MUSTAFA AKILLI
131044017

AGARI YO v1.2.1 PROJECT REPORT

Definition of the game:

Oyunun adı Agarl 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 olduğu 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 ekranda oyuncu 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 büyükse ve yakın bir mesafedeyse oyuncudan 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.