Yıldız Teknik Üniversitesi

Bilg. Bil. Giriş Dönem Projesi

Hazırlayan

Emir Kerem Öztürk

20011613

30 Aralık 2022

TABLE OF CONTENTS

1. Giriş **X**

2. Algoritmalar **X**

3. Çalıştırılması **X**

4. Video **X**

1. Giriş

Bu rapor Bilgisayar Bilimlerine Giriş dersi için verilen Yılan Oyunu dönem projesi hakkındadır.

Yılan oyunu için bizden istenilen, Kullanıcıdan alınan satır ve sütun (NxM) sayısına göre önce boş bir tablonun doldurulması ardından, köşelerinin çizilmesi ve yemek sayısına göre bu yemeklerin rastgele bu tabloya dağıtılmasıdır. Ardından Yılanın başı da tabloda rastgele seçilen bir yere 1 olarak yerleştirilir. Yılan etrafta yön okları ile dolaşıp yemeği toplar, topladıkça kuyruğu başındaki sayıya 1 eklenerek artar ve yılanın başını takip eder. Yılan kendi içinden geçemez ama denerse oyun bitmez, oyun ya tüm yemekler toplanılınca kazanıldı olarak biter veya Yılan # ile belirtilen kenarlara değerse kaybettiniz diye biter.

2. Algoritmalar

```
for (i = 0; i < N; i++) {
    for (j = 0; j < M; j++) {
        gameBoard[i][j] = EMPTY;
    }
}
```

Öncelikle verilen (NxM) e göre tabloyu doldurdum.
Ardından Yılanın kafasını 1 olarak ve verilen toplam yemek sayısına göre **for** döndüm ve yemekleri de tabloya 0 olarak rastgele yerlere yerleştirdim.

```
gameBoard[snakeX][snakeY] = SNAKE;
printf("Yemek sayisini gir: ");
scanf("%d", &foodCount);
for (i = 0; i < foodCount; i++) {
    do {
        foodX[i] = rand() % N;
        foodY[i] = rand() % M;
    } while (gameBoard[foodX[i]][foodY[i]] != EMPTY);
    gameBoard[foodX[i]][foodY[i]] = FOOD;
}
```

Bunlardan sonra ise oyunun algoritması bir **do-while** döngüsü ile başlar. Önce duvarlar belirlenir ve '#' ile gösterilir ve duvara çarptı mı kontrolü ardından yapılır. (alttaki resim)

```
do {
    for (i = 0; i < N; i++){
        if (i == 0){
            printf(" ");
            for (k = 0; k < M; k++)
                printf("%c", WALL);
            printf("\n");
        }
        printf("%c", WALL);
        for (j = 0; j < M; j++){
            printf("%c", gameBoard[i][j]);
        }
        printf("%c", WALL);
        printf("\n");
        if (i == N - 1){
            printf(" ");
            for (k = 0; k < M; k++)
                printf("%c", WALL);
            printf("\n");
        }
    }

    if (gameBoard[snakeX][snakeY] == WALL){
        printf("Oppss, carptiniz. Oyun bitti! \n");
        break;
    }
```

Bunun ardından kullanıcıya gideceği yol sorulur (w = yukarı, a = sol, s = aşağı, d = sağ) gideceği yön belirlenir ve duvara çarptı mı kontrolü yapılır.

```
if (direction == 'w') {
    if (prev == 's')
        continue;
    snakeX--;
}
else if (direction == 'a') {
    if (prev == 'd')
        continue;
    snakeY--;
}
else if (direction == 's') {
    if (prev == 'w')
        continue;
    snakeX++;
}
else if (direction == 'd') {
    if (prev == 'a')
        continue;
    snakeY++;
}
```

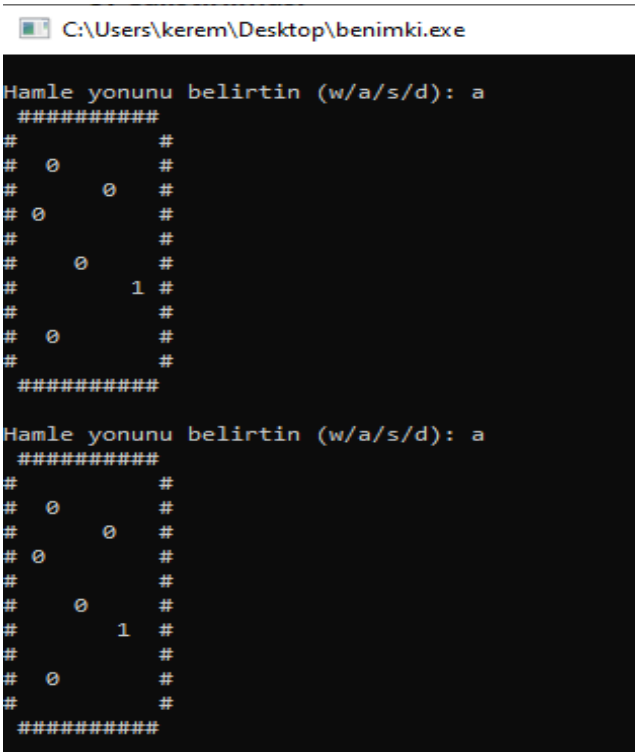
```
if (snakeX < 0 || snakeX >= N || snakeY < 0 || snakeY >= M) {
    printf("Oyun bitti! Oyun alanini terk ettin...");
    exit(0);
}
```

Kontrolden sonra oyun tablosunun içi boşaltılır. Ardından yılanın kuyruk kısmını koordinatlarını giden yöne göre birer kaydırırız. Yemek listesindeki koordinatlar tek tek gezilerek şu an yılanın konumu bu yemeklerden birine denk geliyor ise bu koordinat yemek listesinden çıkarılır ve yılanın kuyruk koordinatları içerisine eklenir. Yılanın boyu bir arttırılır yemek sayısı bir azaltılır. Yılan kuyruk bilgisinin koordinatlarına göre yılan oyun tablosunun içine yazılır.

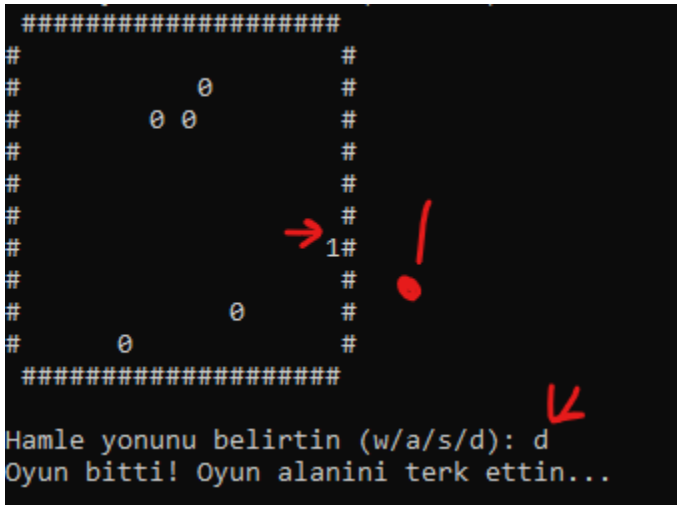
```
for (i = 0; i < N; i++){
    for (j = 0; j < M; j++){
        gameBoard[i][j] = EMPTY;
    }
}

for (i = snakeSize - 1; i > 0; i--){
    tailX[i] = tailX[i - 1];
    tailY[i] = tailY[i - 1];
}
tailX[0] = snakeX;
tailY[0] = snakeY;

int foodIndex = -1;
for (i = 0; i < foodCount; i++){
    if (foodX[i] == snakeX && foodY[i] == snakeY)
        foodIndex = i;
    if (foodIndex > -1 && i >= foodIndex && i <= foodCount - 1){
        foodX[i] = foodX[i + 1];
        foodY[i] = foodY[i + 1];
    }
}
if (foodIndex >= 0){
    foodCount--;
    tailX[snakeSize] = x;
    tailY[snakeSize] = y;
    snakeSize++;
}

for (i = 0; i < snakeSize; i++)
    gameBoard[tailX[i]][tailY[i]] = i + '1';
```

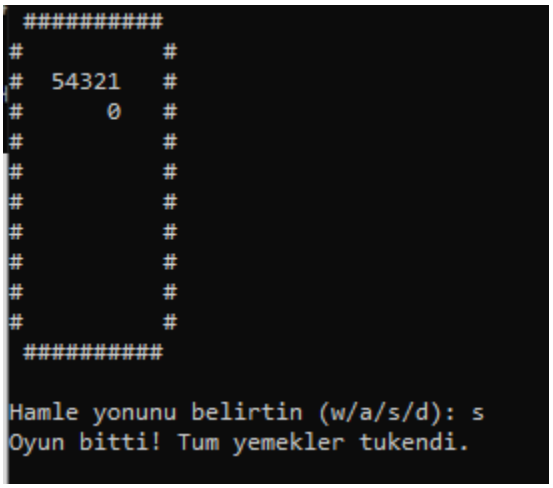
En son ise yemek sayısı kontrol edilir, eğer 0 a ulaşmış ise oyun biter Kullanıcıya tüm yemeklerin tükendiği mesajı gösterilir. **do-while** döngüsün hep devam etmesi için **while (1)** olarak şart konulur.

3. Çalıştırılması



Kenara çarpma durumu:





4. Video

https://www.youtube.com/watch?v=abx1ssXprd8&ab_channel=Obsessed