KARE PUZZLE OYUNU

TBL331: Yazılım Geliştirme Labaratuvarı II

2022-2023 Bahar

Proje I

Binnur Özcan
191307059
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği

github.com/binnurozcan/puzzleGame

***Özet*—Bu proje, bir resmin 16 parçaya ayrılmasını ve bu parçaların kullanıcının müdahalesiyle yer değiştirilerek tekrar birleştirilmesini amaçlayan bir web uygulamasıdır. Kullanıcı, projeye bir resim yükleyerek oyunu başlatır. Yüklenen resim, 4x4 (toplam 16) parçaya bölünür ve bu parçalar karıştırılır. Kullanıcının amacı, parçaları yer değiştirerek resmi tamamlamaktır. Kullanıcı, fare tıklaması kullanarak, iki parçanın yerini değiştirebilir. Proje, parçaların yer değiştirilmesi sırasında kullanıcının hamle sayısını takip eder ve en sonunda tamamlanan resmin hamle sayısı ile birlikte kullanıcıya gösterilmesini sağlar. Ayrıca, kullanıcının oyunu bitirmesine olanak tanıyan bir düğme de vardır. Oyunun sonunda, kullanıcının adı ve hamle sayısı bir iletişim kutusu aracılığıyla gösterilir.**

Anahtar Kelimeler— Puzzle, JavaScipt, HTML5, Canvas, Linked-List, Fisher-Yates algoritması

I. Giriş

Bu proje, JavaScript ile oluşturulmuş bir "Puzzle" oyunu uygulamasıdır. Kullanıcı bir resim seçer ve bu resim 16 parçaya bölünür. Daha sonra kullanıcının bu parçaları karıştırması ve orijinal resimdeki düzgün sıralamayı elde etmesi gerekmektedir. Projenin kullanıcı arayüzü, bir HTML dosyası ile oluşturulmuştur.

II. Kod Açıklamaları

Projenin yapısı, iki sınıftan oluşmaktadır: Node ve LinkedList. Node sınıfı, bağlı listedeki düğümleri temsil etmek için kullanılırken, LinkedList sınıfı, Node nesnelerinin bağlı listedeki işlemlerini yönetmek için kullanılır. Ayrıca, resimlerin karıştırılması için bir shuffle özelliği ve hamle sayısının takibi için bir moveCount özelliği de tanımlanmıştır.
Projede, kullanıcının resim seçmesi için bir dosya girişi elemanı, karıştırma düğmesi ve oyunu bitirme düğmesi bulunmaktadır. Kullanıcının resim seçmesi, FileReader ve Canvas nesneleri kullanılarak gerçekleştirilir. Resim, 16 parçaya bölünür ve bu parçalar bağlı listede depolanır.

A. Node Sınıfı

Node sınıfı list'in düğümlerinin tanımlanması için kullanılır.

```
class Node {
    constructor(data) {
        this.data = data;
        this.next = null;
    }
}
```

Her düğüm, listede saklanacak veri (data) ve bir sonraki düğümün referansı (next) içerir. next, başlangıçta null olarak ayarlanır, çünkü yeni bir düğüm oluşturulduğunda henüz bir sonraki düğüm yoktur. Data, düğümde saklanacak herhangi bir veri türü olabilir (örneğin sayı, dize, nesne vb.).

B. LinkedList Sınıfı

LinkedList sınıfı, bağlı liste veri yapısını temsil eden bir JavaScript sınıfıdır. Bu sınıf, içinde düğümler (Node) barındıran bir bağlı liste oluşturur ve bu düğümler üzerinde bazı işlemler yapmamızı sağlar.

```
class LinkedList {
    constructor() {
        this.head = null;
        this.tail = null;
        this.selected = null;
        this.moveCount = 0;
        this.shuffled = false;

    }
```

head: Bağlı listenin baş düğümünü tutar.
tail: Bağlı listenin son düğümünü tutar.
selected: Kullanıcının seçtiği fotoğrafı tutar.
moveCount: Oyun sırasında yapılan hamle sayısını tutar.
shuffled: Fotoğrafların karıştırılıp karıştırılmadığını tutar.

LinkedList Sınıfının metodları

1. addPhoto()

addPhoto() metodu, verilen fotoğrafı bağlı listeye ekliyor. Fotoğraf, bir düğüm olarak bağlı listede depolanıyor. İlk olarak, fotoğraf için bir yeni düğüm oluşturulur. Daha sonra, bağlantılı listenin boş olup olmadığı kontrol edilir. Eğer boş ise, yeni düğüm hem baş hem de son düğüm olur. Aksi

takdirde, son düğümün next özelliği yeni düğümle değiştirilir ve son düğüm, yeni düğüm olarak güncellenir.

```
addPhoto(photo) {
    const node = new Node(photo);
    if (this.head === null) {
        this.head = node;
        this.tail = node;
    } else {
        this.tail.next = node;
        this.tail = node;
    }
}
```

2. displayPhotos()

displayPhotos() metodu, bağlı listedeki fotoğrafları HTML tablosu şeklinde gösteriyor.

```
displayPhotos(containerId) {
    let current = this.head;
    const container = document.getElementById(containerId);

    const table = document.createElement("table");
    let row = document.createElement("tr");
    let count = 0;

    while (current !== null) {
        const img = document.createElement("img");
        img.src = current.data;

        const cell = document.createElement("td");
        cell.appendChild(img);
        row.appendChild(cell);
        count++;

        if (count % 4 === 0) {
            table.appendChild(row);
            row = document.createElement("tr");
        }

        current = current.next;
```

Bu metot, her fotoğrafın bir <img> etiketi oluşturduktan sonra bu etiketleri <td> etiketleri içine ekliyor ve sonunda tüm <td> etiketlerini bir <tr> etiketi içine koyarak bir HTML tablosu oluşturuyor.
Metod ayrıca her fotoğraf için bir tıklama olayı dinleyicisi ekler, böylece kullanıcılar bir fotoğrafın yerini diğer bir fotoğraf ile değiştirebilirler.

```
// Fotoğrafın tıklanma olayı eklendi
img.addEventListener("click", () => {

    if (this.selected === null) {
        this.selected = img;
    } else {
        const selectedImg = this.selected;
        const temp = selectedImg.src;
        selectedImg.src = img.src;
        img.src = temp;
        this.selected = null;
        this.moveCount++; // hamle sayısı artırılıyor
        const moveCountElement = document.getElementById("move-count");
        moveCountElement.innerHTML = this.moveCount.toString();
    }

});
```

Bu metod ayrıca moveCount adında bir sayacı da günceller. Bu sayacın, kullanıcının bir fotoğrafın yerini değiştirdiği her hamlede bir artması gerekiyor. Sayacın güncellenmiş değeri HTML sayfasındaki ilgili etikete yazdırılır.

```
container.innerHTML = "";
container.appendChild(table);
```

Tablo öğesi containerId'deki HTML öğesine eklenir.

3. shufflePhotos() metodu

shufflePhotos() metodu, bağlı listedeki fotoğrafları karıştırıyor.

```
shufflePhotos() {
    let current = this.head;
    let arr = [];


    // Copying photo links to an array
    while (current !== null) {
        arr.push(current.data);
        current = current.next;
    }

    // Shuffling the array
    for (let i = arr.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [arr[i], arr[j]] = [arr[j], arr[i]];
    }

    // Updating nodes with shuffled links
    current = this.head;
    let i = 0;
    while (current !== null) {
        current.data = arr[i];
        current = current.next;
        i++;
    }

    this.shuffled = true; // shuffled özelliği true olarak ayarlanıyor

}
```

Bağlı listedeki parçaların karıştırılması, Fisher-Yates algoritması kullanılarak gerçekleştirilir.
Bu metot, fotoğrafların bağlı listedeki sırasını karıştırmak yerine, fotoğrafların linklerini bir diziye kopyalıyor, bu diziyi karıştırıyor ve daha sonra bağlı listedeki fotoğraf düğümlerinin verilerini karıştırılmış linklerle güncelliyor. Bu yöntem, bağlı listedeki düğümleri yeniden düzenlemek yerine sadece düğümlerin verilerini güncelleyerek daha hızlı çalışır.

C. Dosya yükleme ve Oyunu bitirme

Oyuncunun dosya yüklemesi yapacağı alanda (file input) bir değişiklik olduğunda çalışacak bir olay dinleyicisi (event listener) bulunur. Seçilen resim dosyasını yüklemek için bir dosya okuyucu (FileReader) oluşturulur ve resmin yüklenmesi tamamlandığında, resim dosyasından 16 parçalı bir bulmaca oluşturmak için canvas kullanarak gerekli işlemler yapılır. Son olarak, oluşturulan fotoğraf parçaları, oluşturulan bağlı listedeki düğümlere eklenecek ve ekranda görüntülenmesi için displayPhotos() metodu çağrılacaktır. Ayrıca, "Tamam" düğmesine tıklanıldığında, oyuncunun adı ve hamle sayısı bir ileti kutusu (alert) ile görüntülenecektir.

```
const photoList = new LinkedList();
const piecesCount = 16;
const fileInput = document.getElementById("file-input");
fileInput.addEventListener("change", function() {
    const playerName = window.prompt("Lütfen isminizi girin:");
    const file = fileInput.files[0];
    const reader = new FileReader();
    reader.readAsDataURL(file);

    reader.onload = function(event) {
        const img = new Image();
        img.src = event.target.result;
        img.onload = function() {
            const canvas = document.createElement("canvas");
            const ctx = canvas.getContext("2d");
            const pieceWidth = img.width / 4;
            const pieceHeight = img.height / 4;
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img, 0, 0);

            for (let i = 0; i < 4; i++) {
                for (let j = 0; j < 4; j++) {
                    const nodeCanvas = document.createElement("canvas");
                    nodeCanvas.width = pieceWidth;
                    nodeCanvas.height = pieceHeight;
                    const nodeCtx = nodeCanvas.getContext("2d");
                    nodeCtx.drawImage(canvas, j * pieceWidth, i * pieceHeight,
                        pieceWidth, pieceHeight, 0, 0, pieceWidth, pieceHeight);
                    const nodeDataUrl = nodeCanvas.toDataURL();
                    photoList.addPhoto(nodeDataUrl);
                }
            }

            photoList.displayPhotos("photo-container");
        };
```

Kodun ana adımları şunlardır:
Yeni bir LinkedList örneği oluşturulur.
Resmin kaç parçaya bölüneceği belirlenir (piecesCount).
Dosya giriş alanı (file input) dinlenir ve dosya yükleme işlemi tamamlandığında çalışacak bir işlev atanır.
İsim alanına kullanıcının adını girmesi için bir pencere (prompt) gösterilir (playerName).
Dosya okuyucu (FileReader) kullanarak resim dosyasını yükler (reader.readAsDataURL(file)).
Resim, img nesnesine yüklenir (img.src = event.target.result) ve yükleme işlemi tamamlandığında çalışacak bir işlev atanır (img.onload).
Resmin parçaları için bir canvas öğesi oluşturulur (canvas) ve her bir parça için bir alt canvas öğesi oluşturulur (nodeCanvas).
Canvaslar kullanılarak resmin parçaları ayrılır ve ayrı canvas öğelerine yerleştirilir (nodeCtx.drawImage()).
Her bir canvas öğesi, addPhoto() yöntemi kullanılarak photoList adlı bir linked list'e eklenir (photoList.addPhoto(nodeDataUrl)).
Son olarak, displayPhotos() yöntemi kullanılarak resim parçaları HTML belgesinde belirtilen bir öğeye (photo-container) yerleştirilir.
Oyunun bitirilmesi için bir düğme (button) oluşturulur (endBtn) ve tıklama işlemi dinlenir. Düğmeye tıklandığında, kullanıcının adı ve hamle sayısı bir uyarı kutusunda (alert()) gösterilir.

```
const endBtn = document.getElementById("end-btn");
endBtn.addEventListener("click", function() {
    const moveCount = photoList.moveCount;
    alert("Kullanıcı adı: "+playerName +"\n" +"Hamle sayısı: "+moveCount);


});
```

D. Skor Tablosu

```
const data = [
    { playerName: "Ahmet", score: 120 },
    { playerName: "Mehmet", score: 90 },
    { playerName: "Ayşe", score: 200 },
    { playerName: "Fatma", score: 150 }
  ];

  // Skora göre sıralama
  data.sort((a, b) => b.score - a.score);

  // Tablo satırlarını oluşturma
  const tableRows = data.map(item => {
    return `
      <tr>
        <td>${item.playerName}</td>
        <td>${item.score}</td>
      </tr>
    `;
  });

  // Tablo satırlarını HTML içine ekleyerek gösterme
  const playerList = document.getElementById("player-list");
  playerList.innerHTML = tableRows.join("");
```

İlk olarak, bir data dizisi tanımlanır. Bu dizi, her biri bir oyuncu adı ve puanı içeren nesneler içerir.Daha sonra, data dizisi, sort yöntemi kullanılarak skora göre ters sıralanır.Sonra, map yöntemi kullanılarak her bir öğe için bir tablo satırı oluşturulur. Bu satır, <tr> etiketleri arasında, iki sütunda oluşur: birincisi oyuncu adını, ikincisi ise skoru gösterir.Son olarak, tablo satırları HTML içine yerleştirilir ve gösterilir. Bunun için, getElementById yöntemi kullanılarak player-list ID'li bir HTML öğesi seçilir ve innerHTML özelliği kullanılarak tablo satırları içindeki HTML kodu yerleştirilir. join() yöntemi kullanılarak tablo satırları arasına boşluk karakteri eklenir.

E. HTML

Sayfa içindeki bileşenler şu şekildedir:

- Fontawesome kitinin CSS dosyası, sayfada ikon kullanımını sağlamak için kullanılıyor.
- meta etiketi ile sayfanın karakter kodlaması belirtiliyor.
- Sayfa başlığı title etiketi ile belirtiliyor.
- style etiketi içinde sayfanın stil özellikleri tanımlanıyor.
- input etiketi ile gizli girdi elemanı oluşturuluyor.
- .container sınıfı ile birlikte, sayfanın ortalanmış halde ve en fazla 600 piksel genişliğinde görüntülenmesi sağlanıyor.
- .top-banner sınıfı ile sayfa üst kısmında koyu yeşil bir arka plan üzerine beyaz renkli başlık yerleştiriliyor.
- .top-banner2 sınıfı ile sayfanın ikinci bölümünde açık yeşil bir arka plan üzerine beyaz renkli düğmeler yerleştiriliyor.
- .score-banner sınıfı ile sayfanın alt kısmında turkuaz rengi bir arka plan üzerine yeşil renkli skor listesi başlığı yerleştiriliyor.

- .scoreboard sınıfı ile skor tablosu oluşturuluyor. Tablo, 2 sütundan oluşuyor: biri oyuncu adı ve diğeri skoru içeriyor.
- .count sınıfı ile sayfanın ortasında siyah renkli bir alan üzerinde oyuncunun hamle sayısı görüntüleniyor.
- .btn-group .button sınıfları ile birlikte düğmeler oluşturuluyor. Düğmeler, yeşil bir arka plan üzerine beyaz renkli yazı ve ikonlar içeriyor
- .script etiketleri ile sayfada kullanılacak olan JavaScript dosyaları çağırılıyor.