BURSA ULUDAĞ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2022 – 2023 Eğitim Öğretim Yılı Güz Yarıyılı

Ödev 2 – 2

Ferit Yiğit BALABAN
032190002
032190002@ogr.uludag.edu.tr

https://github.com/fybx/assignments/blob/main/BMB2009/Assignment2_2/Program.cs

Soru 1 - Kursta geldiğimiz noktaya kadarki konuları kullanarak 5'li bir yılanı w, a, s, d tuşlarıyla dört yönde hareket ettiriniz. Yılanın başlangıçtaki konumu 15'inci satır 38, 39, 40, 41, 42'inci sütunlarda olacaktır. Yılanın kuruğunu temsil etmek için '*' karakterleri, kafasını temsil etmek için '0' karakteri kullanılacaktır.

Cevap 1 -
```
public static class Assignment2_2
{
    private static readonly int[,] InitialBody = {
        { 38, 15 },
        { 39, 15 },
        { 40, 15 },
        { 41, 15 },
        { 42, 15 }
    };

    /// <summary>
    /// Field _gameState defines 3 states the game can take.
    /// If state = 0 game is waiting for a keypress to be started
    ///    bigger  0 game is started, every keypress calculates collisions then redraws
scene if no collision is made
    ///    smaller 0 game is lost, ask for spesific keypress to restart, if pressed set
state to 0
    /// </summary>
    private static int _gameState = 1;

    /// <summary>
    /// Field _snakeBody gives coordinates of snake body parts with the last (4th)
element being the head.
    /// </summary>
    private static int[,] _snakeBody = (int[,])InitialBody.Clone();

    public static void Main(string[] args)
    {
        ConsoleKeyInfo key;
        start:
        Redraw(0);
```

```
        _gameState = 1;
        while (_gameState is 1)
        {
            Redraw(_gameState, _gameState is not 0);
            key = Console.ReadKey();
            int decision = key.Key switch
            {
                ConsoleKey.UpArrow ⇒ 1,
                ConsoleKey.RightArrow ⇒ 2,
                ConsoleKey.DownArrow ⇒ 3,
                ConsoleKey.LeftArrow ⇒ 4,
                ConsoleKey.Q ⇒ 5,
                _ ⇒ 6
            };
            if (decision is 5)
                _gameState = -2;
            CalculateSnakeBody(decision);
            if (CheckCollision())
                _gameState = -1;
            _gameState = _gameState > -1 ? 1 : _gameState;
        }

        if (_gameState is -1)
        {
            Console.Clear();
            Console.WriteLine("Kaybettiniz. Tekrar oynamak için \"r\", çıkmak için \"q\"
tuşuna basınız...");
            do
            {
                key = Console.ReadKey();
                if (key.Key is ConsoleKey.R)
                    goto start;
            } while (key.Key is not ConsoleKey.Q);
        }
    }

    /// <summary>
    /// Does collision checks against console window borders and itself.
    /// </summary>
    /// <returns>Returns true if collision occurs, otherwise false.</returns>
    private static bool CheckCollision()
    {
        int headX = _snakeBody[4, 0];
        int headY = _snakeBody[4, 1];

        if (headX is -1 || headY is -1) return true;
        if (headX == Console.WindowWidth + 1 || headY == Console.WindowHeight + 1) return
true;
        for (int i = 0; i < 4; i++)
            if (_snakeBody[i, 0] == headX && _snakeBody[i, 1] == headY)
                return true;
        return false;
    }

    /// <summary>
    /// Calculates new coordinates of snake's body parts.
    /// </summary>
    /// <param name="direction">Direction player stepped to; up being 1, right 2, down 3
and left 4.</param>
    /// <returns>Returns updated coordinate array of body.</returns>
```

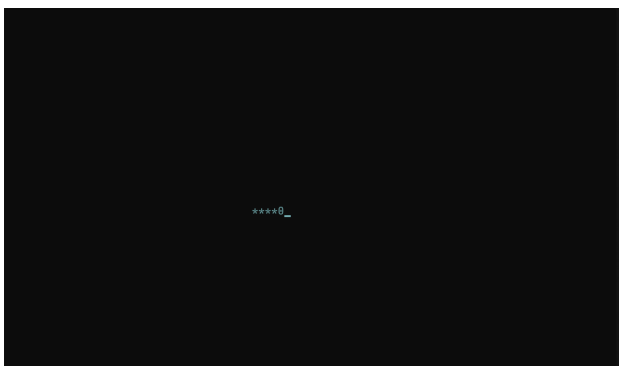
```
    private static void CalculateSnakeBody(int direction)
    {
        if (direction is < 0 or > 4)
            return;
        for (int i = 0; i < 4; i++)
        {
            _snakeBody[i, 0] = _snakeBody[i + 1, 0];
            _snakeBody[i, 1] = _snakeBody[i + 1, 1];
        }
        _snakeBody[4, 0] += direction switch
        {
            2 => 1,
            4 => -1,
            _ => 0
        };
        _snakeBody[4, 1] += direction switch
        {
            1 => -1,
            3 => 1,
            _ => 0
        };
    }

    /// <summary>
    /// Draws a scene or state to console according to state parameter.
    /// </summary>
    /// <param name="state">Current game state</param>
    /// <param name="draw">True by default, can be used to skip states</param>
    private static void Redraw(int state, bool draw = true)
    {
        if (!draw)
            return;

        if (state is 0)
            _snakeBody = (int[,])InitialBody.Clone();
        else
            Console.Clear();
        for (int i = 0; i < 5; i++)
        {
            Console.SetCursorPosition(_snakeBody[i, 0], _snakeBody[i, 1]);
            Console.Write(i is 4 ? "0" : "*");
        }
    }
}
```

Cevap 1 – Ekran Görüntüsü

https://drive.google.com/file/d/1kEso5UvYxIetx17PBxCa9OfFLM8o8yNH/view?usp=sharing



Cevap 1 – Açıklaması

InitialBody statik readonly özelliği: yılanın başlangıç konumunu saklar.

_snakeBody oyuncu tarafından değiştirilebilen yılanın konumunun saklandığı 5x2 boyutlu bir matristir. Dizilerin dizisidir ve temel veri birimi **int32**'dir.

_gameState oyunun içinde bulunduğu durumu işaretler ve temel oyun mantığını inşa etmek için kullanılır.

-2 değeri oyunu bitirir

-1 değeri oyunun kaybedildiğini işaretler ve "tekrar oynamak ister misiniz" sorusunu ekrana getirir

0 değeri oyunun başlamak üzere olduğunu işaretler

1 değeri oyunun devam ettiğini işaretler

CheckCollision() metodu öncelikle kafanın yatay ve dikey eksendeki pozisyonlarını saklar. Eğer kafanın yatay veya dikey komponenti -1 değerinde ise **true** değer döndürür. Yani eğer tavana ya da çerçevenin soluna *çarpıldıysa* **true** döner. Sonrasında kafanın yatay ve dikey komponentlerinin pencerenin maksimum genişliği ve uzunluğu ile kıyaslanması ile pencerenin sağına ya da tabana çarpılıp çarpılmadığına bakılır. Son olarak **for döngüsü** ile vücudu oluşturan diğer noktaların herhangi birisi ile kafanın komponentlerinin birebir oturup oturmadığını kontrol ediyor. Yani kafanın vücuda ait bir noktayla kesişmesi durumunda **true** değer döndürülüyor.

CalculateSnakeBody(int direction) eğer verilen **direction** argümanı *0, 1, 2 veya 3* değilse metottan çıkılır. **direction** eğer

0 ise klavyede yukarı ok tuşuna tıklanmıştır

1 ise klavyede sağ ok tuşuna tıklanmıştır

2 ise klavyede aşağı ok tuşuna tıklanmıştır

3 ise klavyede sol ok tuşuna tıklanmıştır

Metottan çıkılmadıysa öncelikle yılanın vücudunu oluşturan ve başı içermeyen parçaları sırası ile kendisinden bir sonraki vücut parçasının konumuna ilerletilir. Örneğin 3. vücut parçası, 2. vücut parçasının yatay ve dikey komponentlerini alırken, 2. vücut parçası da 1. vücut parçasına ait olanları alır. 1. vücut parçası da kafanın komponentlerini alır. **for döngüsü** tamamlandığında kafanın yatay ve dikey komponentlerinin hesabı için **direction** değişkeninden yararlanılır. *1 ya da 3* sırayla yukarı ve aşağıya gitmeyi işaretliyorsa dikey hareket yaşandığı için dikey yani 2. komponenti değiştirmesi gerekir. Aynı mantıkla *2 ya da 4 sırasıyla* sağ ve sola gitmeyi işaretlediğinden yatayda yapılan hareketi göstermek için kafa noktasının sadece 1. komponenti değiştirilir. Dolayısıyla yatayda hareket varsa dikey, dikeyde hareket varsa yatay komponent güncellenmez.

Redraw(int state, bool draw) verilen **state** argümanı ile yılanın ya ilk konuma çizilmesi sağlanır ya da oyun süreci içerisinde yılanın aktif konumunu gösteren **snakeBody** değişkeninde tutulan veri ekrana çizilir. Çizim için kullanılan **for döngüsü** vücudun 4 parçası ve baş kısmı için **Console.SetCursorPosition()** metodu yardımıyla **snakeBody** değişkeninin her bir parçasının yatay ve dikey komponentlerinin işaretlediği konuma imleç yerleştirilir. Sonrasında **Console.Write** ile yazma işlemi gerçekleşirken eğer iterasyon değişkeni **i = 5** ise yani baş kısmı çiziliyor ise boolean ternary operatörü ile **0** karakteri aksi takdirde ***** karakteri ekrana yazılır.

Ana döngü, Main() metodunda *start isminde bir* ***label*** oyunun kaybedildiği noktada sorulan soruya eğer klavyede **r** tuşuna basarak yanıt verildiyse tekrardan başlayabilmek için bulunuyor. Bu noktadan itibaren öncelikle **gameState** değişkeni **1** durumunu işaretlerken **gameState = 1** iken çalışacak **while** döngüsü başlatılıyor. **while döngüsü** altında öncelikle **Redraw()** çağırılıyor. Sonrasında ise kullanıcıdan bir tuşa basması beklenmeye başlanıyor.

Tuşa basıldığında bir **switch ifadesi** yardımıyla basılan tuş sayısal olarak kodlanıyor (kodu direkt okuduğunuzda hangi tuşun sayısal değerinin ne olduğunu görebiliyorsunuz) bu kodlanan değer **decision** isimli lokal değişkende saklanıyor. Eğer **decision = 5** ise **gameState** değişkeni **-2** olarak ayarlanıyor. Bunu takiben **CalculateSnakeBody()** metodu *sayısal olarak kodlanmış tuş verisini içeren* **decision** değişkeni ile çağırılıyor. Bu hesaplama bittiğinde **snakeBody** değişkeninin güncellenmiş olacağını açıklamıştık, bundandır ki hemen **CheckCollision()** metodu ile çakışma testi yapılıyor, çakışma varsa da **gameState** -1 değerini alıyor. Bir döngünün sonuna gelindiğinde eğer **gameState** -1'den büyük değerler için 1 değerine eşitlenirken, -1 ve -1'den küçük değerler için aynı kalır. Böylece oyunun bitişi ile alakalı yapılan işaretlemeler değiştirilmezken oyunun devam ettiği bilgisi saklanabilir.

Döngü yalnızca giriş aşaması değeri olan 0 ve negatif değerler için kırılacaktır. Eğer değer -1 ise bu durum oyunun kaybedildiğini gösterir. Dolayısıyla da önce ekran temizlenir, sonrasında ekrana mesaj basılarak yeniden başlatma ya da çıkış yapma seçeneği sunulur. Geçerli bir yanıt verilene dek (yani ya **r** ya da **q** tuşuna basılana dek) sorgu ekranda kalır.

Soru Puanlaması

Cevap : 60

Ekran Görüntüsü : 15

Açıklama : 25