MUSTAFA AKILLI

131044017

AGARI YO v1.2.1 PROJECT REPORT

**Definition of the game:**

Oyunun adı AgarI YO. Versiyonu 1.2.1’dir. İlk başladığımda 1.0.0 idi. 21 gün sonunda v1.2.1 ortaya çıktı.

Oyuna küçük bir yuvarlak olarak başlıyorsunuz. Çevrenizdeki yemleri yiyerek büyüyorsunuz. Büyürken de hızınız azalıyor. Sizden büyük botlar sizi yemeğe çalışıyor. Siz ise sizden küçük olan botları yiyebilirsiniz. Yeterince büyüklüğe ulaştığınızda bölünebilme yeteneğini kazanırsınız. Yeşil alanların üstünden geçerken eğer yeşil alandan büyükseniz bölünürsünüz.

Oyunda hayatta kalmaya çalışıyorsunuz. Tüm parçalarınız yenildiği an oyun biter.

**Software Design:**

Oyun 32 adet fonksiyondan oluşmakta. Oyundaki unsurlar şu şekilde: Oyuncu(player), oyuncunun parçaları(split), botlar, yemler, yeşil şeyler, arka plan.

Oyuncunun özellikleri: Yemleri yemek, botları yemek, botlara yenilmek, yeşil alanlarda parçalanmak, bölünebilmek, bölündüğü parçalar ile birleşebilmek.

Splitin özellikleri: Oyuncunun özellikleri ile aynı.

Botların özellikleri: Yemleri yemek, botları yemek, botlara yenilmek, oyuncuyu yemek, oyuncuya yenilmek, yeşil alanlarda parçalanmak, bölünebilmek, bölündüğü parçalar ile birleşebilmek.

Yemlerin özellikleri: Botlara, oyuncuya ve splitlere yenilmek.

Yeşil şeylerin özellikleri: Botları, oyuncuyu ve splitleri parçalamak.

Arka Plan: Oyun sırasında arkada yer alan resim.

**Struct;**

**player\_feature\_t (PLAYER):**

location\_x:bulunduğu x koordinatı

location\_y:bulunduğu y koordinatı

location\_x\_last: hareket etmeden önceki bulunduğu x koordinatı

location\_y\_last: hareket etmeden önceki bulunduğu y koordinatı

radius:yarıçapı

speed:hızı

live\_or\_dead:1 ise yaşıyor. 0 ise yenmiş ve yok edilmiş

**move\_feature\_t(HAREKET YONLERI):**

left: SOL

right: SAĞ

up: YUKARI

down: AŞAĞI

**bots\_feature\_t(BOT):**

location\_x: bulunduğu x koordinatı

location\_y: bulunduğu y koordinatı

radius: yarıçapı

speed: hızı

target\_location\_x: gideceği hedefin x koordinatı

target\_location\_y: gidecegi hedefin y koordinatı

distance\_target\_location\_x: x koordinatında gideceği hedefe olan uzaklığı

distance\_target\_location\_y: y koordinatında gideceği hedefe olan uzaklığı

random\_color\_red: rastgele kırmızı tonu

random\_color\_green: rastgele yeşil tonu

random\_color\_blue: rastgele mavi tonu

**location\_t(KOORDİNATLAR):**

location\_x: x koordinatı

location\_y: y koordinatı

**division\_t(SPLIT):**

location\_x: bulundugu x koordinatı

location\_y: bulunduğu y koordinatı

location\_x\_last: hareket etmeden önceki bulunduğu x koordinatı (gidilen yonun bulunması)

location\_y\_last: hareket etmeden önceki bulunduğu y koordinatı (gidilen yonun bulunması)

radius: yarıçapı

speed: hızı

defeated: 1 ise yaşıyor. 0 ise yenmiş ve yok edilmiş

aktif: bolunmeden sonra 1 olur ve birleşme olmaz. 0 olduðu zaman birleşme gerçekleşir.

aktif\_counter: sayaç yaklaşık 25 saniye sonra aktifi 0 yaparak birleşmeye izin verir.

move\_way: Bolunmede ilerleme yönü

move\_way\_counter: Bölünmede ilerleme süresi (yaklaşık 1 saniye)

**circle\_t(YUVARLAK CİSİMLER):**

location\_x: bulundugu x koordinatı

location\_y: bulunduğu y koordinatı

radius: yarıçapı

red: kırmızı tonu

green: yeşil tonu

blue: mavi tonu

**color\_feature\_t(RENK TONLARI):**

red: KIRMIZI TONU

green: YEŞIL TONU

blue: MAVI TONU

**Fonksiyonlar:**

**void Baslat();**

Allegro başlangıç kodlarını içeren fonksiyon.

**void Bitir();**

Allegro Bitirme Kodlarını içeren fonksiyon.

**void increment\_speed\_counter();**

Allegro için gerekli kod.

**void blue\_mouse\_icon(location\_t \*sprite, player\_feature\_t player);**

Eğer oyuncu kontrol aracı olarak mouse’yi seçerse ekranda mavi renkli mouse ikonu gözükür. Bu ikon ile oyuncu yuvarlağın hangi yöne gittiğini değiştirebilir.

**void camera\_move (player\_feature\_t player, division\_t splits[SIZE\_ARRAY], location\_t \*camera, int number\_of\_split);**

Kamera fonksiyonu oyuncu odaklı bir fonksiyon. Oyuncunun en büyük parçasını takibe alır ve o nereye giderse ekranı oraya kaydırır.

**void choose\_thema(BITMAP \*bmp,BITMAP \*sprite,double still\_run, int \*choose\_tema, color\_feature\_t \*yem\_color, color\_feature\_t \*player\_color);**

Bu ekrandaoyuncu oynamak istediği karakteri ve görmek isteği arka planı yani temayı seçebilir. 3 tema ve 2 karakter bulunmaktadır.

**double screen\_first(double first\_screen,BITMAP \*bmp,BITMAP \*sprite);**

Başlangıç ekranında 2 adet seçim bulunmakta. Başlata tıklandığında kontrol seçim ekranına gider. Çıkışa basıldığında oyundan çıkar.

**double player\_control(BITMAP \*bmp,BITMAP \*sprite, double still\_run);**

Eğer başlat tuşuna basılırsa kontrol cihazı seçim ekranı gelir. 3 seçenek bulunmakta. Klavye, mouse ve gamepad. Oyuncu hangisini seçerse oyun içinde o aktif hale gelir.

**void game(double still\_run,BITMAP \*bmp,BITMAP \*arka\_fon,BITMAP \*sprite\_game,int choose\_control, color\_feature\_t yem\_color, color\_feature\_t player\_color);**

Oyunun gerçekleştiği fonksiyondur. Diğer fonksiyonlar bu fonksiyon içerisinde yer alır. Ve sırayla çağırır. Esc tuşuna basıldığında ya da oyuncu ölünce duraklatma ekranı gelir. Bu duraklatma ekranında 3 seçenek bulunmakta. Eğer oyuncu yaşıyorsa devam seçeneği, oyuna yeniden başlamak için yeniden başlat seçeneği, oyundan çıkmak için çıkış seçeneği.

**void draw\_circles(BITMAP \*bmp, player\_feature\_t player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split, color\_feature\_t yem\_color, color\_feature\_t player\_color, location\_t yem[SIZE\_ARRAY], circle\_t green\_things[SIZE\_ARRAY], BITMAP \*green\_things\_pic, FONT \*font1);**

Oyuncunun, yemlerin ve botların şekilleri yuvarlak. Yeşil alanlar şekli elips. Oyuncu ve botlar arasında sıralama yapılır ve en büyük olan en son çizilir. Böylece büyük olan küçük olanın üstünde gözükür.

**void eat\_bot(player\_feature\_t \*player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Bu fonksiyon botun diğer botlar ve oyuncu tarafından yenilmesini sağlar. Eğer botu, kendinden büyük olan bot ya da kendinden büyük olan oyuncu kapsıyorsa, o botun büyüklüğünün belli bir oranı yiyene aktarılır ve yenilen bot için yeni rastgele koordinatlar belirlenir. Bot o koordinatlarda yeniden oyuna katılır.

**void eat\_player(player\_feature\_t \*player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Bu fonksiyon oyuncunun diğer botlar tarafından yenilmesini sağlar. Eğer oyuncuyu, kendinden büyük olan bot kapsıyorsa, o oyuncunun büyüklüğünün belli bir oranı yiyene aktarılır ve playerın yaşam değeri 0’ a eşitlenir.

**void eat\_split(player\_feature\_t \*player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Bu fonksiyon splitin diğer botlar tarafından yenilmesini sağlar. Eğer spliti, kendinden büyük olan bot kapsıyorsa, o splitin büyüklüğünün belli bir oranı yiyene aktarılır ve splitin yaşam değeri 0’ a eşitlenir.

**void eat\_yem(player\_feature\_t \*player, location\_t yem[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split, bots\_feature\_t bots[SIZE\_ARRAY]);**

Bu fonksiyon yemlerin diğer botlar ve oyuncu tarafından yenilmesini sağlar. Eğer yem, kendinden büyük olan bot ya da kendinden büyük olan oyuncu kapsıyorsa, o yemin büyüklüğünün belli bir oranı yiyene aktarılır ve yenilen yem için yeni rastgele koordinatlar belirlenir. yem o koordinatlarda yeniden oyuna katılır.

**void random\_yem\_location(location\_t yem[SIZE\_ARRAY]);**

Yemler için rastgele koordinatlar

**void random\_bot\_location(bots\_feature\_t bots[SIZE\_ARRAY], int target);**

Botlar için rastgele koordinatlar

**void random\_bot\_target\_location(bots\_feature\_t bots[SIZE\_ARRAY], int target);**

Botlar için rastgele hedef koordinatlar

**void random\_color\_bot(bots\_feature\_t bots[SIZE\_ARRAY]);**

Botlar için rastgele renk karışımı

**void random\_green\_things\_location(circle\_t green\_things[SIZE\_ARRAY], int target);**

Botlar için rastgele koordinatlar

**void move\_player(player\_feature\_t \*player, double choose\_control,division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Oyuncunun hareket etmesini sağlayan fonksiyon. Oyuncu bölündükten yaklaşık 25 saniye sonraya kadar, bölündüğü parça ile birleşemez. Bu süre geçtikten sonra birleşme gerçekleşir. Bu 25 saniye boyuncu birleşmeyi engelleyen, bu fonksiyondur.

**void move\_bot(bots\_feature\_t bots[SIZE\_ARRAY], player\_feature\_t player, division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Botların hareket etmesini sağlayan fonksiyon. Bot stratejisi bu fonksiyon içinde meydana gelir. Botlar savunma odaklı. Öncelikle Eğer oyuncu büyükseve yakın bir mesafedeyseoyuncudan kaçar. Eğer diğer botlardan küçükse ve yakın bir mesafedeyse diğer botlardan kaçar. Daha sonra eğer oyuncu küçükse ve yakın bir mesafedeyse oyuncuya doğru gider. Eğer yakın mesafedeki bot küçükse bota doğru gider eğer bunlardan hiçbiri gerçekleşmezse rastgele bir hedef belirler ve oraya doğru hareket eder.

**void move\_splits(player\_feature\_t \*player, double choose\_control,division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Splitin hareket etmesini sağlayan fonksiyon. Split bölündükten yaklaşık 25 saniye sonraya kadar, bölündüğü parça ile birleşemez. Bu süre geçtikten sonra birleşme gerçekleşir. Bu 25 saniye boyuncu birleşmeyi engelleyen, bu fonksiyondur.

**void speed\_update(player\_feature\_t \*player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Oyuncunun, splitlerin ve botların hızları güncellenir. Hız; ilk hız bölü yarıçap şeklinde hesaplanır. Yuvarlağın yarıçapı ne kadar artarsa o kadar yavaşlar.

**void division\_space(player\_feature\_t \*player, double choose\_control, division\_t splits[SIZE\_ARRAY], int \*number\_of\_split);**

Oyuncu ya da split belli bir boyuttan büyükse, 2 parçaya ayrılmasına izin verilir. Bu durumu kontrol eden fonksiyondur.

**void division\_space\_split(division\_t splits[SIZE\_ARRAY], int number\_of\_split, int sayac\_split);**

Spliti 2 parçaya ayıran fonksiyon.

**void division\_space\_player(player\_feature\_t \*player, division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Oyuncuyu 2 parçaya ayıran fonksiyon

**void division\_space\_move\_way(player\_feature\_t \*player, division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

Bölünme için 8 farklı yön bulunmakta 1 numara batı olmak kaydıyla, 3 numara kuzeyi, 5 numara doğuyu, 7 numara ise güneyi gösterir. 2 numara kuzeybatı, 4 numara kuzeydoğu, 6 numara güneydoğu, 8 numara ise güneybatı yönünü temsil eder. Kullanıcı hangi yönde hareket ediyorsa bölünme o yöne doğru gerçekleşir.

**void division\_space\_merger\_again(player\_feature\_t \*player, division\_t splits[SIZE\_ARRAY], int number\_of\_split);**

sayaç 1000 rakamına yaklaşık 25 saniye sonra ulaşır. Ulaştıktan sonra oyuncunun ayrılan parçası birleşme yeteneği kazanır.

**void division\_green\_things(player\_feature\_t \*player, bots\_feature\_t bots[SIZE\_ARRAY], division\_t splits[SIZE\_ARRAY], int \*number\_of\_split, circle\_t green\_things[SIZE\_ARRAY]);**

Yeşil alanlar üstünde belli bir büyüklükte olan split ya da oyuncu parçalara ayrılır. Bu fonksiyon bunun kontrolünü yapar.

**void division\_green\_things\_player(player\_feature\_t \*player, division\_t splits[SIZE\_ARRAY], int \*number\_of\_split, circle\_t green\_things[SIZE\_ARRAY], int sayac\_division);**

Oyuncuyu parçalara ayıran fonksiyon.

**void division\_green\_things\_split(division\_t splits[SIZE\_ARRAY], int \*number\_of\_split, circle\_t green\_things[SIZE\_ARRAY], int sayac\_division, int target\_split);**

Spliti parçalara ayıran fonksiyon.