KARE PUZZLE OYUNU

İsmail Kocatürk, Zeynep Gül Aslan

Bilişim Sistemleri Mühendisliği Teknoloji
Fakültesi/Kocaeli Üniversitesi
201307045@kocaeli.edu.tr, 191307049@kocaeli.edu.tr

***Özet*—Bu ödev kapsamında, bir puzzle çözme oyunu tasarımı gerçekleştirilmesi beklenmektedir. Proje, kullanıcı arayüzü (GUI) hazırlanarak, masaüstü uygulaması veya web tabanlı olarak geliştirilebilir. Puzzle parça eşleştirmeleri için kullanılacak yöntemin bağlı liste veri yapısı olması zorunludur.**

Projenin isterleri arasında, puzzle görselinin 16 parçadan oluşması, Github platformunun kullanımı, herhangi bir dosya veya URL'den resim/fotoğraf dosyasının okunması, parçaların karıştırılması, doğru yere yerleştirildiğinde parçaların kilitlenmesi, en yüksek skor bilgisinin arayüzde yer alması, doğru hamlelerin +5 puan, yanlış hamlelerin -10 puan olarak tasarlanması, puan durumunun arayüz üzerinden görüntülenebilmesi, en yüksek puanların azalan sırada gösterilmesi gibi özellikler yer almaktadır.

Projenin tamamlanması için bir isim istenmekte ve bu isim, skor metin belgesine kaydedilmektedir. Ayrıca, oyun boyunca kullanıcının hamle sayısı da kaydedilmekte ve en yüksek skorlar, her açılışta azalan sırada gösterilmektedir..

Anahtar kelimeler—GUI, Bağlı Liste, Puzzle

***Abstract*— Abstract—Within this assignment, it is expected to design a puzzle solving game. The project can be developed as a desktop application or web-based by preparing the user interface (GUI). The method to be used for puzzle piece matching must be a linked list data structure.**

Among the requirements of the project, the puzzle image consists of 16 pieces, the use of the Github platform, the reading of the image / photo file from any file or URL, the mixing of the pieces, the locking of the pieces when placed in the right place, the highest score information on the interface, the correct moves +5 points, There are features such as designing wrong moves as -10 points, displaying the score status on the interface, displaying the highest scores in descending order.

A name is requested for the completion of the project and this name is recorded in the score text document. In addition, the user's number of moves throughout the game is also recorded, and the highest scores are displayed in descending order at each launch.

Keywords— GUI, Linked List, Puzzle

1. GIRIŞ

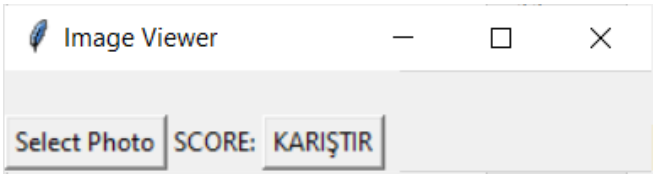
Puzzle çözme oyunları, akıl zekasını geliştirmek ve eğlenceli vakit geçirmek için popüler bir oyun türüdür. Bu ödev kapsamında, bir puzzle çözme oyunu tasarımı gerçekleştirilmiştir. Oyun, kullanıcı arayüzü hazırlanarak, masaüstü uygulamasına çevrilmiştir.

Projenin temel özellikleri arasında, 16 parçadan oluşan bir puzzle görselinin kullanılması, Github platformunun zorunlu olarak kullanılması, resim/fotoğraf dosyasının okunması, parçaların karıştırılması, doğru yerleştirilen parçaların kilitlenmesi, en yüksek skor bilgisinin arayüzde yer alması, doğru hamlelerin +5 puan, yanlış hamlelerin -10 puan olarak tasarlanması, puan durumunun arayüz üzerinden görüntülenebilmesi, en yüksek puanların azalan sırada gösterilmesi yer almaktadır.

Ödevin tamamlanması için, kullanıcının ismi ve hamle sayısı kaydedilmekte ve en yüksek skorlar her açılışta azalan sırada gösterilmektedir. Bu ödev kapsamında, puzzle parça eşleştirmeleri için bağlı liste veri yapısının kullanılması zorunludur.

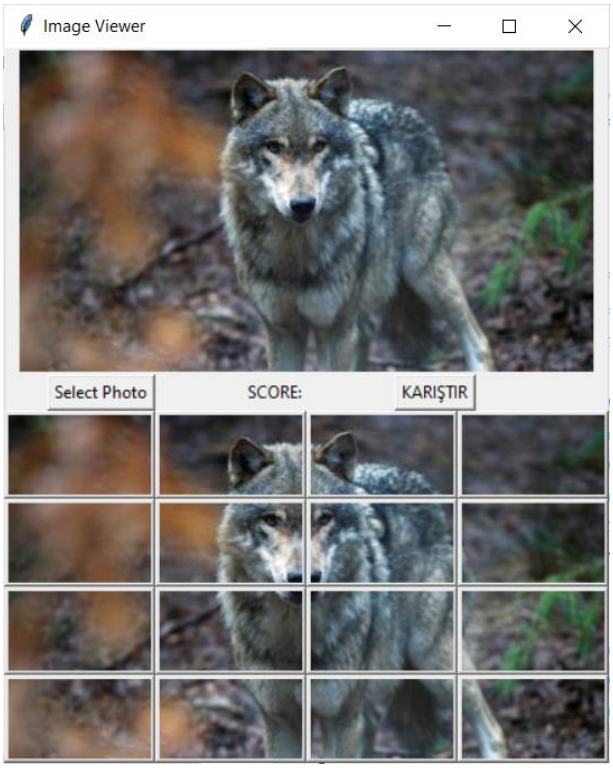
1.1 Kullanıcıya Fotoğraf Seçtirme ve Fotoğrafı Butonlar üstünde görüntüleme

Bu uygulama, kullanıcının bir fotoğraf seçmesine izin verir. Seçilen fotoğraf daha sonra 16 parçaya bölünür. Her bir parça, bir butonla temsil edilir ve kullanıcı butonları tıklayarak fotoğrafın farklı bölümlerini görüntüleyebilir.



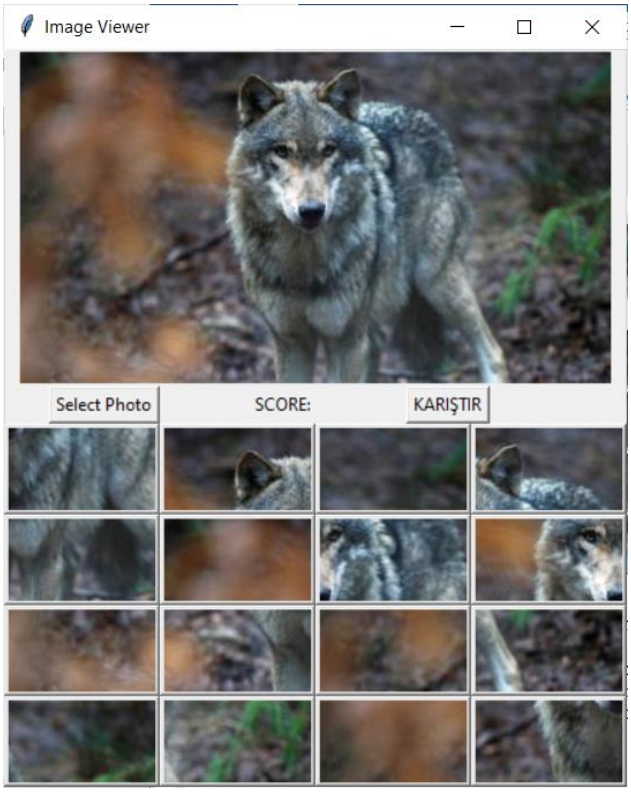
Şekil-1 Kullanıcının fotoğraf seçtiği yer.

Uygulamada kullanılan `ImageViewer` sınıfı, uygulamanın ana yapısını oluşturur. Bu sınıf, `__init__`, `select_photo` ve `display_photo` adlı üç yöntem içerir. `__init__` yöntemi, uygulamanın GUI yapısını oluşturur. `select_photo` yöntemi, kullanıcının fotoğraf seçmesine olanak tanır ve `display_photo` yöntemi, seçilen fotoğrafı ekranda gösterir.



Şekil-2 Fotoğrafın butonlar ile görüntülendiği yer.

Ayrıca, seçilen fotoğrafın 16 parçaya bölünmesi için `display_photo` yöntemi, `PIL` kütüphanesi içindeki `Image` sınıfını kullanır. Bu sınıf, seçilen fotoğrafın boyutlarını ve her bir parçanın boyutlarını hesaplar. Daha sonra, her bir parça, `crop` yöntemi ile kesilir ve her bir butona yerleştirilir.

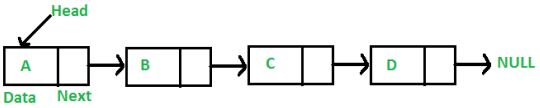


Şekil-3 Karıştır butonu.

1.2 Puzzle Parçalarını Eşleştirmek İçin Bağlı Liste Kullanımı

Puzzle parçaları, bağlı listelerdeki düğümler olarak temsil edilebilir. Her düğüm, parça hakkındaki bilgileri, örneğin şekli, rengi ve konumu gibi, içeren verileri içerir. Bir düğümden diğerine olan bağlantı, parçaların eşleştirilmesini temsil eder.

Bağlı liste kullanarak puzzle parçalarını eşleştirmek için, her bir puzzle için bir bağlı liste oluşturarak başlayabiliriz. Bağlı listenin her düğümü, parçayı temsil eder ve veri alanları, parça hakkındaki bilgileri içerir. Daha sonra, bağlı listeleri dolaşarak, her düğümün veri alanlarını karşılaştırarak eşleşmeleri bulabiliriz.

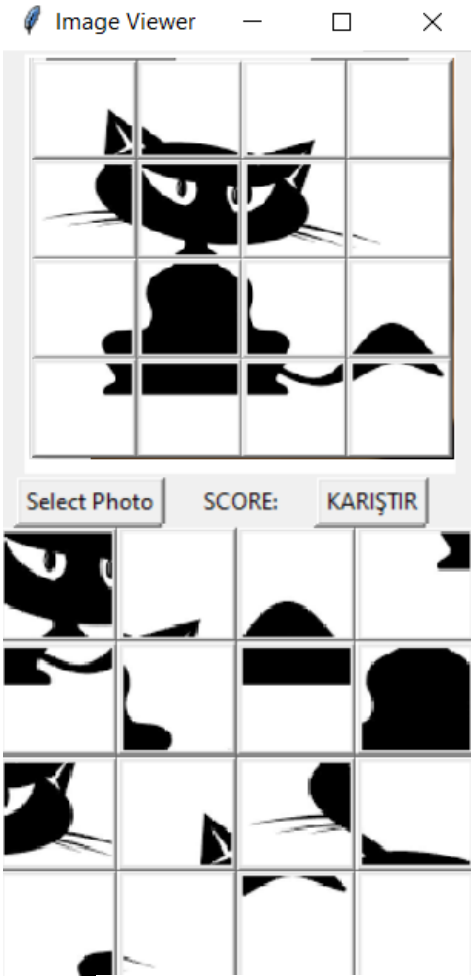


Şekil-4 Bağlı Liste.

Bir eşleşme bulunduğunda, düğümler arasındaki bağlantıları güncelleyerek parçaların eşleştirilmesini temsil edebiliriz. Örneğin, A düğümü B düğümüyle eşleşirse, A düğümündeki bağlantıyı B düğümüne, B düğümündeki bağlantıyı da A düğümüne yönlendirebiliriz. Bu, puzzle parçalarının eşleştirilmesini temsil eden bağlantılı düğümler zinciri oluşturur.

2. GUI TASARIMI

GUI (Graphical User Interface), Grafiksel Kullanıcı Arayüzü, elektronik cihazların simgeler, ikonlar ve diğer görsel grafikler yardımıyla kullanmasına yardımcı olması amacıyla geliştirilmiş tasarımlardır. GUI öncesi komut bazlı işletim sistemi kullanılan eski nesil bilgisayarlar ve elektronik cihazlarda herhangi bir işlemi gerçekleştirmek için komut satırı kullanılıyordu. Kullanıcılar bilgisayar üzerindeki her işlemi klavye üzerinden ve komutları kullanarak gerçekleştiriyordu.



Şekil-13 GUI Tasarımı.

3. BENZER ÇALIŞMALAR VE LITARATÜR TARAMASI

Puzzle çözme oyunları, yazılım geliştirme alanında sıklıkla kullanılan ve popüler olan bir oyun türüdür. Oyunların temel amacı, oyuncuların verilen parçaları doğru bir şekilde birleştirerek tam bir resim oluşturmalarını sağlamaktır.

Puzzle çözme oyunlarının, yazılım geliştirme sürecinde problem çözme ve mantıksal düşünme becerilerinin geliştirilmesinde önemli bir araç olduğu bilinmektedir. Bu oyunlar, yazılım geliştiricilerin kodlama problemlerine alternatif çözümler düşünmelerine ve daha etkili bir şekilde kod yazmalarına yardımcı olabilir. Bu ödevde de olduğu gibi bu tür oyunlar, yazılım geliştiricilerin algoritmaların ve veri yapılarının nasıl çalıştığını daha iyi anlamalarını sağlayabilir.

4. PROJEYİ GELİŞTİRİRKEN KULLANDIĞIMIZ YAZILIMSAL MİMARİ YÖNTEM VE TEKNİKLER

Projeyi geliştirirken Geleneksel Süreç Modellerindan Arttırımsal (Augmentative) Model'i kullandık.

Arttırımsal Modelde üretilen ve uygulamaya alınan her ürün sürümü birbirini içeren ve giderek artan sayıda işlev içerecek biçimde geliştirilmektedir. Öncelikle ürüne ilişkin çekirdek bir kısım geliştirilerek uygulamaya alınmakta, ardından yeni işlevsellikler eklenerek yeni sürümler elde edilmektedir. Tıpkı bu ödevi yaptığımız hafta boyunca her gün gelişimini izlememiz gibi.

Uygulama, kullanıcının bir fotoğraf seçmesine, fotoğrafı 16 parçaya bölmesine, bu parçaları karıştırmasına ve parçaları doğru sıraya yerleştirmesine olanak tanır. Ayrıca uygulama, kullanıcının puanını tutar ve doğru veya yanlış hamleler için kullanıcıya geri bildirim sağlar. Kodların ayrıntılı açıklaması şöyledir:

tkinter: Python'da grafiksel kullanıcı arayüzü (GUI) oluşturmak için kullanılan bir modüldür.

PIL: Python Imaging Library, Python ile görüntü işleme işlemleri yapmak için kullanılan bir modüldür.

random: Rastgele sayılar oluşturmak için kullanılan bir modüldür.

ImageViewer: Ana sınıf, kullanıcı arayüzünü oluşturur ve fotoğrafı görüntüler.

init: Yapılandırıcı metot, sınıf özelliklerini ve başlangıç durumunu tanımlar.

self.master: Tkinter penceresini tutar.

self.score: Başlangıç puanıdır.

self.image_label: Fotoğrafın görüntülendiği etiket.

self.select_button: Kullanıcının fotoğraf seçmesine olanak tanıyan düğme.

self.score_label: Kullanıcının puanını görüntüleyen etiket.

self.mix_button: Parçaları karıştırmak için düğme.

self.grid_layout: 16 parçanın görüntülendiği çerçeve.

self.piece_list: Parça indekslerini tutar.

select_photo: Fotoğraf seçme işlemini gerçekleştirir.

display_photo: Seçilen fotoğrafı görüntüler ve parçalara ayırır.

mix_pieces: Parçaları karıştırır ve yeni konumlara yerleştirir.

move_piece: Kullanıcının parçaları sürükleyip bırakarak hareket ettirmesine olanak tanır ve doğru veya yanlış hamleler için geri bildirim sağlar.

is_adjacent: İki parçanın bitişik olup olmadığını kontrol eder.

name == 'main': Kodların doğrudan çalıştırılıp çalıştırılmadığını kontrol eder ve uygulamayı başlatır.

```
import tkinter as tk
from PIL import Image, ImageTk
from tkinter import filedialog,
messagebox
import random
```

Bu kod, Python programlama dilinde GUI (Grafiksel Kullanıcı Arayüzü) oluşturmak için kullanılan Tkinter adlı bir kütüphaneyi içeriyor. Ayrıca PIL (Python Imaging Library) kütüphanesinden Image ve ImageTk modülleri de kullanılarak resimleri işlemek için kullanılıyor.

Kod içerisinde "filedialog" ve "messagebox" adlı iki modül daha kullanılmış. "filedialog" modülü, kullanıcının bilgisayarında dosya açma ve kaydetme işlemlerini gerçekleştirebilmesini sağlar. "messagebox" modülü ise kullanıcıya mesaj göstermek için kullanılır.

Son olarak, "random" modülü rastgele sayılar üretmek için kullanılır.

```
class ImageViewer:
    def __init__(self, master):
        self.master = master
        self.master.title("Image
Viewer")
        self.score = 0  # başlangıç
puanı

        # GUI tasarımları
        self.image_label =
tk.Label(self.master)
        self.select_button =
tk.Button(self.master, text="Select
Photo", command=self.select_photo)
        self.score_label =
tk.Label(self.master, text="SCORE:
{}".format(self.score))
        self.mix_button =
tk.Button(self.master, text="KARIŞTIR",
command=self.mix_pieces)
        self.grid_layout =
tk.Frame(self.master)
```

Bu kod bir Resim Görüntüleyici sınıfını tanımlıyor ve sınıfın özelliklerini ve metodlarını belirliyor.

__init__ metodunda, sınıfın özellikleri tanımlanıyor. Bu özellikler arasında ana pencere master, başlangıç puanı score, resim görüntüsü için etiket image_label, fotoğraf seçme düğmesi select_button, puanı görüntülemek için etiket score_label, karıştırma işlemini başlatmak için düğme mix_button ve görüntü parçalarını düzenlemek için bir çerçeve grid_layout bulunuyor.

image_label, seçilen resmin görüntüsünü tutmak için kullanılırken, select_button kullanıcının fotoğraf seçmesine izin verir. score_label, mevcut puanı gösterirken, mix_button kullanıcının görüntü parçalarını karıştırmasına izin verir. grid_layout ise, resim parçalarının düzenlendiği bir çerçevedir.

```
# düzenleme
self.image_label.grid(row=0, column=0,
columnspan=4)
self.select_button.grid(row=1, column=0)
self.score_label.grid(row=1, column=1)
self.mix_button.grid(row=1, column=2)
self.grid_layout.grid(row=2, column=0,
columnspan=4)

# butonlar için bağlı liste oluştur
self.piece_list = [i for i in range(16)]
```

Bu kod bloğu, ImageViewer sınıfının init() metodunun bir parçasıdır ve GUI bileşenlerini yerleştirmek için kullanılır.

image_label: resim için bir etiket bileşenidir.

select_button: kullanıcının resim seçmek için tıkladığı bir düğme bileşenidir.

score_label: oyun puanını gösteren bir etiket bileşenidir.

mix_button: resim parçalarını karıştırmak için kullanıcının tıkladığı bir düğme bileşenidir.

grid_layout: resim parçalarını yerleştirmek için bir ızgara düzeni bileşenidir.

Bağlı liste, resim parçalarını yerleştirmek için kullanılır ve 16 tamsayı değeri içerir (0-15 arasındaki sayılar).

```
def select_photo(self):
    file_path =
filedialog.askopenfilename(filetypes=[("
Image Files", "*.png;*.jpg;*.bmp")])
    if file_path:
        self.display_photo(file_path)
```

select_photo fonksiyonu, kullanıcının bilgisayarında bulunan bir resim dosyasını seçmesine olanak tanır. Dosya seçimi yapılınca, seçilen dosya yolunu alarak display_photo fonksiyonunu çağırır. Bu fonksiyon, seçilen resmi görüntüler.

Bu metot seçilen resmi ekranda göstermek ve resmi 16 parçaya bölüp butonlara yerleştirmek için kullanılır.

İlk olarak, file_path parametresi olarak aldığı resmi Image.open() fonksiyonu ile açar ve ImageTk.PhotoImage() fonksiyonu ile tkinter ile uyumlu bir fotoğraf nesnesi haline dönüştürür.

Son olarak, puanın güncellenmiş halini skor tablosuna yazdırır.

Sonra, resmin boyutuna göre her biri piece_width ve piece_height boyutunda olan 16 parçaya ayırır.

Önceki butonları temizlemek için winfo_children() fonksiyonunu kullanır ve sonrasında resimdeki her parça için bir buton oluşturur. Her butonun resmini, butona tıklandığında çağrılacak move_piece() metodu için bir argüman olan butonun orijinal indeksini taşıyan bir lambda fonksiyonuyla birlikte tanımlar.

. Son olarak, resimdeki her parçanın orijinal pozisyonunu takip eden bir piece_list listesi, doğru ve yanlış konumlar için iki adet boş liste oluşturur

```
def display_photo(self, file_path):
    # resim yükle, görüntüle
    image = Image.open(file_path)
    photo = ImageTk.PhotoImage(image)

self.image_label.configure(image=photo)
    self.image_label.image = photo

    # Resmi 16 parçaya böl
    piece_width = image.width // 4
    piece_height = image.height // 4

    # önceki düğmeleri temizle
    for widget in
self.grid_layout.winfo_children():
        widget.destroy()

    # resmi sırayla parça parça
butonlara böl
    self.buttons = []
    for i in range(16):
        row, col = divmod(i, 4)
        left = col * piece_width
        top = row * piece_height
        right = (col + 1) * piece_width
        bottom = (row + 1) *
piece_height
        piece = image.crop((left, top,
right, bottom))
        piece_photo =
ImageTk.PhotoImage(piece)
        piece_button =
tk.Button(self.grid_layout,
image=piece_photo,

command=lambda index=i:
self.move_piece(index))
        piece_button.image = piece_photo
        piece_button.grid(row=row,
column=col)

self.buttons.append(piece_button)

    # parçaların orijinal sırasını takip
et
    self.piece_list = list(range(16))
    self.correct_positions = []
    self.incorrect_positions = []
```

Bu fonksiyon, resim parçalarını karıştırmak ve düğmeleri yeni konumlarıyla güncellemek için kullanılır. İlk olarak, self.piece_list adlı liste rastgele karıştırılır. Ardından, 16 parçalı resmin her bir parçası için, bu parçanın yeni konumuna göre ilgili düğmenin grid() yöntemi çağrılır. Yani, düğmelerin sırası ve konumu karıştırılır.

```
def mix_pieces(self):
    # Parça listesini karıştır
    random.shuffle(self.piece_list)
    # Düğmeleri yeni konumlarla yeniden
çiz
    for i in range(16):
        row, col =
divmod(self.piece_list[i], 4)
        self.buttons[i].grid(row=row,
column=col)
```

```
def move_piece(self, index):
    # Seçilen parçanın bulunduğu satır
ve sütun bilgilerini elde et
    row, col =
divmod(self.piece_list[index], 4)

    # Bitişik parça indekslerini bul
    adjacent_indices = []
    if row > 0:

adjacent_indices.append(self.piece_list[
index - 4])
    if row < 3:

adjacent_indices.append(self.piece_list[
index + 4])
    if col > 0:

adjacent_indices.append(self.piece_list[
index - 1])
    if col < 3:

adjacent_indices.append(self.piece_list[
```

```
index + 1])

    # Doğru hamle yapıldığını kontrol et
    if self.piece_list[index] in
self.correct_positions:
        messagebox.showinfo("Doğru
Hamle!", "Tebrikler! Doğru hamle
yaptınız.")
        self.score += 5

self.buttons[self.piece_list[index]].con
fig(state="disabled")

self.correct_positions.remove(self.piece
_list[index])
        if not self.correct_positions:

messagebox.showinfo("Tebrikler!",
"Bulmacayı tamamladınız!")
    else:
        messagebox.showerror("Yanlış
Hamle!", "Hata! Yanlış hamle yaptınız.")
        self.score -= 10

self.incorrect_positions.append(self.pie
ce_list[index])
        for pos in adjacent_indices:
            if pos in
self.correct_positions:

self.buttons[pos].config(state="normal")
        self.correct_positions =
list(range(16))

self.correct_positions.remove(self.piece
_list[index])
        self.incorrect_positions = []

    self.score_label.config(text="PUAN:
{}".format(self.score))
```

Seçilen parçanın doğru veya yanlış hamle yapılıp yapılmadığını kontrol eden, puan hesabını ve skor tablosunu güncelleyen bir fonksiyondur. Eğer seçilen parça doğru hamle yapılan parçalar arasındaysa, kullanıcıya tebrik mesajı verilir, puanı arttırılır, o parçanın butonu devre dışı bırakılır ve doğru hamle yapılan parçalar listesinden çıkarılır. Doğru hamle yapılan parça sayısı 0'a düştüğünde, kullanıcıya bulmacayı tamamladığına dair mesaj verilir.

Eğer seçilen parça yanlış hamle yapılan parçalar arasındaysa, kullanıcıya hata mesajı verilir, puanı azaltılır ve yanlış hamle yapılan parçalar listesine eklenir. Ayrıca, bitişik parçaların doğru hamle yapılan parçalar listesinden çıkarılır ve tekrar doğru hamle yapılabilecek parçalar listesi, 16 parçanın indekslerinden oluşan bir liste olarak yeniden oluşturulur.

```
def is_adjacent(self, index1, index2):
            # İki parçanın bitişik olup
olmadığını kontrol et
            return abs(index1 - index2)
== 1 or abs(index1 - index2) == 4

if __name__ == '__main__':
    root = tk.Tk()
    viewer = ImageViewer(root)
    root.mainloop()
```

Bu kod, bir görüntüyü 16 parçaya bölüp, bu parçaları karıştıran ve ardından kullanıcının bu parçaları orijinal konumlarına göre yeniden sıralamasını sağlayan bir oyun oluşturur. Oyuncu doğru hamle yaparsa puan kazanır, yanlış hamle yaparsa puan kaybeder. Oyun, tkinter modülü kullanılarak bir GUI (grafik kullanıcı arayüzü) olarak tasarlanmıştır. Kod, if __name__ == '__main__': bloğundaki kodlarla çalıştırılır ve ana pencere başlatılır.

5. KARŞILAŞTIĞIMIZ ZORLUKLAR

- Yapboz parçalarının yerini değiştirme özelliği olan def move_piece(self, index) fonksiyonunu yazmak.
- Def move_piece fonsiyonun da doğru cevap olduğunda parça kitleniyor fakat karıştır butonu ile sabit kalmıyor bu problemi de çözmeye çalışıyoruz.

KAYNAKÇA

[1] https://wmaraci.com/nedir/gui#:~:text=GUI%20(Graphical%20User%20Interface)%2C,yard%C4%B1mc%C4%B1%20olmas%C4%B1%20amac%C4%B1yla%20geli%C5%9Ftirilmi%C5%9F%20tasar%C4%B1mlard%C4%B1r.

[2] www.notion.so

[3] Her Yönüyle Python Fırat Özgül