3'lük modelin her yüzünde 9 kare olmak üzere alanı toplam 54 karedir ve ortadaki görünmeyen küp hariç 26 tane küçük küplerden oluşmaktadır. Yüzeyindeki kareler genel olarak altı farklı renk ile etiketlendirilmiştir. Bulmaca çözüldüğünde küpün her yüzü tek renkten oluşur.

Yaratıcısı tarafından ilk olarak "Sihirli Küp" adı verilen bulmacaya 1980 yılında "Rubik Küpü" adı verildi ve aynı yılın mayıs ayında tüm dünyada dağıtımına başlandı. 2007 yılına kadar 300 milyon adet satıldığı ve dünyanın en çok satılan oyuncağı olduğu söylenir.

Birbirinden bağımsız olarak Rubik Küp'ün birçok çözüm yöntemi bulunmuştur. En popüler yöntem David Singmaster tarafından geliştirilmiş ve 1980 yılında Notes on Rubik's Magic Cube (Rubik'in Sihirli Küpü Üzerine Notlar) adlı kitapta yayımlanmıştır. Bu çözümde küp seviye seviye çözülüyor. Önce üst seviye, sonra orta, en sonda da alt seviye çözülüyor.

1.2. Permütasyonlar

Normal (3x3x3)'lük Rubik Küpü $(8! \times 3^{8-1}) \times (12! \times 2^{12-1})/2$ = 43.252.003.274.489.856.000 büyüklükte uzay durumuna sahiptir. Bu sayı (~4.3 × 1019) olarak da yazılabilir ve 43 kentilyon olarak okunur. Aslında Küpü oluşturan parçalar (8! × 38) × (12! × 212) = 519.024.039.293.878.272.000 (yaklaşık 519 kentilyon) kadar farklı duruma getirilebilir ama bunun yalnızca on ikide biri (1/12) ulaşılabilir durumdur. Çünkü tek bir kenarı değiştirebilecek ya da tek bir köşeyi döndürebilecek hareket sırası mümkün değildir. Bu nedenle ancak küpü söküp tekrar birleştirerek ulaşılabilir.

1.3. Rubik Küp ile Aşina Olmak

1.3.1. Rubik Küpün parçalarının tanımlaması

***Kenar Parçaları*:** İki yüzü olan parçalardır. Orta katlarda bulunurlar ve sayısı 12'dir.

ŞEKİL 1.3.1.1 Kenar Parçaları

***Koşe Parçaları*:** Üç yüzü olan parçlardır. Sekiz tanedirler ve koşelerde bulunmaktadırlar.

ŞEKİL 1.3.1.2 Köşe Parçaları

***Merkez Parçalar*:** Bir tane yüzü olan parçalardır. Sayısı 6 tane (Rubik Küpün yuzey sayısı kadar) ve Rubik Küpün her yüzeyinin ortasında birer tane bulunmaktadır. Merkez parçalar çok önemli bilgi taşımaktadırlar. *Merkez parçalar birbirlerine göre sabit ve her birisi bulunmakta olduğu yüzeyin rengini belirler.*

ŞEKİL 1.3.1.3 Merkez Parçaları

1.3.2. Küpün Her Tarafı (Yüzeyi) Bir Harf İle Temsil Edilir:

R = Right Layer (Sağ Katman)

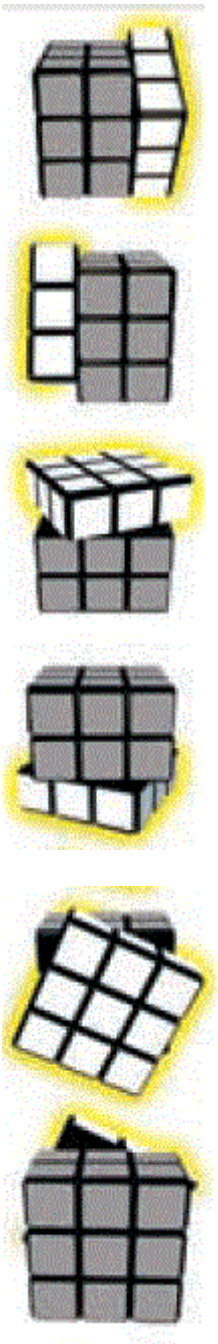
L = Left Layer (Sol Katman)

U = Up Layer (Üst Katman)

D = Down Layer (Alt Katman)

F = Front Layer (Ön Katman)

B = Back Layer (Arka Katman)

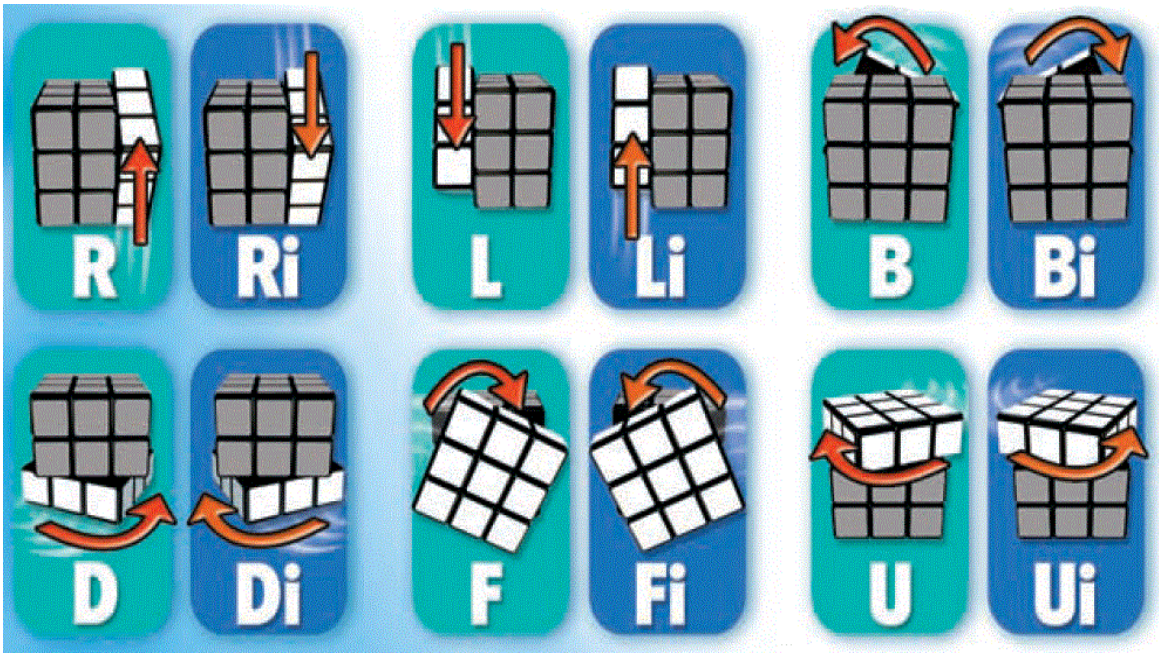


ŞEKİL 1.3.1.4 Katmanların Gösterimi

1.3.3. Katmanların Döndürme Gösterimi:

Bir katmanın saat yönüne döndürmesini o katmanın sembölü ile, saat yönünün tersine döndürmesini o katmanın sembölüne ‘i’ harfini ekleyerk gösteririz.

Not: saat yönünü belirlemek için döndürmek istediğimiz yüzeyin bize karşı olduğunu farz ederiz.



ŞEKİL 1.3.3.1. Katmanların Döndürme Gösterimi

1.3.4. Çözüm Aşamaları

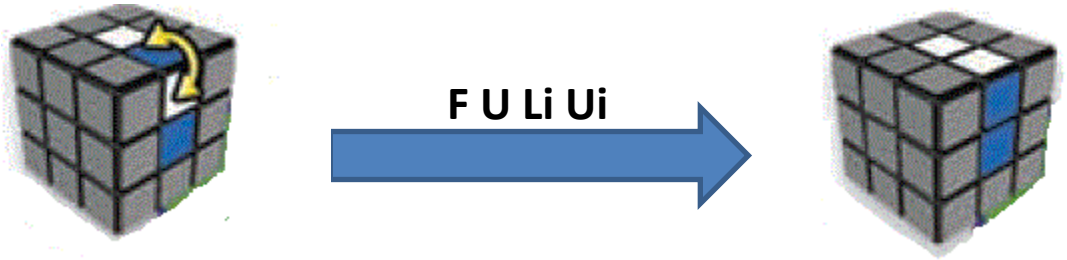
Birbirinden bağımsız olarak Rubik Küpünün birçok çözüm yöntemi bulunmuştur. En popüler yöntem David Singmaster tarafından geliştirilmiş ve 1980 yılında Notes on Rubik's Magic Cube (Rubik'in Sihirli Küpü Üzerine Notlar) adlı kitapta yayımlanmıştır. Bu çözümde küp seviye seviye çözülüyor. Önce üst seviye, sonra orta, en sonra da alt seviye çözülüyor. Biz burada seviye seviye yöntemini kullanacağız.

1. Aşama, Üst Katta Arti İşareti Oluşturulması

Küpün her hangı bir yüzünü üst yüz olarak seçerek üst katmanda bir *artı işareti* oluşturulur.

ŞEKİL 1.3.4.1. Üst Katmanda Artı İşareti Oluşturma

Arti işaretini oluşturmaya çalışırken bazen aşağıdaki resimde gösterilen duruma karşılaşılabilir yani üst ktatta bir kenar parçanın yeri doğru ama sadece ters dururuyorsa, o zaman ilgili parça ön katmanda olacak şekilde küpü ele alıp aşağıdaki hareketleriyapacağız.

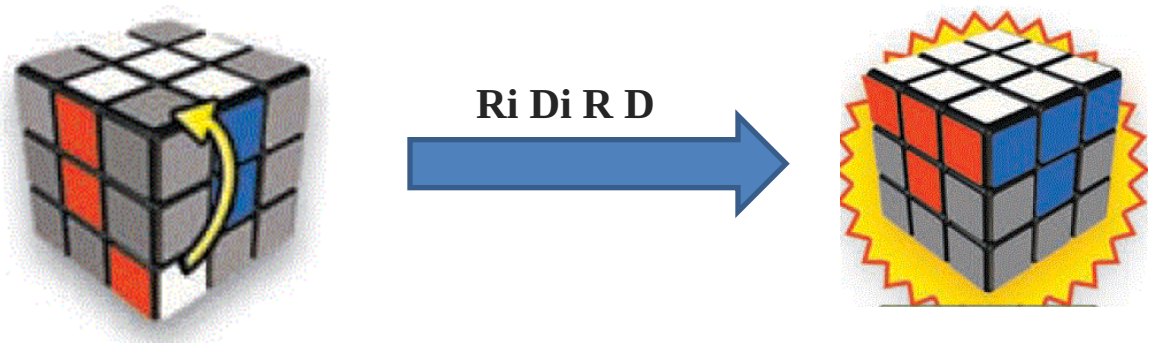


ŞEKİL 1.3.4.2. Üst Katmanda Artı İşareti Oluşturma

2. Aşama, Köşeleri Çözmek

Bir koşe parçanın yerini tespit etmek için o koşe parçanın yüzeylerinin renklerine bakarız. Her koşe parçası küpün 3 yüzey kesişiminde bulunmaktadır. Koşe parcanın yüzeylerinin renkleri de bu yüzeyleri belirler.

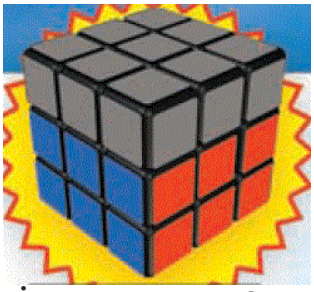
Artı işareti oluşturulduktan sonra üst katmanda geriye kalan dört köşeyi tamamlamak için uygun köşeler halledilecek köşenin tam aşağısına getirilerek resimde ki hareketler uygulanır. Bu hareketler uygulandıktan sonra eğer ilgili köşe yerine gelmemişse bu hareketler ilgili köşe çözülenene kadar tekrar edilir. Her dört köşe için bu aşamalar tekrar edilir.



ŞEKİL 1.3.4.3. Köşe Parçaları Cözmek

3. Aşama: Orta Katmanı Çözmek

Tamamlanan katman alttayken, şimdi orta katmanı çözmeye çalışacağız, yani orta katmandaki kenar parçaları yerlerine yerleştirmek.

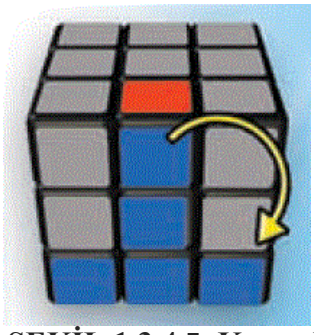


ŞEKİL 1.3.4.4. Hedef Durum

İpucu:

- Şimdi üst yüzey sarı olacak (bunu üsteki merkez parçasının renginden anlarız) dolayısıyla üstte bulunan ve sarı içermeyen kenar paçalar orta katmana aittir anlamına gelir.
- Mavi sütuna dikkat edelim (bu kırmızı, turuncu veya yeşil olabilir)
- Üst katmanı çevirerek sarı içermeyen kenar parça ile bir sütün oluşturulur. Kenar parçanın üst yüzeyinin rengi o parçanın orta katamanın hangı tarafına ait olduğunu gösterir.

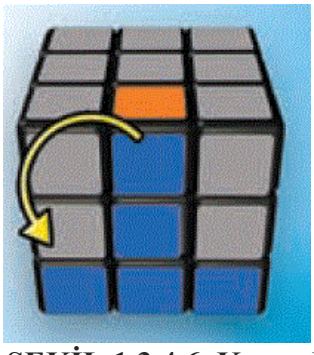
1. Eğer kenar parça orta katmanın sağına ait ise aşağıdaki hareket dizisi yapılır.



U R Ui Ri Ui Fi U F

ŞEKİL 1.3.4.5. Kenar Parça Orta Katmanın Sağına Ait Olduğu Durum

2. Eğer kenar parça orta katmanın soluna ait ise aşağıdaki hareket dizisi yapılır.



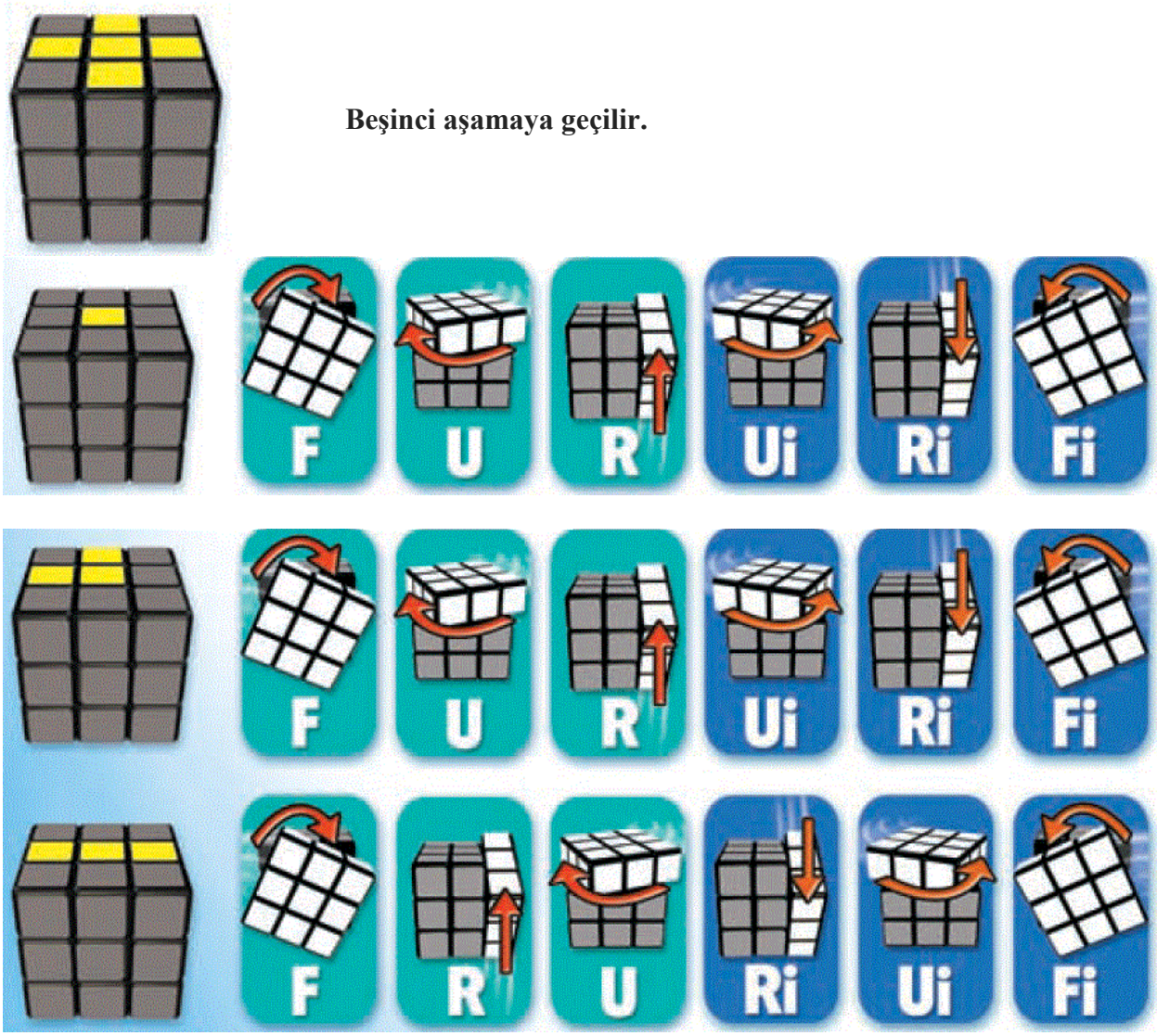
Ui Li U L U F Ui Fi

ŞEKİL 1.3.4.6. Kenar Parça Orta Katmanın Soluna Ait Olduğu Durum

4. Aşama, Son Katmanda Artı İşareti Oluşturulması

ŞEKİL 1.3.4.7. Dördüncü Aşamanın Hadefi

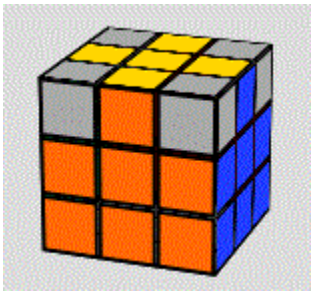
Üst kattaki sarı deseni aşağıdakilerin birisi ile uyuşturp ilgili hareket dizisi gerçeleştirilir.



Beşinci aşamaya geçilir.

ŞEKİL 1.3.4.8. Dördüncü Aşamanın Hareketleri

5. Aşama, Üst Kattaki Kenar Parçaların Etraftaki Merkez Parçaları İle Uyuşturulması



ŞEKİL 1.3.4.9. Beşinci Hadef Durumu

1. Eğer iki kesişen yüzeylerdeki kenar paçalar düzeltilmiş durumda ise, o zaman düzeltilmiş parçalar birisi sağ tarafta diğeri arka tarafta olacak şekilde küpü ele alıp aşağıdaki hareketler yapılır.

R U Ri U R U U Ri

2. Eğer iki paralel yüzeylerdeki kenar paçalar düzeltilmiş durumda ise, o zaman düzeltilmiş parçalar birisi bize doğru ve diğeri arka tarafta olacak şekilde küpü ele alıp aşağıdaki hareketler yapılır.

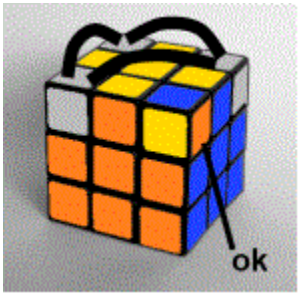
R U Ri U R U U Ri

6. Aşama, Üst Kattaki Köşe Parçalar Uygun Yerlerine Yerleştirilmesi

Doğru yerinde olmayan köşe parça ön katmanın sağ-üst koşesindeyken aşağideki hareketleri gerçekleştirilir.

U R Ui Li U Ri Ui L

Örneğin aşağıdaki şekilde doğru yerinde olan bir köşe parçası gösterilmiş.

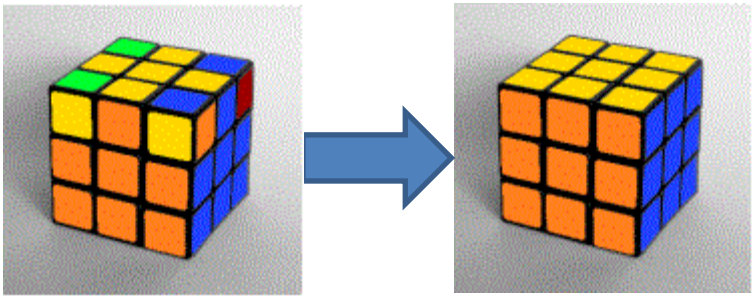


ŞEKİL 1.3.4.10. Doğru Yerinde Olan Bir Köşe Parçasının Gösterilişi

7. Aşama, Üstteki Köşe Parçaların Düzeltilmesi

Bu aşamada, aşamanın başında küpü nasıl elimize aldıysak, küp tamamen çözülüne kadar küpü o şekilde tutmaya devam ederiz. Yerinde düzgün durmayan köşe parçası sağ-üst köşedeyken aşağıdaki hareketler ilgili parça düzeltilene kadar tekrarlanır. İlgili parça yerine geldikten sonra üst katmanı çevirerek düzeltilmemiş başka bir koşe parçayı ön katmanın sağ-üst köşesine getirerek ayni hareketleri tekrarlarız.

(Ri Di R D)x2



ŞEKİL 1.3.4.11. Üstteki Köşe Parçaların Düzeltilmesi

2. YAPILAN ÇALIŞMALAR

Yapılan tasarım projesi çalışmasında Rubik Küpün çözümü ve simulasyonu gerçekleştirilmeye çalışılmıştır. Tasarımı iki kısım olarak görebiliriz.

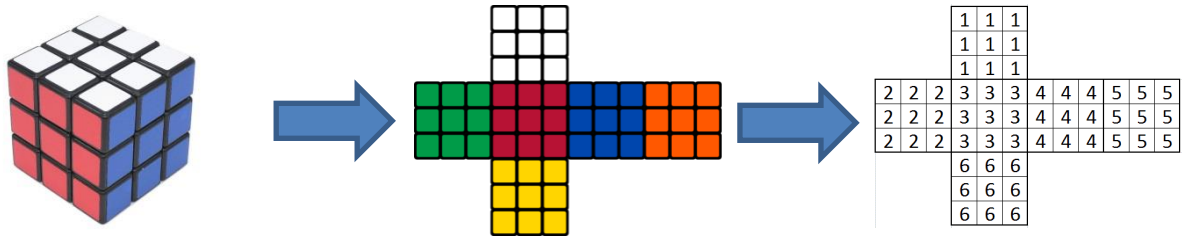
1. kısım: küpü kod ortamında bir class ile modellemek.

2. kısım: küpü "GENEL BİLGİLER" bölümünde anlatılan yöntem ile çözen fonksiyonları yazmak ve uygulanan hareketleri bir dizide saklamak.

3. kısım: küpü grafiksel olarak modellemek ve 2. kısımda dizide saklanan komutları bu küp üzerine uygulayarak küpü çözmek.

2.1. Küpü Kod Ortamında Bir Class ile Modellemek.

Küpü kod ortamında modellemek için her yüzü iki boyutlu diziler ile temsil edilir.



ŞEKİL 2.1.1. Küpün Matrisler ile Temsil Edilmesi

Küpü temsil eden bir sınıf (Kup) tanımlanır.

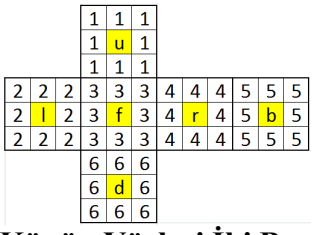
```
class Kup
{
public:
	int f[3][3];
	int l[3][3];
	int b[3][3];
	int r[3][3];
	int u[3][3];
	int d[3][3];
	void sag(int a[3][3]);
	void sol(int a[3][3]);
	void intial(int b[3][3], int val);
```

```
public:
	Kup()
	{
		intial(f, 1);
		intial(l, 2);
		intial(b, 3);
		intial(r, 4);
		intial(u, 5);
		intial(d, 6);
	}
	void R();
	void L();
	void U();
	void D();
	void F();
	void B();
	void Ri();
	void Li();
	void Ui();
	void Di();
	void Fi();
	void Bi();
	void print();
};
```

Kod Parçası 2.1.1. Kup Sınfı

Kup sınfının field'leri: küpün her yüzü iki boyutlu *int* dizisi ile temsil edilir. Dizi elemanları renk bilgisini taşırlar.

Aşağıda küpün her yüzü ve o yüzü temsil eden dizinin ismi gösterilmiştir.



ŞEKİL 2.1.2. Küpün Yüzleri İki Boyutlu Diziler ile Temsil edilmesi

Örneğin kübün üst yüzü aşağıdaki iki boyutlu dizi ile temsil edilir.

```
int u[3][3] = { 1, 1, 1,
                1, 1, 1,
                1, 1, 1 };
```

Kup sınfının metodları: küp yüzeylerin çevirme işlemlerini yapan 12 tane metod var. Bu metodlar aşağıda listelenmiştir.

```
void R();
void L();
void U();
void D();
void F();
void B();
```

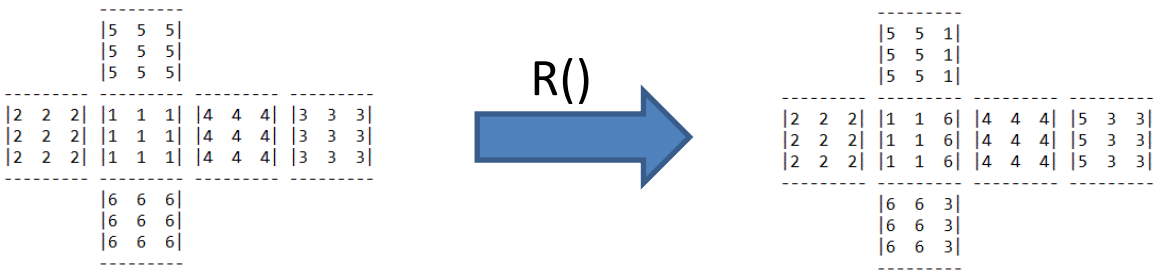
```
void Ri();
void Li();
void Ui();
void Di();
void Fi();
void Bi();
```

Örneğin R() fonksiyonu küpün sağ katmanını saat yönünde 90 derece döndürme işlemini modellemek için küp yüzeylerini temsil eden diziler üzerinde aşağıdaki işlemleri yapar.

```
void R()
{
	int up[3] = { u[0][2], u[1][2], u[2][2] };
	u[0][2] = f[0][2]; u[1][2] = f[1][2]; u[2][2] = f[2][2];
	f[0][2] = d[0][2]; f[1][2] = d[1][2]; f[2][2] = d[2][2];
	d[0][2] = b[2][0]; d[1][2] = b[1][0]; d[2][2] = b[0][0];
	b[0][0] = up[2]; b[1][0] = up[1]; b[2][0] = up[0];

	sag(r);
}

void sag(int ar[3][3])
{
	int temp = ar[0][0];
	ar[0][0] = ar[2][0]; ar[2][0] = ar[2][2];
	ar[2][2] = ar[0][2];
	ar[0][2] = temp; temp = ar[0][1];
	ar[0][1] = ar[1][0]; ar[1][0] = ar[2][1];
	ar[2][1] = ar[1][2];
	ar[1][2] = temp;
}
```

Kod Parçası 2.1.2. Küpün Sağ Katmanını Döndüren Fonsksiyon

Aşağıda R() fonksiyonunun küpün yüzlerini temsil eden dizilerin üzerindeki etkisi gösterilmiştir.



ŞEKİL 2.1.2. Küpün Yüzleri İki Boyutlu Diziler ile Temsil edilmesi

2.2. kısım, Küpü Çözen Fonksiyonların Tanımlanması

Bu kısımda Küpü "GENEL BİLGİLER" Bölümünde Anlatılan Yöntem ile Çözen Fonksiyonları tanımlamaya Çalışılır.

Artık gerçek dünyadaki bir küpün bütün özelliklerini ve davranışlarını taklit eden bir Kup sınfımız var. Şimdi 'Çözüm Aşamaları' bölümünde anlatılan aşamaları Kup nesnesi üzerinde gerçekleştiren başka bir sınıf (Cozen) tanımlamaya calışacağız.

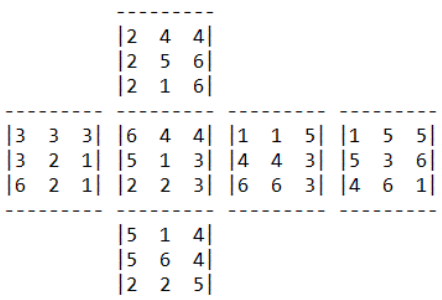
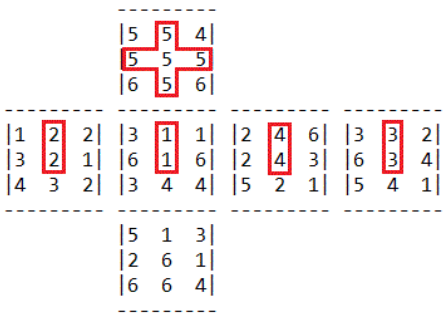
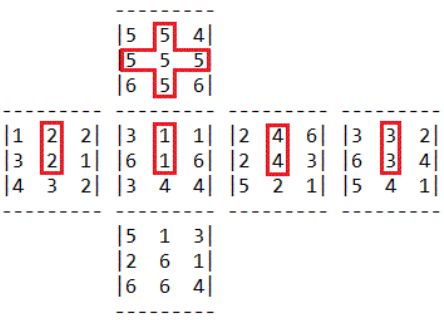
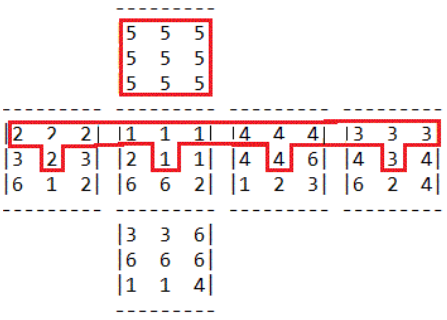
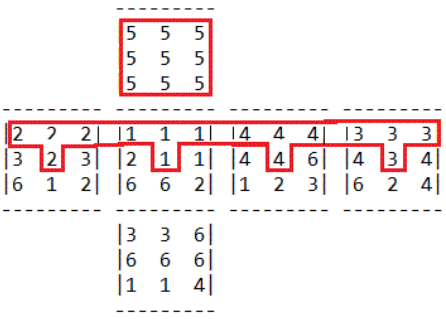
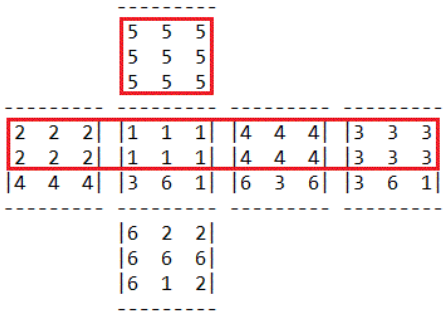
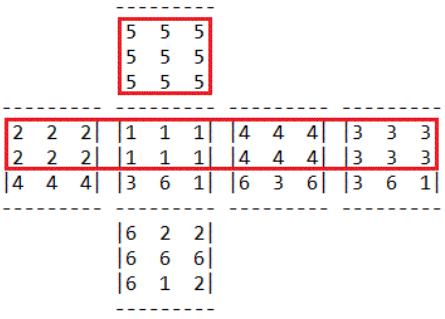
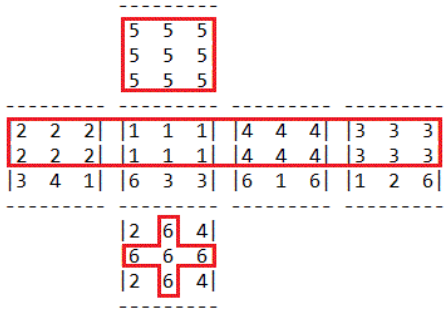
```
class Cozen
{
private:
	Kup k;
	string Moves;
	string moves[400];
	int moveSize;
	void Arti();
	void koseler1();
	void ortaKat();
	void Arti2();
	void Bacak();
	void Kose2();
	void son();
	void tamam();
public:
	Cozen();
	void kupuCoz(Kup k, string* &mvs, int &mvsSize);
};
```

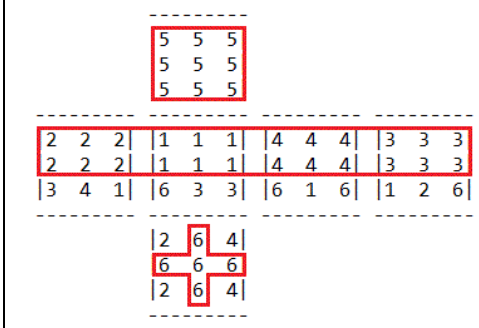
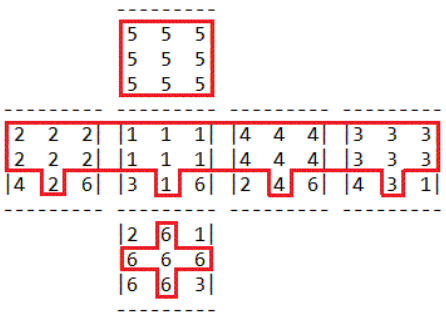
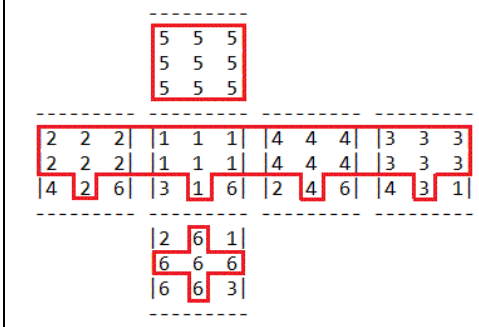
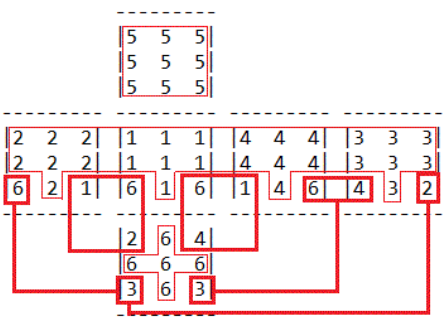
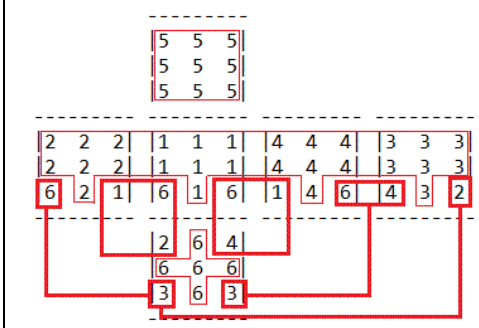
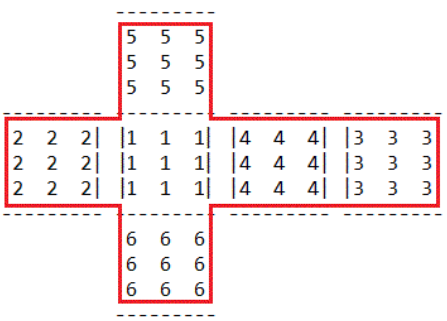
Kod Parçası 2.1.3. Küpü Çözen Class

Cozen'in field'leri:

`k:`	bir Kup nesnesi
`Moves:`	k küpü üzerinde yapılan (U F B Fi ... gibi) her hareket bu string'de saklanır.
`moves:`	(U F B Fi ... gibi) her komuta kolayca ulaşabilmek için `Moves` string'i string dizisine çevirilip bu dizide saklanır.
`moveSize:`	`moves` dizisinin içerdiği eleman sayısını tutar.

Cozen'in private metodları:

Metodun Adı	Aldığı Küp	Verdiği Küp
Arti()		
koseler1()		
ortaKat()		
Arti2()		

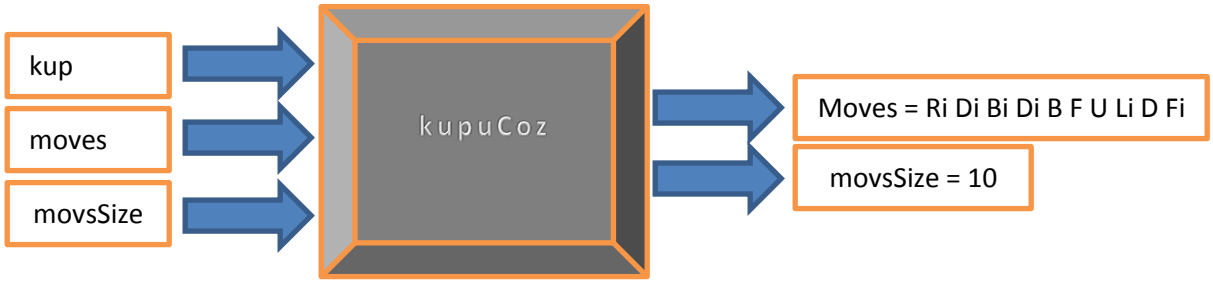
`Bacak()`		
`Kose2()`		
`son()`		

TABLO 2.1.1. Cozen Class'ın Privat Metodların İşleyişleri

Cozen'in public metodu:

```
void kupuCoz(Kup kup, string* &moves, int &movesSize);
```

Cozen sınfının (kupuCoz adında) sadece bir tane public metodu var. Bu metod 3 tane parametresi var. kupuCoz fonksiyonu birinci parametre olarak bir Kup nesnesi alıyor. Fonksiyon bu küpü çözdükten sonra, küp üzerinde yaptığı bütün hareketleri komut olarak ikinci parametre olarak aldığı diziye atıyor ve dizinin içerdiği komut sayısını de üçüncü parametresine atıyor. Bu şekilde Cozen'i kullanan program bu komut dizisini kullanarak küpü çözer.



ŞEKİL 2.1.3. kupuCoz Fonksiyonunun Şematiği

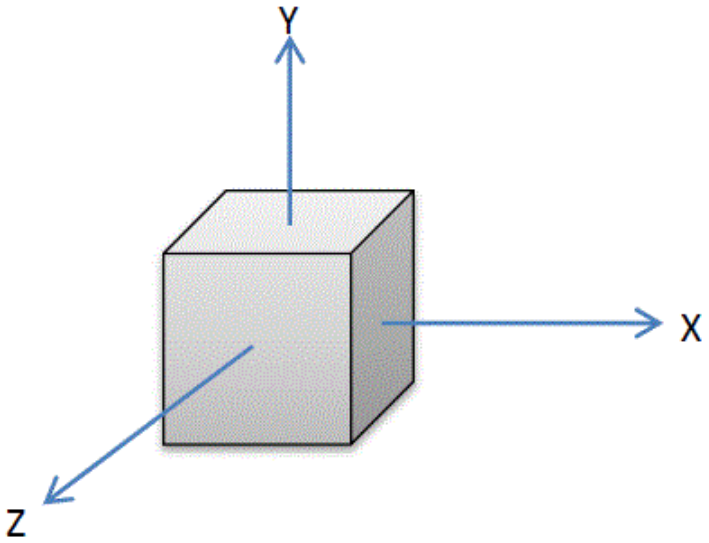
2.3. Kısım, Küpün Grafiksel Olarak Modellenmesi ve Çözülmesi

Bu bölümde OpenGL dili kullanılarak küpü grafiksel olarak modellemek ve 2. kısımda dizide saklanan komutları bu küp üzerine uygulayarak küpün çözülmesi amaçlanır. Bu amaç için önce küpün (merkezindeki görünmeyen) tek bir parçacasını çiziyoruz. Bu küçük küpün kodu aşağıda verilmiştir.

```
//alt
glColor3f(0, 0, 0);
glVertex3f(0.5, -0.5, 0.5);
glVertex3f(0.5, -0.5, -0.5);
glVertex3f(-0.5, -0.5, -0.5);
glVertex3f(-0.5, -0.5, 0.5);
// arka
glVertex3f(-0.5, -0.5, -0.5);
glVertex3f(-0.5, 0.5, -0.5);
glVertex3f(0.5, 0.5, -0.5);
glVertex3f(0.5, -0.5, -0.5);
// sol
glVertex3f(-0.5, -0.5, -0.5);
glVertex3f(-0.5, 0.5, -0.5);
glVertex3f(-0.5, 0.5, 0.5);
glVertex3f(-0.5, -0.5, 0.5);
// on
glVertex3f(-0.5, -0.5, 0.5);
glVertex3f(-0.5, 0.5, 0.5);
glVertex3f(0.5, 0.5, 0.5);
glVertex3f(0.5, -0.5, 0.5);
//sag
glVertex3f(0.5, -0.5, 0.5);
glVertex3f(0.5, 0.5, 0.5);
glVertex3f(0.5, 0.5, -0.5);
glVertex3f(0.5, -0.5, -0.5);
//ust
glVertex3f(0.5, 0.5, 0.5);
glVertex3f(0.5, 0.5, -0.5);
glVertex3f(-0.5, 0.5, -0.5);
glVertex3f(-0.5, 0.5, 0.5);
```

Kod Parçası 2.1.4. Merkezdeki Küçük Küpün Kodu

Yukarıdaki kodun sonucunda aşağıda gösterildiği gibi dünya koordinat sisteminin merkezinde küçük bir küp yaratılır.



ŞEKİL 2.1.4. Rubik Küpün Merkezindeki Küçük Küp

Şimdi bu küpü referans alarak Rubik Küpün diğer bütün küplerininin köşe koordinatlarını hesaplayabiliriz. Örneğin bu küpün hemen sağ tarafındaki küpü üretmek için bu küpün bütün koordinatlarının x bileşenlerine 1 birim eklemek yeterli olacak.

Dolayısıyla her küçük küp sadece bir (x, y, z) üçlüsü ile belirlenebilir. Örneğin referans küpün hemen sağındaki küp (1, 0, 0) üçlüsü ile belirlenir.

Küplerin yüzlerini boyamak: Küplerin yüzlerini boyamak için boyamak istediğimiz yüzlerine bir kare yapıştırılır. Bu karelerin köşe koordinatları da boyamak istediğimiz yüzün köşe koordinatlarından elde edilir.

Bir küpün konumuna göre bir, iki veya üç yüzü boyalı olabilir. Dolayısıyla her küpün renk bilgisini tutmak için 3 tane RGB üçlüsü tutmak lazım. Ayrıca hatırlayalım ki Kup sınfı her rengi bir tam sayı (1, 2, 3, 4, 5, 6) ile temsil eder. Dolayısıyla her RGB üçlüsüne bir `int` bileşen daha eklemek lazım. Şimdi çözülmesi gereken bir sorun daha var. Küpün hangi yüzü veya yüzleri boyanmalı?

Bir küpün ayni anda iki paralel yüzü boyalı olamaz. Örneğin bir küpün ayni anda hem sağ hem de sol yüzü boyalı olamaz. Dolyısı ile küpün sağ veya sol tarafları, alt veya üst tarafları, ön veya arka tarafları sırası ile yuz1, yuz2 ve yuz3 char değişkenleri ile belirtilir; bu yüzler sırasıyla renk1, renk2 ve renk3 renkleri ile boyanır.

Örneğin eğer `yuz1= 'r'` ve `renk1 = ( 1, 0, 0 )` ise, küpün sağ yüzünün kırmızı ile boyanmasını belirler.

Şimdi bir küpü modellemek için hangi bilgilerin tutulması gerektiğini biliyoruz, bir küpü temsil eden aşağıdaki sınfı tanımlıyoruz.

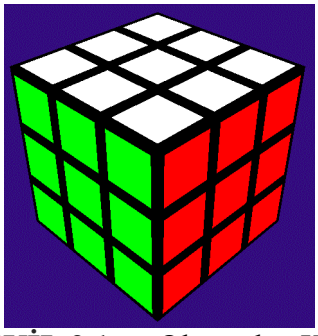
```
typedef struct
{
	float r;
	float g;
	float b;
	int rnk;
}Renk;

class KUP
{
public:
	float x;
	float y;
	float z;
	Renk renk1;
	Renk renk2;
	Renk renk3;
	char yuz1 = ' ';
	char yuz2 = ' ';
	char yuz3 = ' ';
};
```

Kod Parçası 2.1.5. Rubik Küpü Oluşturan Küçük Küplerin Oluşturulduğu Class

Rübik Küpümüz bu küçük küplerin 27 tanesinden oluştuğundan aşağıdaki 3x3x3 üç boyutlu KUP dizisini tanımlarız.

```
KUP kup[3][3][3];
```

Bu dizinin elemanlarını uygun değerlerle doldurduktan sonra, dizi kullanılarak Rübik Küpü aşağıdaki gibi oluşturulur.



ŞEKİL 2.1.5. Oluşturlan Küp

Rubik Küpünün döndürma işlemlerini modellemek için de aşağıdaki OpenGl fonksiyonunu kullanarak Rubik Küpünün belli katmanını belli yöne çevirebiliriz.

```
void glRotated(GLdouble angle, GLdouble x, GLdouble y, GLdouble z);
```

şimdi küpü grafiksel olarak modelledik. Grafiksel Küpü karıştırdıktan sonra kupuOku fonksiyonu grafiksel küpe göre Rubik Küpün yüzlerini temsil eden bir Kup nesnesinin iki boyutlu `int` dizilerini dolduruyor. Sonra bu Kup nesnesi bir Cozen nesnesinin `kupuCoz` fonksiyonuna parametre olarak verilir. Bu fonksiyon kupu çözmek için bir komut dizisi bize verir. Ondan sonra bu komut dizisi kullanılarak grafiksel Küp modeli çözülür.

3. Programın Ara Yüzü

Programı çalıstırdıktan sonra, Rubik Küpünün çözülmüş halı ekranda gösterilir. Küpü karıştırmak için klavyeden b, f, d, l, r, u, B, F, D, L, R, U karakterleri girilir. Sonra ‘c’ karakterini bastıktan sonra program Küpü çözmeye başlar.

Klavyeden Girilen Karakterlerin Sonuçları

b:	Küpün arka katmanını saat yönüne çevirir.
B	Küpün arka katmanını saat yönün tersine çevirir.
f:	Küpün ön katmanını saat yönüne çevirir.
F	Küpün ön katmanını saat yönün tersine çevirir.
d:	Küpün alt katmanını saat yönüne çevirir.
D:	Küpün alt katmanını saat yönün tersine çevirir.
l:	Küpün sol katmanını saat yönüne çevirir.
L:	Küpün sol katmanını saat yönün tersine çevirir.
r:	Küpün sağ katmanını saat yönüne çevirir.
R:	Küpün sağ katmanını saat yönün tersine çevirir.
u:	Küpün üst katmanını saat yönüne çevirir.
U:	Küpün üst katmanını saat yönün tersine çevirir.
c:	Küpü çözmeye başlar.

ÖNERİLER

Bu programın, Rubik Küpü çözmek için ürettiği komut dizisi Arduino'ya gönderilerek gerçek bir Rubik Küp çözülebilir.