ERCİYES ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
PROGRAMMING LABORATORY NOTLARI

12. HAFTA NOTU

(JAVA İLE BASİT OYUN TASARIMI)

Bu not dersin 12. haftasında yapacağınız uygulamaya hazırlanmanızı sağlayacak olup temel kavramları anlatan bilgileri içermektedir.

DERSİN ÖĞRETİM ÜYESİ

Prof. Dr. Bahriye AKAY

DENEYİ YAPTIRAN ÖĞRETİM ELEMANI

Arş. Gör. Mert Korkut Çoban

2026
KAYSERİ

1. JAVA İLE OYUN TASARIMI HAKKINDA TEMEL BİLGİLER

1.1.Oyun Tasarımı Nedir?

Oyun tasarımı, kullanıcının belirli kurallar çerçevesinde etkileşime girdiği, görsel ve mantıksal öğelerden oluşan yazılımların planlanması ve geliştirilmesi sürecidir. Bir oyunda genellikle oyuncu, hedef, kurallar, puanlama sistemi, engeller ve kazanma/kaybetme durumları bulunur.

Basit bir bilgisayar oyununda kullanıcı klavye, fare veya ekrandaki butonlar aracılığıyla oyuna müdahale eder. Program ise bu girdilere göre ekrandaki nesnelerin konumunu değiştirir, çarpışmaları kontrol eder, puanı günceller ve oyunun devam edip etmeyeceğine karar verir.

Bu deneyde amaç, karmaşık bir oyun motoru kullanmak değil; Java programlama dili ve JavaFX arayüz kütüphanesi ile temel oyun mantığını öğrenmektir.

1.2.Java ile Oyun Geliştirme

Java, nesne yönelimli programlama yaklaşımını destekleyen, farklı işletim sistemlerinde çalışabilen ve grafik arayüz uygulamaları geliştirmek için çeşitli kütüphaneler sunan bir programlama dilidir.

Java ile oyun geliştirirken oyun içerisindeki varlıklar genellikle nesneler olarak düşünülür. Örneğin bir oyuncu karakteri, bir engel, bir hedef nesne veya skor bilgisi program içerisinde ayrı değişkenler ya da sınıflar ile temsil edilebilir.

Basit bir oyun uygulamasında aşağıdaki temel işlemler bulunur:

· Oyun penceresinin oluşturulması,
· Oyuncu veya oyun nesnelerinin ekrana çizilmesi,
· Klavye veya fare girdilerinin alınması,
· Nesnelerin konumlarının güncellenmesi,
· Çarpışma veya temas durumlarının kontrol edilmesi,
· Skor veya can gibi bilgilerin güncellenmesi,
· Oyunun kazanma veya kaybetme durumunun belirlenmesi.

Bu deney kapsamında öğrenciler, JavaFX kullanarak görsel arayüzü olan basit bir oyun uygulamasının nasıl oluşturulduğunu görecektir.

1.3.Oyun Döngüsü Mantığı

Oyunların temel çalışma prensiplerinden biri oyun döngüsüdür. Oyun döngüsü, oyun çalıştığı sürece sürekli tekrar eden işlemler bütünüdür. Bu döngü içerisinde genellikle kullanıcı girdileri alınır, oyun durumu güncellenir ve ekrandaki görüntü yenilenir.

Basit bir oyun döngüsü şu şekilde düşünülebilir:

1. Kullanıcıdan giriş alınır.
2. Oyuncunun veya nesnelerin konumu güncellenir.
3. Çarpışma veya hedefe ulaşma durumu kontrol edilir.
4. Skor, süre veya can bilgisi güncellenir.
5. Ekran yeniden çizilir.

JavaFX uygulamalarında bu mantık, belirli zaman aralıklarında çalışan yapılar veya olay tabanlı fonksiyonlar ile gerçekleştirilebilir. Örneğin klavyeden bir tuşa basıldığında oyuncu nesnesinin konumu değiştirilebilir.

1.4.Olay Tabanlı Programlama

JavaFX ile geliştirilen grafik arayüz uygulamalarında olay tabanlı programlama yaklaşımı kullanılır. Bu yaklaşımda program, kullanıcının yaptığı işlemlere göre tepki verir.

Örneğin:

· Bir butona tıklanması,
· Klavyeden bir tuşa basılması,
· Farenin hareket ettirilmesi,
· Pencerenin kapatılması

birer olaydır.

Oyun tasarımında olay tabanlı programlama oldukça önemlidir. Çünkü oyuncunun klavye veya fare ile yaptığı işlemler oyunun akışını doğrudan etkiler. Örneğin kullanıcı sağ ok tuşuna bastığında karakter sağa, sol ok tuşuna bastığında karakter sola hareket edebilir.

1.5. Koordinat Sistemi ve Nesne Konumları

Grafik tabanlı oyunlarda ekrandaki nesnelerin yerleri koordinat sistemi ile belirlenir. Yatay eksen genellikle x, dikey eksen ise y koordinatı ile ifade edilir.

Ekranın sol üst köşesi çoğunlukla başlangıç noktasıdır. x değeri sağa doğru arttıkça nesne sağa hareket eder. y değeri aşağı doğru arttıkça nesne aşağı hareket eder.

Örneğin:

· x değerinin artırılması nesneyi sağa taşır,
· x değerinin azaltılması nesneyi sola taşır,
· y değerinin artırılması nesneyi aşağı taşır,
· y değerinin azaltılması nesneyi yukarı taşır.

Bu nedenle oyun geliştirme sürecinde nesnelerin koordinatlarının doğru güncellenmesi gerekir.

1.6. Çarpışma Kontrolü

Çarpışma kontrolü, oyun içerisindeki iki nesnenin birbirine temas edip etmediğini belirleme işlemidir. Basit oyunlarda oyuncunun bir hedefe ulaşması, bir engele çarpması veya bir nesneyi toplaması çarpışma kontrolü ile anlaşılır.

JavaFX içerisinde görsel nesnelerin sınır bilgileri kullanılarak çarpışma kontrolü yapılabilir. Örneğin oyuncu karakterini temsil eden bir şekil ile hedef nesneyi temsil eden başka bir şeklin sınırları kesişiyorsa, bu durumda çarpışma meydana gelmiş kabul edilebilir.

Çarpışma kontrolü sonucunda:

· Skor artırılabilir,
· Oyun sonlandırılabilir,
· Oyuncunun canı azaltılabilir,
· Yeni hedef nesnesi oluşturulabilir,
· Ekrana bilgilendirme mesajı yazdırılabilir.

1.7. Skor ve Oyun Durumu

Bir oyunda skor, oyuncunun başarısını temsil eden sayısal bir değerdir. Skor genellikle hedef toplandığında, doğru işlem yapıldığında veya belirli bir görevin tamamlanmasıyla artırılır.

Oyun durumu ise oyunun devam edip etmediğini, kazanılıp kazanılmadığını veya kaybedilip kaybedilmediğini ifade eder. Basit bir oyunda oyun durumu şu şekilde takip edilebilir:

· Oyun başladı,
· Oyun devam ediyor,
· Oyun kazanıldı,
· Oyun kaybedildi.

Bu durumlar değişkenler yardımıyla kontrol edilebilir. Örneğin oyuncu hedef nesneye ulaştığında skor artırılır, belirli bir skora ulaşıldığında oyun kazanıldı mesajı gösterilir.

2. DENEYDE KULLANILACAK KÜTÜPHANELER VE FONKSİYONLAR HAKKINDA KISA BİLGİ

2.1. JavaFX

JavaFX, Java ile masaüstü tabanlı grafik arayüz uygulamaları geliştirmek için kullanılan bir kütüphanedir. Pencere, buton, metin, şekil, resim ve olay yönetimi gibi birçok bileşen sunar.

Bu deneyde JavaFX, oyun penceresini oluşturmak, oyun nesnelerini ekranda göstermek ve klavye/fare etkileşimlerini almak için kullanılacaktır.

JavaFX ile bir uygulama geliştirilirken genellikle Application sınıfı genişletilir ve start() metodu içerisinde uygulamanın ana penceresi hazırlanır.

2.2. Application Sınıfı

Application sınıfı, JavaFX uygulamalarının temel sınıfıdır. JavaFX ile geliştirilen bir program bu sınıftan türetilir.

Temel yapı şu şekildedir:

```
public class Main extends Application {
@Override
public void start(Stage stage) {
// Uygulama içeriği burada oluşturulur.
}}
```

Burada start() metodu, JavaFX uygulaması başladığında çalışan ana metottur. Oyun penceresi, sahne ve ekrandaki nesneler genellikle bu metot içinde hazırlanır.

2.3.Stage

Stage, JavaFX uygulamasında ana pencereyi temsil eder. Bir oyun uygulamasında kullanıcının gördüğü pencere Stage nesnesi üzerinden oluşturulur.

Stage üzerinde pencere başlığı ayarlanabilir, sahne eklenebilir ve pencere ekranda gösterilebilir.

Örnek kullanım:

```
stage.setTitle("Basit Oyun");
stage.setScene(scene);
stage.show();
```

Bu yapı ile pencereye başlık verilir, oluşturulan sahne pencereye eklenir ve pencere görünür hâle getirilir.

2.4.Scene

Scene, JavaFX uygulamasında pencerenin içeriğini temsil eder. Bir başka ifadeyle Stage pencerenin kendisi, Scene ise pencerenin içinde gösterilecek alandır.

Oyun nesneleri, butonlar, yazılar ve şekiller Scene içerisinde yer alır. Scene aynı zamanda klavye olaylarını yakalamak için de kullanılabilir.

Örnek kullanım:

```
Scene scene = new Scene(root, 600, 400);
```

Bu örnekte 600 piksel genişliğinde ve 400 piksel yüksekliğinde bir sahne oluşturulmaktadır.

2.5.Pane

Pane, JavaFX içerisinde görsel bileşenleri üzerinde tutan bir yerleşim alanıdır. Oyun geliştirme uygulamalarında şekiller, metinler ve diğer nesneler Pane üzerine eklenebilir.

Örnek kullanım:

```
Pane root = new Pane();
```

Daha sonra oyun nesneleri şu şekilde Pane içerisine eklenebilir:

```
root.getChildren().add(player);
```

Bu yapı, player isimli nesneyi ekranda görünür hâle getirir.

2.6.Shape Sınıfları

JavaFX içerisinde Rectangle, Circle ve Line gibi şekil sınıfları bulunur. Basit oyun tasarımlarında oyuncu, engel veya hedef nesneleri bu şekillerle temsil edilebilir.

```
Rectangle player = new Rectangle(40, 40);
Circle target = new Circle(15);
```

Bu örnekte oyuncu dikdörtgen, hedef ise daire olarak oluşturulmuştur. Şekillerin konumu setX(), setY(), setCenterX() ve setCenterY() gibi metotlarla ayarlanabilir.

2.7.Text

Text sınıfı, ekrana yazı yazdırmak için kullanılır. Oyunlarda skor, süre, bilgilendirme mesajı veya oyun sonu mesajı Text nesnesi ile gösterilebilir.

Örnek kullanım:

```
Text scoreText = new Text("Skor: 0");
```

Skor değiştiğinde metin şu şekilde güncellenebilir:

```
scoreText.setText("Skor: " + score);
```

2.8.Klavye Olayları

JavaFX uygulamalarında klavye olayları setOnKeyPressed() metodu ile yakalanabilir. Bu yapı, kullanıcının hangi tuşa bastığını kontrol etmek için kullanılır.

Örnek kullanım:

```
scene.setOnKeyPressed(event -> {
   switch (event.getCode()) {
      case RIGHT:
         player.setX(player.getX() + 10);
         break;
      case LEFT:
         player.setX(player.getX() - 10);
         break;
   }
});
```

Bu örnekte sağ ok tuşuna basıldığında oyuncu sağa, sol ok tuşuna basıldığında oyuncu sola hareket eder.

2.9.Bounds ve Çarpışma Kontrolü

JavaFX içerisinde nesnelerin ekrandaki sınırlarını almak için getBoundsInParent() metodu kullanılabilir. İki nesnenin sınırlarının kesişip kesişmediği intersects() metodu ile kontrol edilebilir.

Örnek kullanım:

```
if
(player.getBoundsInParent().intersects(target.getB
oundsInParent())) {
   score++;
}
```

Bu yapı, oyuncu nesnesi ile hedef nesnesi temas ettiğinde skorun artırılmasını sağlar.

2.10. ArrayList

ArrayList, Java'da aynı türden birden fazla veriyi dinamik olarak saklamak için kullanılan bir liste yapısıdır. Dizilerden farklı olarak boyutu program çalışırken artırılabilir veya azaltılabilir. Bu nedenle oyunlarda birden fazla nesnenin saklanması, eklenmesi, silinmesi veya güncellenmesi gereken durumlarda kullanılabilir.

Örnek kullanım:

```
ArrayList<Rectangle> objects = new ArrayList<>();
objects.add(newObject);
objects.remove(objects.size() - 1);
```

Bu örnekte objects listesi, Rectangle türündeki oyun nesnelerini saklamak için kullanılır. add() metodu listeye yeni eleman eklerken, remove() metodu listeden eleman silmek için kullanılır.

2.11. Random

Random sınıfı, rastgele sayı üretmek için kullanılır. Oyunlarda hedef nesnenin rastgele bir konumda oluşturulması, engellerin farklı yerlerde görünmesi veya oyun akışının değişken hâle getirilmesi için kullanılabilir.

Örnek kullanım:

```
Random random = new Random();
int x = random.nextInt(500);
int y = random.nextInt(300);
```

Bu örnekte x ve y değişkenleri için rastgele konum değerleri üretilmektedir.

2.12. Timeline

Timeline, JavaFX içerisinde belirli zaman aralıklarında tekrar eden işlemler yapmak için kullanılabilir. Basit oyunlarda zamanlayıcı, hareketli nesne veya periyodik kontrol işlemleri için tercih edilebilir.

Örneğin bir nesnenin belirli aralıklarla hareket ettirilmesi veya oyundaki sürenin azaltılması Timeline ile yapılabilir.

3. DERSİN ÖĞRENME ÇIKTILARI

Bu deney sonunda öğrencilerin aşağıdaki kazanımları elde etmesi beklenmektedir:

- Java ile grafik arayüzlü basit bir oyun uygulamasının temel yapısını açıklayabilmek,
- JavaFX kütüphanesinin oyun tasarımındaki kullanım amacını kavrayabilmek,
- Stage, Scene ve Pane kavramlarını açıklayabilmek,
- JavaFX içerisinde Rectangle, Circle ve Text gibi temel görsel bileşenleri kullanabilmek,
- Klavye olaylarını yakalayarak kullanıcı girdisine göre işlem yapabilmek,
- Koordinat sistemi mantığını kullanarak oyun nesnelerinin konumunu değiştirebilmek,
- Basit çarpışma kontrolü yapabilmek,
- Skor bilgisini değişkenler yardımıyla takip edebilmek ve ekranda gösterebilmek,
- Random sınıfı ile oyun nesnelerini rastgele konumlandırabilmek,
- Timeline veya olay tabanlı yapılar ile oyun akışını kontrol edebilmek,
- Java programlama dilinde nesne yönelimli düşünme yaklaşımını basit oyun tasarımı üzerinde uygulayabilmek,
- Geliştirilen oyun uygulamasının çalışma mantığını teknik olarak yorumlayabilmek.