**YILAN BÜYÜTME OYUNU BBG PROJE ÖDEVİ**

Deniz Ege Dereli/22011066

**Tanıtım:**

Yılan büyütme oyunu, kontrol ettiğimiz bir yılanı adım adım oynatarak yemeklere götürmeye ve büyütmeye çalıştığımız bir oyun. Yılanın parçaları, 1-2-3… şeklinde sıralanıyor. Tüm yemekler tükendiğinde oyun bitiyor. Ayrıca yılan bir duvara çarptığında veya kendi üstüne geldiğinde de oyun bitiyor.

Bu ödevi, programı iki ayrı parçaya ayırarak yaptım. İlk parça, yani kurulum kısmı, kullanıcıdan oyun ile ilgili bilgilerin alındığı kısım. Mesela oyun alanının boyutu, oyun alanındaki yemek sayısı gibi. İkinci parça ise oyunun asıl oynandığı kısım. Yılanın hareketinden yemeklerin yenmesi, hepsi bu kısımda geçiyor.

**Birinci Kısım:**

Öncelikle değişkenlerin tanımlanmasıyla başlıyoruz. Ardından yaptığımız ilk şey, *srand* fonksiyonu ile rastgele fonksiyonu *rand* için *seed* oluşturmak. Amacımız, her açılışta oyunun farklı çalışması. Bunun için de *time.h* kütüphanesinden *time* fonksiyonunu kullanıyoruz. Bu fonksiyon, şu anki tarih ve saati tek bir sayı haline çeviriyor. Ve bu sayı, her açılışta farklı olacağından *seed* oluşturmak için birebir.

Ardından oyun alanının boyutunu alıyoruz. Boyutun minimum 2x2 olması önemli. Aksi takdirde oyun çalışmayacaktır. Ardından da yemek sayısını soruyoruz. Oyun alanı MxN boyutlarında ise yemek sayısı en çok M\*N-1 olabilir. Bunun nedeni, yılana yer ayırmak. Elbette oyun alanının boyutları negatif olamaz. Ve yemek sayısı da sıfırdan büyük olmalı.

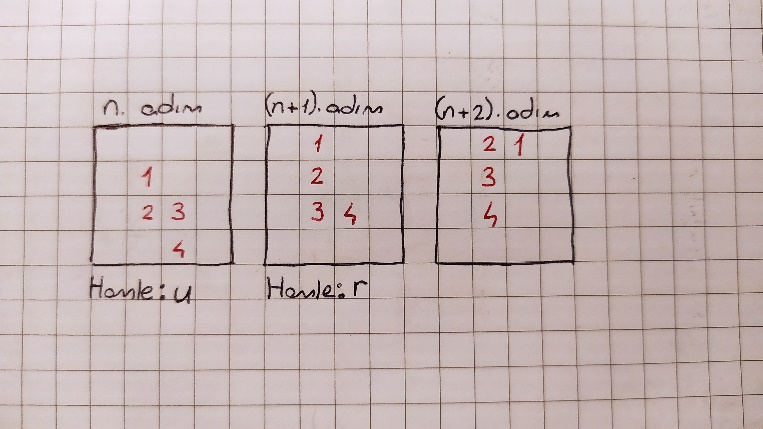
Oyun alanına yemekler ve yılanın başlangıç noktası (kafası) rastgele yerleştirilir. Bunun için *rand* ile oluşturulan sayıya mod işlemi uygulanır, yani sayının kalanı alınır. Buradaki mantık, rastgele oluşturulmuş sayıyı istediğimiz aralığa çekmektir. Bu aralık, oyun alanının en ve boyudur. Örneğin mod 5 yaptığımız zaman sonuç, ancak 0, 1, 2, 3, 4 olabilir. Burada *food\_count* ve *total\_food* isimli iki değişken kullandım. İlki, yemek sayısını tutan bir sayaç. İkincisi ise oyun boyunca toplam yemek sayısını tutar. Ardından yılan için başlangıç noktası seçilir.

Bu kısım aslında kodun ikinci parçasıyla da ilişkili, ancak programın içinde önce yön soruluyor, sonra tahta yazdırılıyor. Bu nedenle tahtanın başlangıç halini yazdırıyoruz. Renklendirmek için de ANSI renk karakterlerini kullanıyorum. Burada önemli bir ayrıntı, her bir hücre üç karakter genişliğinde. Bu sayede iki basamaklı sayıların da basıldıktan sonra güzel görünmeleri sağlanmış oluyor. Ayrıca bir çerçeve de ekrana basılıyor.

**İkinci Parça:**

Burası, oyunun asıl çalıştığı kısım. Oyun çalıştıkça *while* döngüsü çalışmaya devam ediyor. *game\_running* değişkeni, oyunun devam edip etmediğini bildiriyor.

Kaçıncı adımda olunduğu yazdırıldıktan sonra, hareket edilecek yön soruluyor. En başta, bu döngü öncesinde sıfırıncı adımın bastırıldığını söylemiştim. Yön kullanıcıdan alındıktan sonra, sırasıyla yılanın gövdesi ve başı hareket ettiriliyor. Böyle bir sıralamanın olmasının nedeni, yılanın hareket algoritmasında yatıyor.

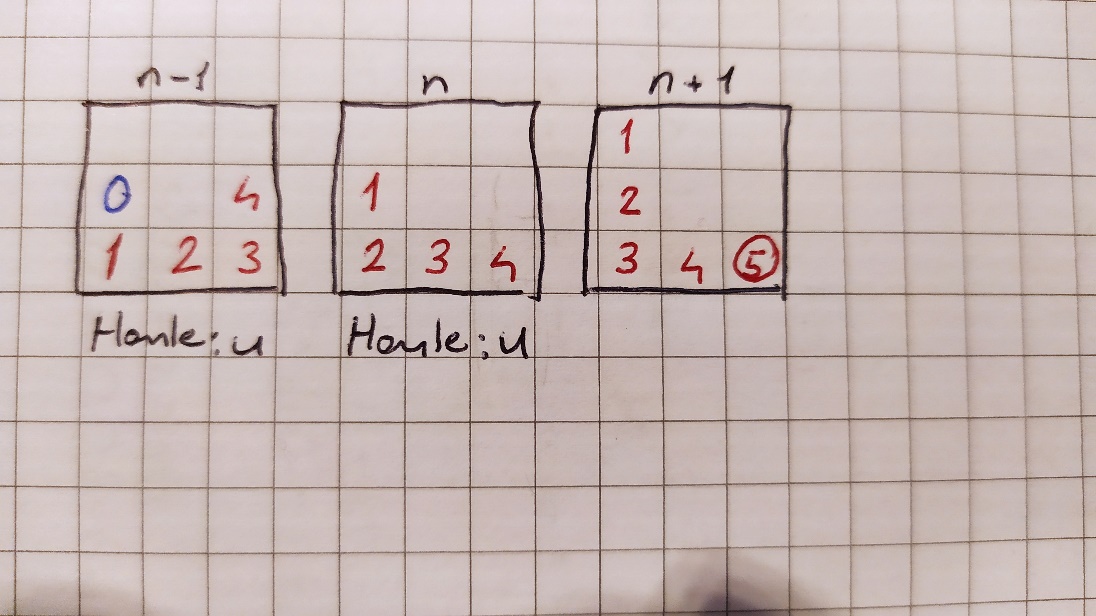
İncelemelerim sırasında fark ettim ki yılanın her bir parçası, her adımda bir önceki parçanın yerine geçiyor. Ardından kafa da istenilen yere yazılıyor. Kafanın vücuttan sonra hareket ettirilmesinin nedeni ise kafadan hemen önceki parçanın, bir sonraki adımda kafanın yerine gelecek olması. Hareket sistemini de bundan yararlanarak tasarladım.

Yılanın duvarlara çarptığında yok olması ve oyunun bitmesi için, yılanın kafasının oyun alanında olup olmadığını test ediyorum. Eğer dışarı çıkmışsa, yılanın uzunluğu sıfır yapılıyor ve oyun alanı boş olarak yazdırıldıktan sonra da oyun sonlanıyor.

Sonrasında, yemeğin üzerinde olunup olunmadığının test edilmesi geliyor. Ancak bu kısım, küçük bir ayrıntı içeriyor. Yılan, yemeğin üzerine gelince değil, bir sonraki adımda büyümeli. Bu sorun ile uzun süre uğraştıktan sonra, iki *if* ile çözülebilecek bir sistem geliştirdim. Bu konuda ilk düşüncem, yılan bir yemek üzerine geldiği zaman bunu bir değişken ile saklamak, sonraki adımda da yılanı büyütmekti. Bunun için bir *if…else* sistemi kullanmayı denediysem de, üst üste birden çok yemek yenmesi durumu sıkıntı çıkartıyordu. Bu nedenle, yemek üzerinde olunup olunmadığının her adımda yeniden değerlendirileceği, iki *if* içeren yeni bir sistem geliştirdim.

Bu sistem için, *on\_food* adlı bir değişken kullandım. Yılanın yemek üzerinde olup olmadığını saklıyor. İlk *if* önermesi, önceki turda yemek üzerinde olunduysa çalışıyor ve yılanı büyütüyor. İkinci *if* önermesi ise yılanın bu tur yemek üzerinde olup olmadığına bakıyor.

Yılanı büyütmek, yine üzerinde uzun süre düşündüğüm kısımlardan biriydi. Her bir parçanın, her adımda, bir önceki parçanın yerine geldiğini söylemiştim. Yılan, yemek üzerine geldiği adımdan bir sonraki adımda büyüyor. Yılan, yemek üzerine *n.* adımda geliyor ve *(n+1).* adımda da büyüyor olsun. Yılanın yeni parçası, *(n+1).* adımda, *n.* adımda yılanın en son parçasının olduğu konumda olmalıdır. Bu sayede yılanın büyümesi sağlanır.



Yemekler tükendiği zaman, oyun biter. Bunu kontrol etmek için de her yemek yendiğinde *food\_count* değişkeni kullanılır. Bu değişkenin yemekler oyun alanına yerleştirilirken de kullanıldığını söylemiştim. O işlemler sonrası *food\_count*,sıfır olmaktadır. Şimdi de *food\_count*, *total\_food* değişkenine ulaşıncaya kadar bir arttırılır.

Son olarak da yılanın, daha doğrusu tüm oyun alanının ekrana yazdırılması var. Yılanın kafası dışında parçalarının koordinatları, ayrı bir dizide saklanıyor. Önce, bu koordinatlardaki parçalar ve kafa, oyun alanı matrisine yazılıyor. Daha sonra, sınırlarla birlikte oyun alanı, içinde yılan da bulunacak şekilde ekrana yazılıyor. Yazma işlemi tamamlandıktan sonra da işlemlerde sıkıntı çıkartmaması için yılan, oyun alanından kaldırılıyor. Ve her adım bu işlem tekrar ediyor.

**Kapanış:**

Oyun sonlandığı zaman, oyunun içinde bulunduğu asıl döngü sonlanıyor ve ekrana, kullanıcının kaç adım oynadığı, ne kadar yemek yediği gibi bilgiler yazdırılarak program sonlanıyor.