TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİLGİSAYAR BİLİMLERİNE GİRİŞ DÖNEM PROJESİ
PANEL DE PON

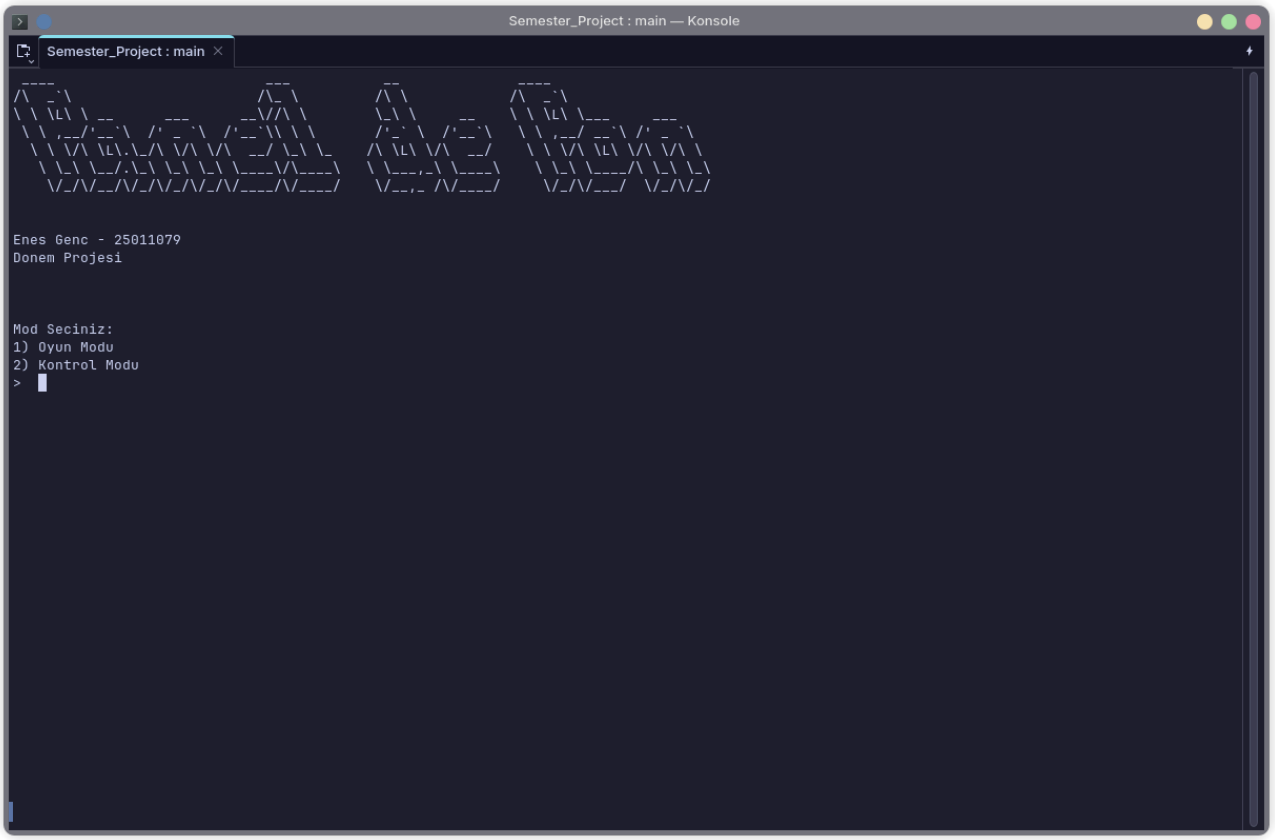
ENES GENÇ
25011079

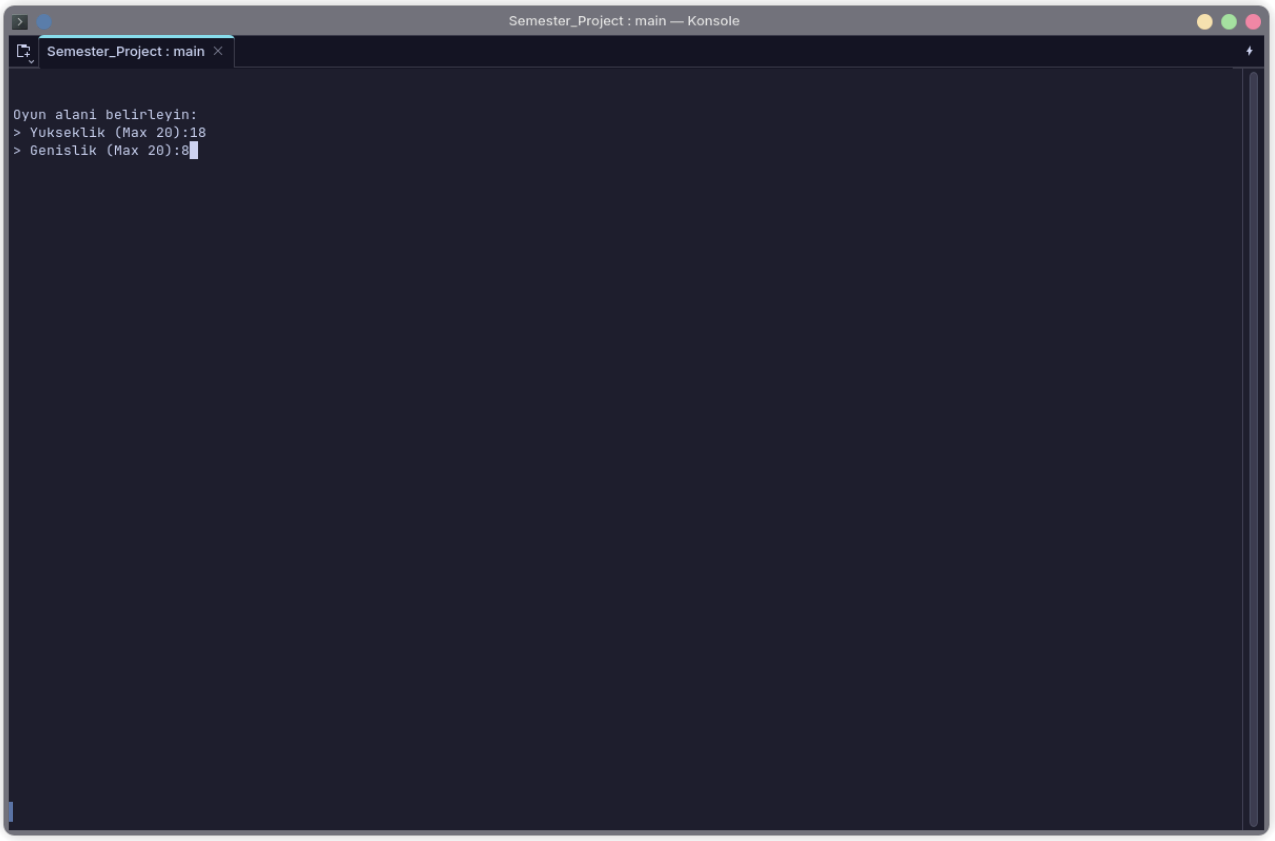
ÖĞRETİM GÖREVLİSİ
Doç.Dr. HAFİZA İREM TÜRKMEN ÇİLİNGİR

İSTANBUL
2026

1. Oyunun Oynanışı

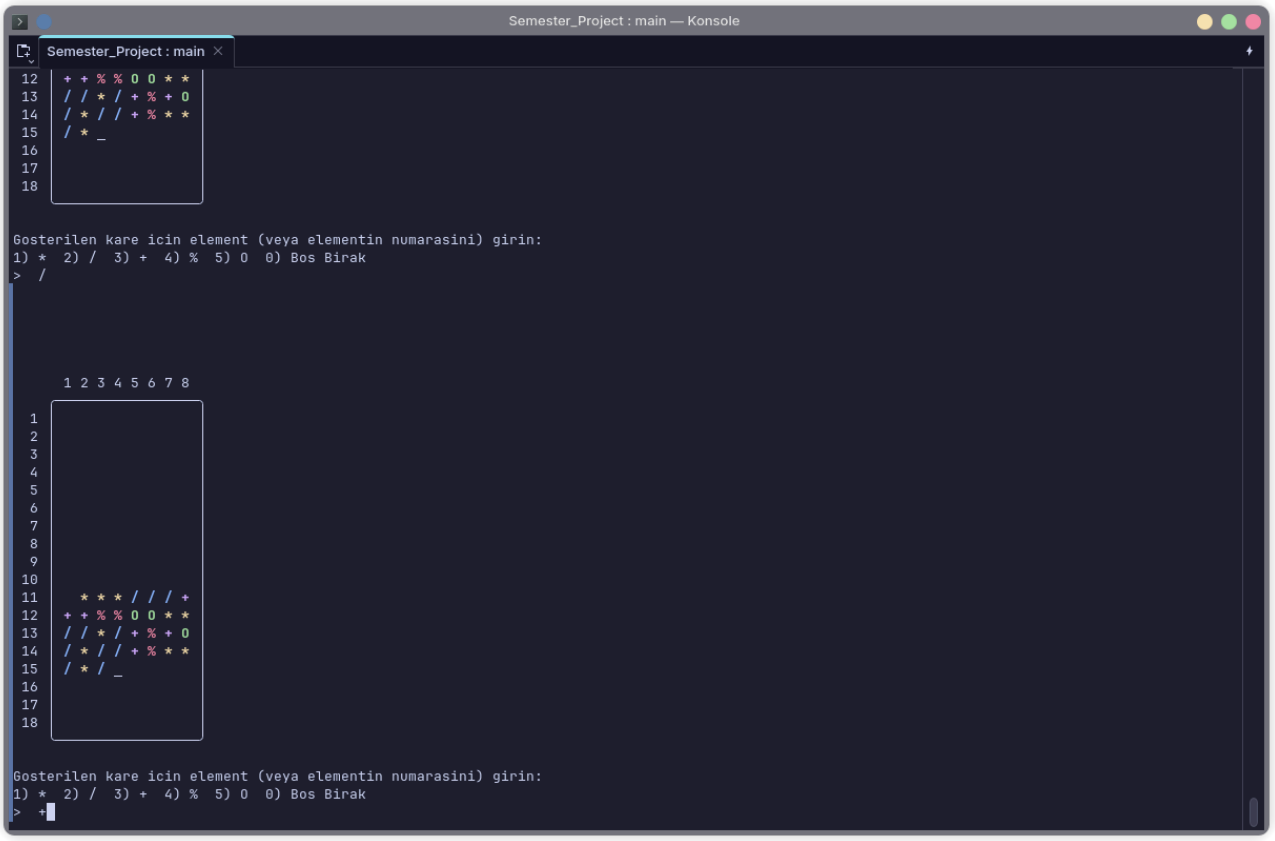
Açılışta kullanıcı mod seçer. Mod seçiminin ardından oyun alanının büyüklüğünü belirler.



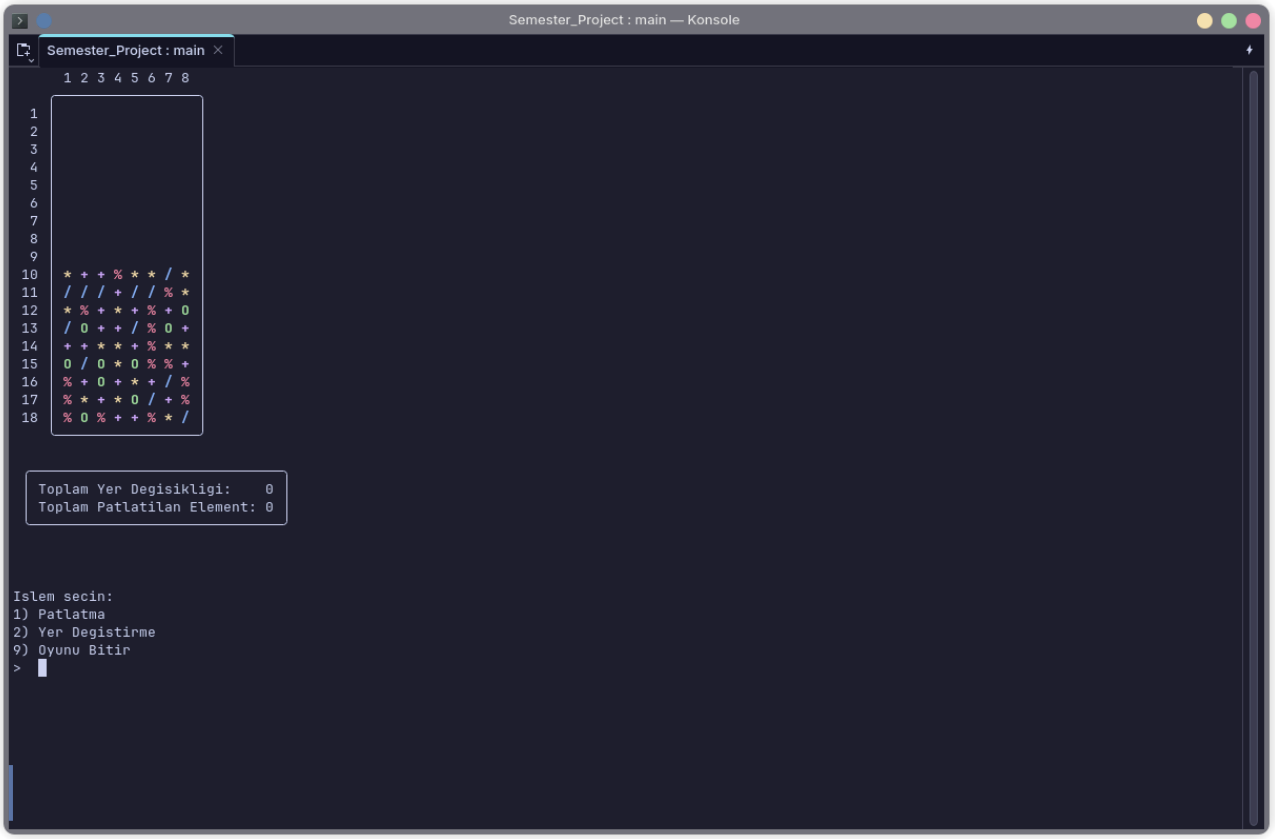


Oyun alanı büyüklüğü belirlendikten sonra oyun alanı doldurulur. Oyun modu seçilmişse oyun alanı rastgele doldurulur. Kontrol modu seçilmişse oyun alanı kullanıcı tarafından doldurulur. Ayrıca kontrol modundayken ekran asla tamamen temizlenmez, önceki durumların görünmesi sağlanır.

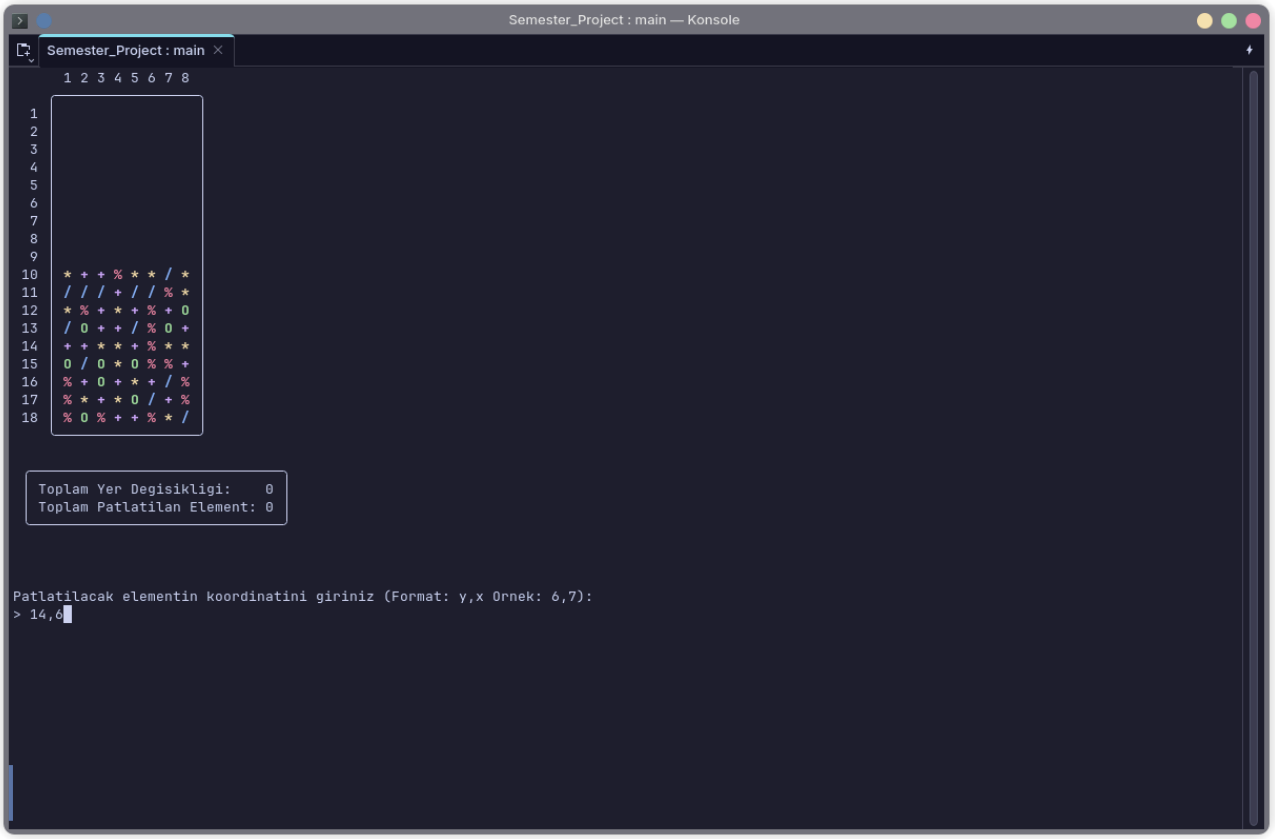
Kontrol modunda oyun alanının doldurulusu:



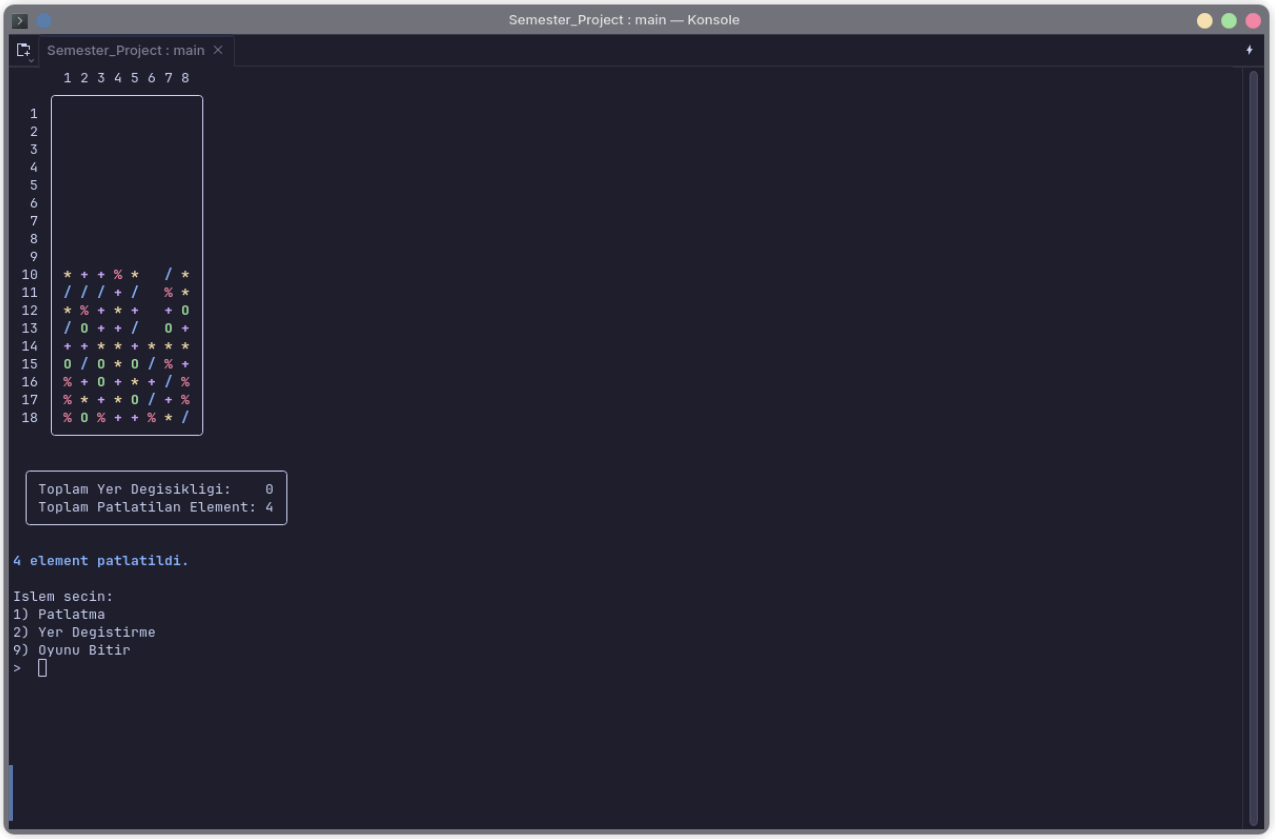
Oyun alanı doldurulduktan sonra oyun başlar ve kullanıcıya yapacağı işlem sorulur.



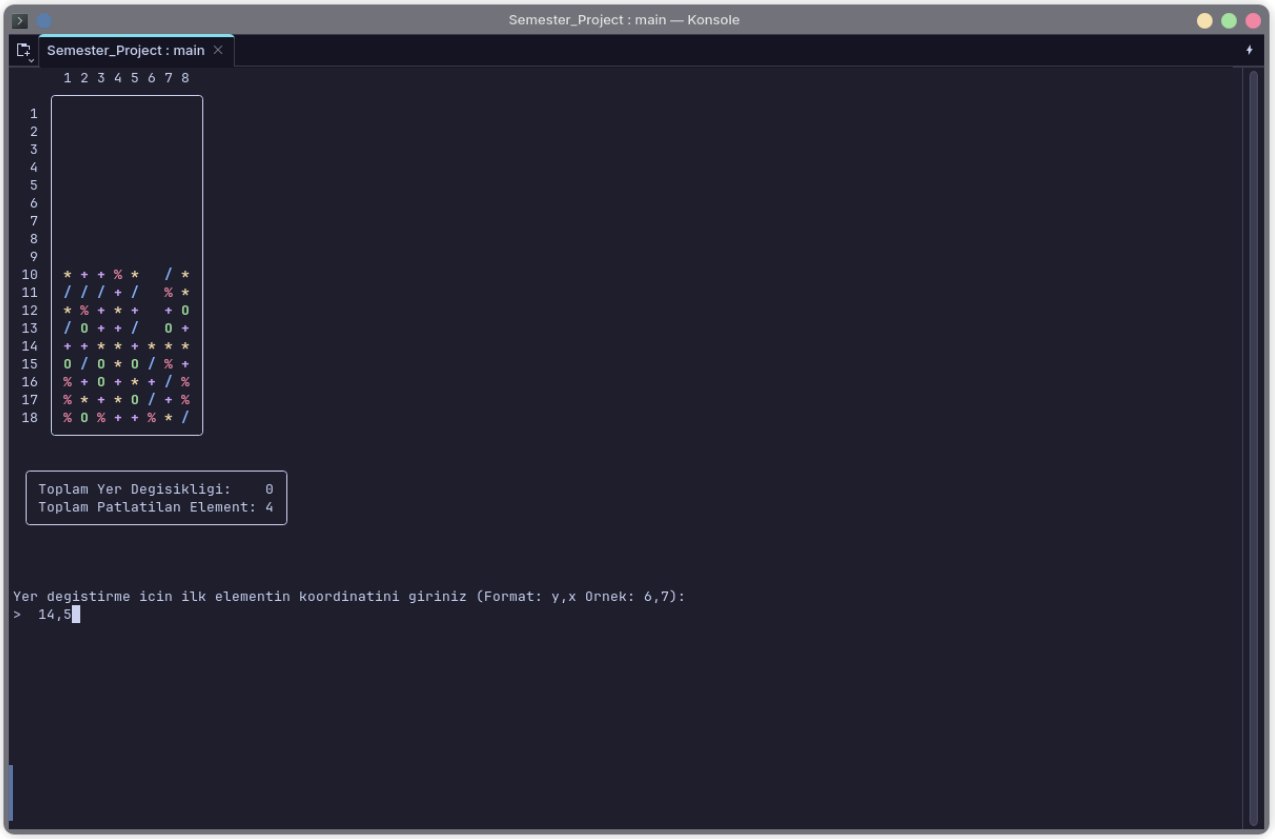
Patlatma seçilmesi durumunda kullanıcıdan patlatmak istediği elementin koordinatı istenir.



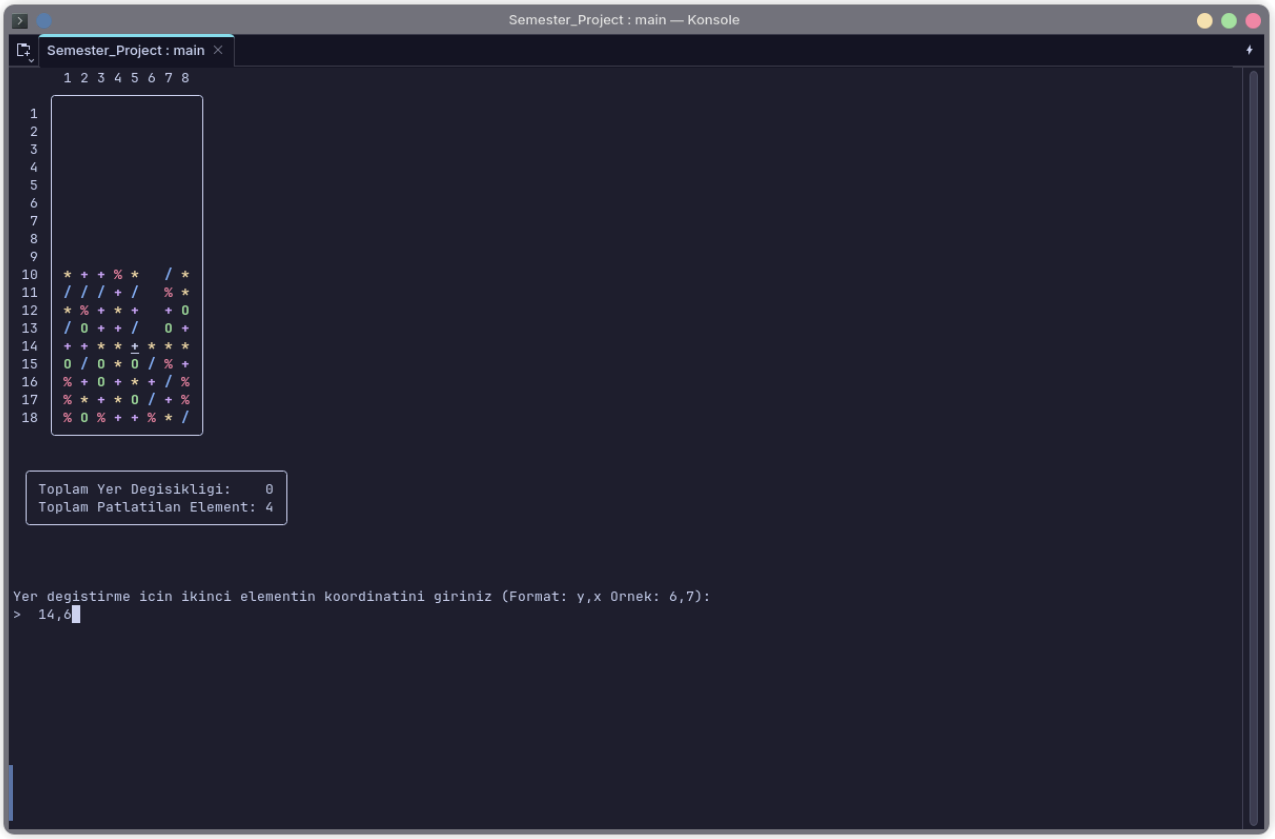
Seçilen element eğer kendisiyle aynı 3 veya daha fazla element ile yatay veya dikey bir grup oluşturuyorsa (ikisinin olduğu durumda daha uzun olan) bu grup patlatılır ve oyun alanından silinir. Eğer üstlerinde elementler varsa bu elementler oluşan boşluğa düşer.



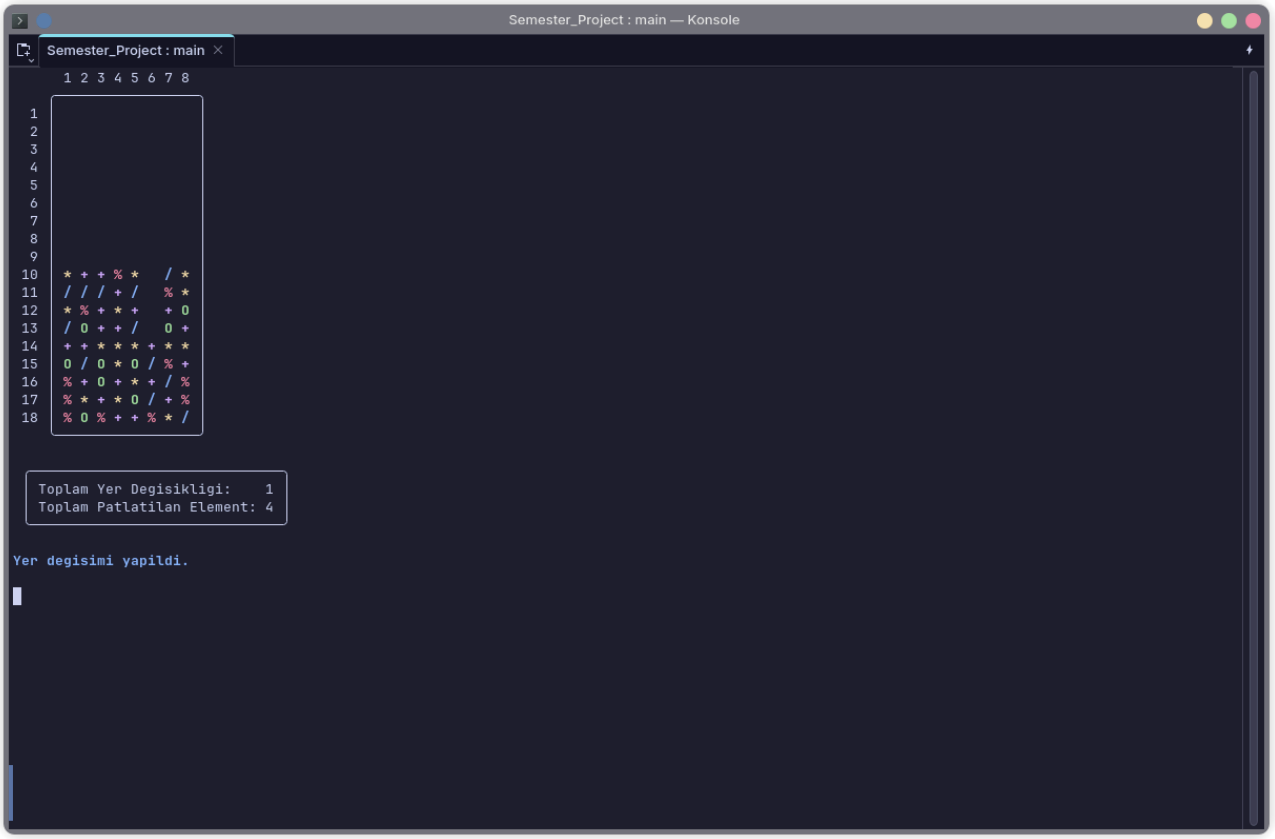
Yer değiştirme seçilmesi durumunda kullanıcıdan yer değiştirmek istediği elemanlardan ilkinin koordinatı istenir.



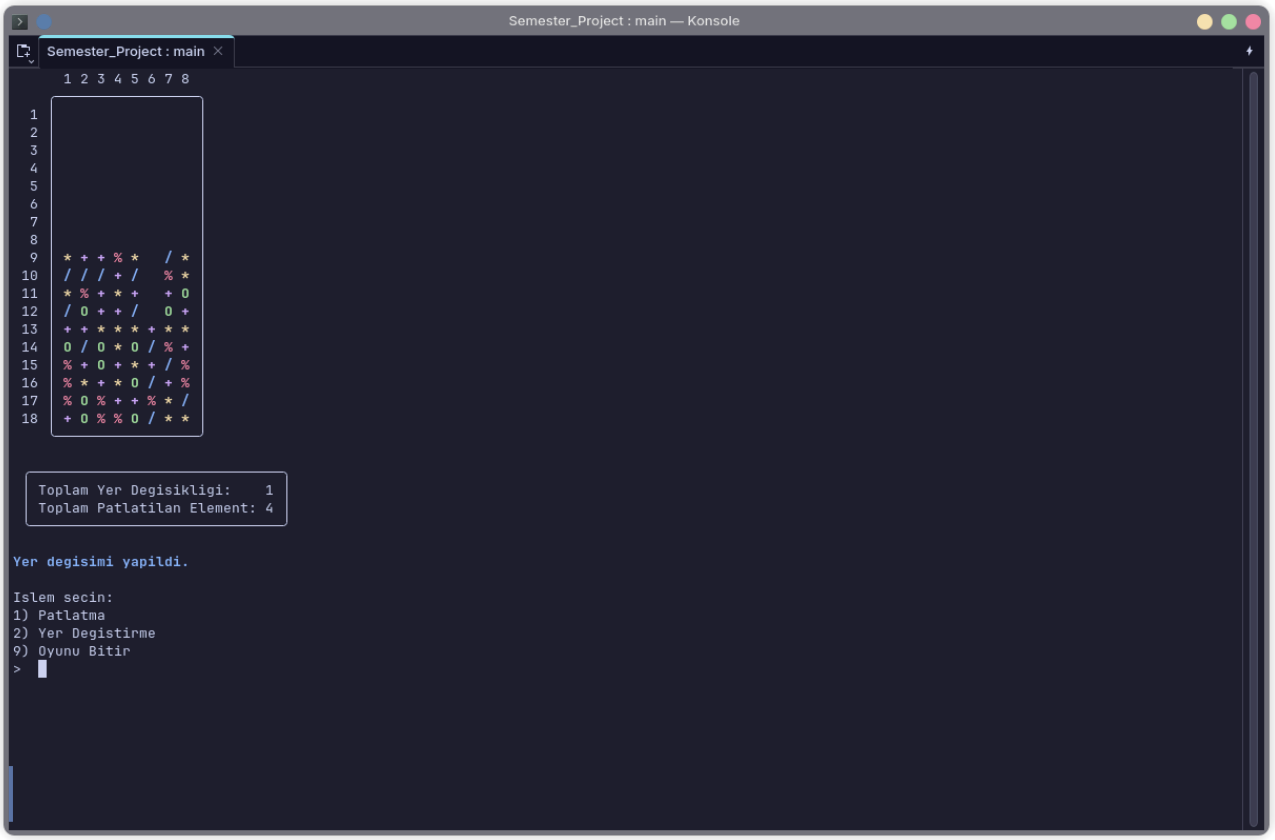
İlk elementin koordinatı girildikten sonra girilen element oyun alanında beyaz olarak ve altı çizili şekilde gösterilir, ikinci elementin koordinatı istenilir. Panel de Pon oyununda sadece yan yana olan elementler yer değiştirebileceği için kullanıcının y koordinatını girmesine izin verilmez (girili gelir), kullanıcı sadece x koordinatını girebilir.



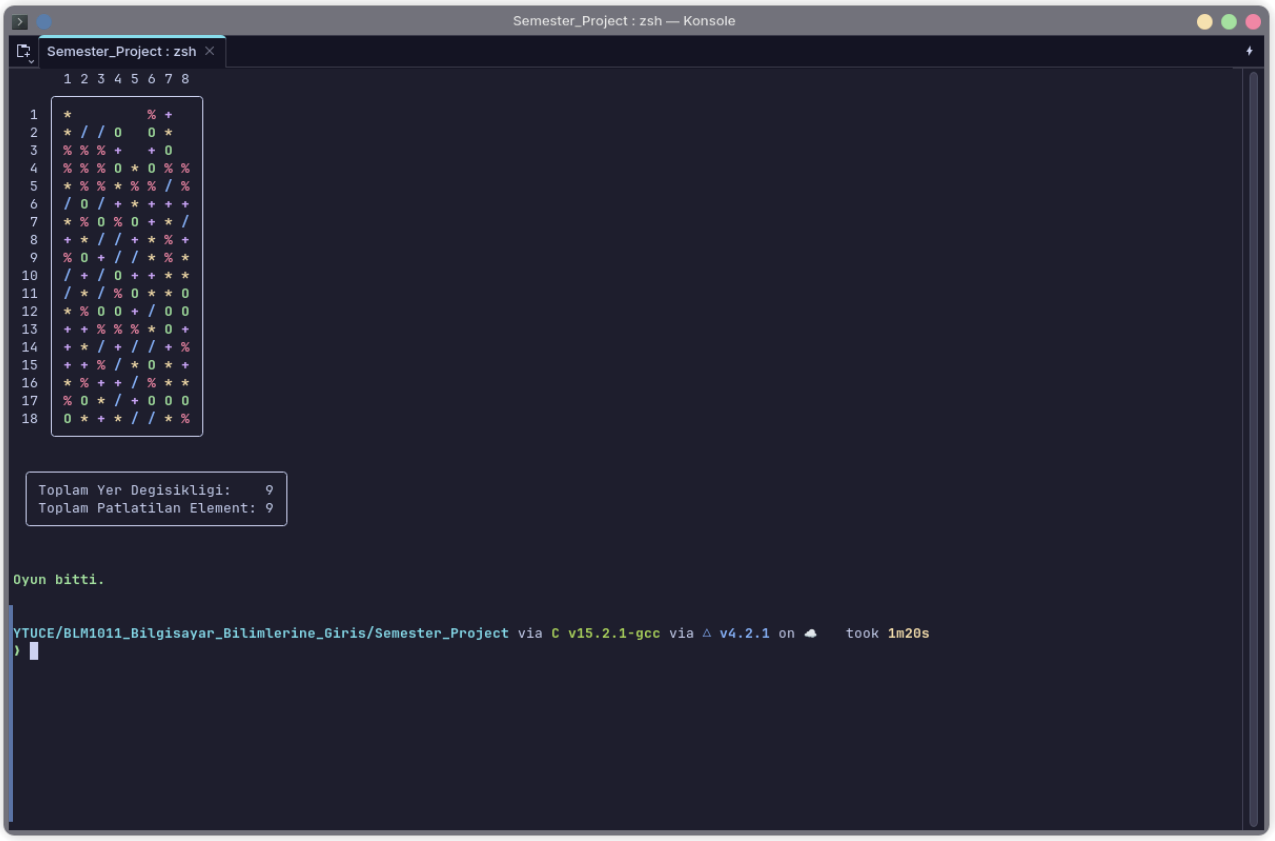
İkinci koordinat onaylandıktan sonra yer değiştirme yapılır.



Kullanıcıya yer değişimi yapılmış durum gösterilir, daha sonra oyun alanına yeni bir satır eklenir ve kullanıcıya yeni durum gösterilir.



Oyun elementlerden herhangi biri en üst satıra gelene kadar devam eder. Kullanıcı işlem seçim ekranında oyunu daha önce bitirebilir.



Bazı elementler ilk satıra çıktığı için oyun bitti.

2. Algoritma ve Kod Yapısı

Sabitler:

```
13      #ifdef _WIN32
14      #define BORDER_H "-"
15      #define BORDER_V "|"
16      #define BORDER_TL "/"
17      #define BORDER_TR "\\"
18      #define BORDER_BL "\\"
19      #define BORDER_BR "/"
20      #else  #ifdef _WIN32
21      #define BORDER_H "─"
22      #define BORDER_V "│"
23      #define BORDER_TL "┌"
24      #define BORDER_TR "┐"
25      #define BORDER_BL "└"
26      #define BORDER_BR "┘"
27      #endif  #ifdef _WIN32 #else
```

```
34      #define RED "\033[1;31m"
35      #define GREEN "\033[1;32m"
36      #define YELLOW "\033[1;33m"
37      #define BLUE "\033[1;34m"
38      #define PURPLE "\033[1;35m"
39      #define HIGHLIGHT "\033[1;37m\033[4;37m"
40      #define RESET "\033[0m"
41
42      #define CHAR1 '*'
43      #define CHAR2 '/'
44      #define CHAR3 '+'
45      #define CHAR4 '%'
46      #define CHAR5 'O'
47
48      #define BLOCK1 YELLOW "*" RESET
49      #define BLOCK2 BLUE   "/" RESET
50      #define BLOCK3 PURPLE "+" RESET
51      #define BLOCK4 RED    "%" RESET
52      #define BLOCK5 GREEN  "O" RESET
```

Arayüzün daha estetik görünmesi için Windows için ASCII, diğer platformlar için Unicode çerçeve karakterleri kullanılmıştır ve bunların her biri BORDER önekiyle tanımlanmıştır.
Kodun okunabilirliğini artırmak için elementler, renk kodları ve elementlerin renkli halleri de burada tanımlanmıştır.

Oyunun Yazdırılması:

```
52      void clear(int fullClear) {
53          if (!fullClear) {
54              int i;
55              for (i = 0; i < 5; i++) {
56                  printf( format: "\n");
57              }
58              return;
59          }
60
61      #ifdef _WIN32
62          system("cls");
63      #else  #ifdef _WIN32
64          system( command: "clear");
65      #endif  #ifdef _WIN32 #else
66      }
```

```
118     int calculateDigits(int num) {
119         int i;
120         int digits = 1;
121
122         for (i = num; i > 10; i /= 10) {
123             digits++;
124         }
125
126         return digits;
127     }
128
129     void drawWhitespace(int n) {
130         int i;
131         for (i = 0; i < n; i++) printf( format: " ");
132     }
```

“clear” fonksiyonu ekranın gerektiğinde temizlenmesi için kullanılmaktadır. fullClear parametresi kontrol modunda ekranın tamamen temizlenmesi istenmediğinde 0 verilir, fullClear 0 iken ekranı tamamen temizlemek yerine ekranı kaydırır.
“calculateDigits” fonksiyonu verilen sayının kaç hane olduğunu hesaplar. “drawWhitespace” fonksiyonu ekranda istenilen sayıda boşluk bırakır. Bu iki yardımcı fonksiyon “drawStats” fonksiyonunda kullanılır.

```
134  void drawStats(int swp, int exp) {
135      int i;
136      int flLength = 26 + calculateDigits(swp);
137      int llLength = 29 + calculateDigits(exp);
138      int maxLength = flLength > llLength ? flLength : llLength;
139
140      printf( format: " %s", BORDER_TL);
141      for (i = 0; i < maxLength; i++) printf( format: "%s", BORDER_H);
142      printf( format: "%s\n", BORDER_TR);
143
144      printf( format: " %s Toplam Yer Degisikligi:", BORDER_V);
145      drawWhitespace( n: maxLength - flLength + 1);
146      printf( format: "%d %s\n", swp, BORDER_V);
147
148      printf( format: " %s Toplam Patlatilan Element:", BORDER_V);
149      drawWhitespace( n: maxLength - llLength + 1);
150      printf( format: "%d %s\n", exp, BORDER_V);
151
152      printf( format: " %s", BORDER_BL);
153      for (i = 0; i < maxLength; i++) printf( format: "%s", BORDER_H);
154      printf( format: "%s\n", BORDER_BR);
155  }
```

“drawStats” fonksiyonu oyuncunun istatistiklerini yazdırmak için kullanılır. Başlangıçta çerçevenin doğru yazdırılabilmesi için satırların uzunlukları hesaplanır. Çerçeve en uzun satıra göre hesaplanır. Yazdırma mantığı:

- Çerçevenin üst kenarı yazdırılır.
- İstatistikler yazdırılır, istatistiklerin sağında “drawWhitespace” fonksiyonu kullanılarak boşluk bırakılır, ayrıca başa ve sona dikey çerçeve karakterleri yazdırılır.
- Çerçevenin alt kenarı yazdırılır.

```c
void drawBoard(int board[20][20], int boardX, int boardY, int highlight, int highlightX, int highlightY) {
    int x, y;

    for (y = -2; y <= boardY; y++) {
        int coordinateLine = y == -2;
        int firstLine = y == -1;
        int lastLine = y == boardY;

        for (x = -1; x <= boardX; x++) {
            int firstChar = x == -1;
            int lastChar = x == boardX;

            int ch = board[y][x];
            char* block;

            if (ch == CHAR1) block = BLOCK1;
            else if (ch == CHAR2) block = BLOCK2;
            else if (ch == CHAR3) block = BLOCK3;
            else if (ch == CHAR4) block = BLOCK4;
            else if (ch == CHAR5) block = BLOCK5;
            else block = " ";

            if (!ch) ch = ' ';

            if (coordinateLine && x == -1) {
                x = 0;
                printf(format: "      %d", x + 1);
            }
            else if (coordinateLine) { if (x < boardX) printf(format: " %d", (x + 1) % 10); }
            else if (firstLine && firstChar) printf(format: "    %s", BORDER_TL);
            else if (firstLine && lastChar) printf(format: "%s%s", BORDER_H, BORDER_TR);
            else if (lastLine && firstChar) printf(format: "    %s", BORDER_BL);
            else if (lastLine && lastChar) printf(format: "%s%s", BORDER_H, BORDER_BR);
            else if (firstLine || lastLine) printf(format: "%s%s", BORDER_H, BORDER_H);
            else if (firstChar) printf(format: " %2d %s", y + 1, BORDER_V);
            else if (lastChar) printf(format: " %s", BORDER_V);
            else if (x == highlightX && y == highlightY && highlight) printf(format: " %s%c%s", HIGHLIGHT, ch, RESET);
            else printf(format: " %s", block);
        }

        printf(format: "\n");
    }
}
```

“drawBoard” fonksiyonu oyun alanının yazdırılmasını sağlar. Parametre olarak oyun alanı matrisini ve büyüklüğünü, vurgulamanın aktiflik durumunu, vurgulanacak bir element varsa bu elementin koordinatlarını alır.

“for” döngüsü aracılığıyla oyun alanı yazdırılır. Döngü 0’dan değil, dikey eksen için -2 ve yatay eksen için -1’den başlatılır. y = -2 durumunda (en üst satır) yatay eksen koordinatları yazdırılır. Yatay eksen koordinatları sütun hizalamasının bozulmaması amacıyla tek hane olarak yazdırılır, 2 haneli koordinatların sadece birler basamağı yazdırılır (koordinat mod 10). x ve y’nin değerlerine göre ilk/son satır ve ilk/son karakter kontrolleri yapılır (ilk satır ve ilk karakter kontrollerinde koordinatlar ihmal edilir) ve bunlara göre çerçeve karakteri seçilir. Oyun elementleri bu koşulların dışında kaldığı için çerçeve karakteri yazdırılmaz ve sonraki aşamaya geçilir.

Oyun elementleri yazdırılırken elemente denk gelen “blok” seçilir. Bloklar elementlerin renk kodlarını içeren halleridir, eğer yazdırılacak karede herhangi bir element yok ise blok olarak boşluk karakteri seçilir. Vurgulama aktif ise elementin vurgulanması gerekip gerekmediği koordinatları karşılaştırılarak kontrol edilir. Vurgulanacak element seçilen blok kullanılmadan HIGHLIGHT renk kodu ile yazdırılır, geri kalan elementler ise seçilen blok ile yazdırılır. Okunabilirliği arttırmak amacıyla aynı satırda her element arasında boşluk bırakılır.

Oyun Alanının Doldurulması:

```
236     void randomFillBoard(int board[20][20], int x, int y, int chars[5]) {
237         int i, j;
238
239         for (i = y / 2 + y % 2; i < y; i++) {
240             for (j = 0; j < x; j++) {
241                 board[i][j] = chars[rand() % 5];
242             }
243         }
244     }
245
```

“randomFillBoard” fonksiyonu oyun modunda oyun alanının rastgele doldurulması için kullanılır. Oyun alanını, boyutlarını ve oyun elementlerini parametre olarak alır. Aldığı “board” parametresi asıl oyun alanının referansı olduğu için doğrudan ana oyun alanı üzerinde değişiklik yapar. Döngü oyun alanının yarısından başlar ve her kareyi verilen elementlerden rastgele biriyle doldurur. Oyun alanının yüksekliğinin tek sayı olduğu durumlarda doldurulan satır sayısı oyun alanı yüksekliğinin yarısının aşağı yuvarlanmış halidir.

```
246     void controlModeFillBoard(int board[20][20], int boardX, int boardY) {
247         int i, j;
248         int invalid = 0;
249
250         for (i = 0; i < boardY; i++) {
251             for (j = 0; j < boardX; j++) {
252                 clear(0);
253                 drawBoard(board, boardX, boardY,  highlight: 1,  highlightX: j,  highlightY: i);
254                 printf( format: "\n");
255                 if (invalid) {
256                     printf( format: "%sHatalı giriş yaptınız.%s\n", RED, RESET);
257                     invalid = 0;
258                 } else {
259                     printf( format: "\n");
260                 }
261                 printf( format: "Gosterilen kare icin element (veya elementin numarasini) girin:\n");
262                 printf( format: "1) *  2) /  3) +  4) %%  5) 0  0) Bos Birak\n");
263                 printf( format: "> ");
264
265                 char input;
266                 scanf( format: " %c", &input);
267
268                 if (input == '0') board[i][j] = 0;
269                 else if (input == CHAR1 || input == '1') board[i][j] = CHAR1;
270                 else if (input == CHAR2 || input == '2') board[i][j] = CHAR2;
271                 else if (input == CHAR3 || input == '3') board[i][j] = CHAR3;
272                 else if (input == CHAR4 || input == '4') board[i][j] = CHAR4;
273                 else if (input == CHAR5 || input == '5') board[i][j] = CHAR5;
274                 else {
275                     j--;
276                     invalid = 1;
277                 }
278             }
279         }
280
281         clear(0);
282         drawBoard(board, boardX, boardY,  highlight: 0,  highlightX: 0,  highlightY: 0);
283         wait( seconds: 2);
284     }
```

“controlModeFillBoard” fonksiyonu oyun alanının kontrol modunda doldurulması için kullanılır. Aldığı parametreler “randomFillBoard” fonksiyonu ile aynıdır. Bu fonksiyon “drawBoard” fonksiyonunun vurgulama özelliğinden faydalanarak oyun alanındaki her kareyi sırayla vurgular ve kullanıcıdan vurgulanan alan için element belirlemesini ister. Geçersiz girdi yapıldığı durumlarda tekrar girdi istenir. Kullanım kolaylığı için elementlerin sembolleri ile beraber eşleştirilmiş numara ile de girilmesine izin verilir.

Patlatma Mekaniği:

```
297  int explode(int board[20][20], int boardX, int boardY, int x, int y) {
298      int i, j;
299      int xs, xe, ys, ye;
300
301      int ch = board[y][x];
302
303      for (i = x; i >= 0 && board[y][i] == ch; i--) {
304      }
305      xs = i + 1;
306      for (i = x; i < boardX && board[y][i] == ch; i++) {
307      }
308      xe = i - 1;
309      for (i = y; i >= 0 && board[i][x] == ch; i--) {
310      }
311      ys = i + 1;
312      for (i = y; i < boardY && board[i][x] == ch; i++) {
313      }
314      ye = i - 1;
315
316      int xle = xe - xs + 1;
317      int yle = ye - ys + 1;
318      int sideways = 0;
319
320      if (xle < 3 && yle < 3) {
321          return 0;
322      }
323
324      if (xle >= 3 && xle >= yle) {
325          sideways = 1;
326      }
```

Patlatma için “explode” fonksiyonu kullanılır. Parametre olarak oyun alanını, alanın büyüklüğünü ve patlatma için seçilen elementin koordinatlarını alır. İşlem sonunda patlatılan element sayısını döndürür.

1. Seçilen elementten başlayarak tüm yönlerde farklı türde bir elemente denk gelene kadar arama yapılır. Arama yapılan yöne göre farklı türdeki elementin indisinin bir eksiği/fazlası o eksende başlangıç/bitiş koordinatı seçilir.
2. Başlangıç ve bitiş koordinatlarının farkları ile iki eksende de patlatılabilecek eleman sayısı bulunur. İki eksende de patlatılabilecek eleman sayısı 3’ten azsa patlatma yapılmaz ve 0 döndürülür.
3. İki eksenden en az birinde 3’ten fazla eleman patlatılabilecekse, X ekseninde patlatma yapılıp yapılamayacağı ve X ekseninde patlatma yapılabilecek eleman sayısının Y ekseninden büyük olup olmadığına bakılır. Koşul sağlanırsa patlatmanın X ekseninde yapılması için “sideways” değişkeni 1 yapılır. Koşul sağlanmıyorsa “sideways” değişkeninin değeri 0 olarak kalır ve patlatma Y ekseninde yapılır.

```
328         if (sideways) {
329             for (i = xs; i <= xe; i++) {
330                 for (j = y - 1; j >= 0; j--) {
331                     board[j + 1][i] = board[j][i];
332                 }
333                 board[0][i] = 0;
334             }
335 
336             return xle;
337         }
338 
339         for (i = ye; i >= 0; i--) {
340             if (i - yle >= 0) {
341                 board[i][x] = board[i - yle][x];
342             }
343             else {
344                 board[i][x] = 0;
345             }
346         }
347 
348         return yle;
349     }
```

- Yatay patlatmalarda patlatılan gruptaki her elementin üstündeki elementler bir aşağıya kaydırılır. Bu sayede patlatılan elementler oyun alanından kalkmış ve üstündeki elementler yerçekimi etkisiyle aşağıya düşmüş olur. En üstteki element bir alta taşındıktan sonra en üst elementler 0'a eşitlenir. Her ne kadar oyun en üst satırda herhangi bir element varken devam etmiyor olsa bile, bu kontrol asıl oyun akışına aittir, "explode" fonksiyonunun böyle bir varsayım yapmaması gerekmektedir.
- Dikey patlatmalarda kaydırma yapılırken döngü patlatılacak grubun en altındaki elementten başlayarak yukarı doğru yapılır. Buradaki kaydırma, gruptaki her elementin kendisinden grup büyüklüğü kadar yukarısındaki elemente eşitlenmesidir (örneğin grup büyüklüğü 3 ise her element 3 yukarısındaki elemente eşitlenir.). Eşitleme gerçekleşmeden önce sınır kontrolü yapılır, aşağıya kaydırılması planlanan element oyun alanı dışında kalıyor ise patlatılan element 0'a eşitlenir (örneğin grup büyüklüğünün 3 olduğu bir durumda 2. satırdaki elementin 3 yukarısı oyun alanının dışındadır, bu nedenle bu element 0'a eşitlenir.)

Ana Oyun Akışı:

```
362 ▷  int main() {
363        int board[20][20] = {0};
364        int boardX, boardY;
365        int gamemode; // Kontrol = 0; Oyun = 1
366
367        int chars[5] = {CHAR1, CHAR2, CHAR3, CHAR4, CHAR5};
368        int swapCount = 0;
369        int explosionCount = 0;
370
371        int choice = 0;
372        int step = 0;
373
374        char* message;
375        int messageInt;
376        int showInfo = 0;
377        int showError = 0;
378        int showInt = 0;
379
380        int chx1, chy1, chx2, scanres;
381
382        srand( seed: time(NULL));
383
384        clear(1);
385        gamemode = mainMenu( invalidInput: 0);
```

Oyun çalıştırıldığında oyun alanını saklayan matris tanımlanır ve tüm elemanları sıfıra eşitlenir, oyun alanı boyutları da tanımlanır fakat değeri kullanıcıdan alınacağı için değeri tanımlanmaz. Elementlerin saklandığı “chars” dizisi tanımlanır. Bu dizi rastgele element ekleyen fonksiyonlara parametre olarak verilir. “swapCount” ve “explosionCount” değişkenleri ile kullanıcının istatistikleri saklanır.
“choice” ve “step” değişkenleri kullanıcı seçimini ve bu seçim bir işlem başlattıysa işlem aşamasını saklamak için kullanılmaktadır.
“message” değişkeni bilgilendirme ve hata mesajları için kullanılmaktadır, “showInfo” ve “showError” değişkenleriyle mesaj tipi belirlenir, “showInt” ve “messageInt” değişkenleri beraber kullanılarak mesajdan önce sayı gösterilebilir.
“chx1”, “chy1”, “chx2” değişkenleri kullanıcıdan alınan koordinatların geçici olarak saklandığı değişkenlerdir. “scanres” değişkeni “scanf” fonksiyonunun döndürdüğü değeri alır ve buna göre girdi doğrulaması yapılır.
“srand” fonksiyonu ile RNG o anki tarih saat ile seedlenerek RNG’nin her seferinde aynı değerleri oluşturması önlenir.

Bu tanımlamalardan sonra “clear” fonksiyonu kullanılarak ekran temizlenir. Daha sonra kullanıcıdan oyun modunun alınması için “mainMenu” fonksiyonu çağırılır.

```c
int mainMenu(int invalidInput) {
    printf( format: " ____                          ___          __              ____                  \n");
    printf( format: "/\\  _`\\                       /\\_ \\        /\\ \\            /\\  _`\\               \n");
    printf( format: "\\ \\ \\L\\ \\ __      ___      __\\//\\ \\       \\_\\ \\     __    \\ \\ \\L\\ \\___     ___    \n");
    printf( format: " \\ \\ ,__/'__`\\  /' _ `\\  /'__`\\\\ \\ \\      /'_` \\  /'__`\\   \\ \\ ,__/ __`\\ /' _ `\\  \n");
    printf(
        format: "  \\ \\ \\/\\ \\L\\.\\_/\\ \\/\\ \\/\\  __/ \\_\\ \\_    /\\ \\L\\ \\/\\  __/    \\ \\ \\/\\ \\L\\ \\/\\ \\/\\ \\ \n");
    printf(
        format: "   \\ \\_\\ \\__/.\\_\\ \\_\\ \\_\\ \\____\\/\\____\\   \\ \\___,_\\ \\____\\    \\ \\_\\ \\____/\\ \\_\\ \\_\\\n");
    printf( format: "    \\/_/\\/__/\\/_/\\/_/\\/_/\\/____/\\/____/    \\/__,_ /\\/____/     \\/_/\\/___/  \\/_/\\/_/\n");
    printf( format: "\n\n");
    printf( format: "Enes Genc - 25011079\n");
    printf( format: "Donem Projesi\n\n");

    if (invalidInput) {
        printf( format: "%sHatali giris yaptiniz. Lutfen tekrar giris yapin.%s\n\n", RED, RESET);
    }
    else {
        printf( format: "\n\n"); // Hatali giriste kaymayi onleme amacli hata mesajinin oldugu yeri bos birakiyor.
    }

    printf( format: "Mod Seciniz:\n");
    printf( format: "1) Oyun Modu\n");
    printf( format: "2) Kontrol Modu\n");
    printf( format: ">  ");

    int input;
    scanf( format: "%d", &input);

    if (input == 1) {
        return 1;
    }
    if (input == 2) {
        return 0;
    }

    clear(1);
    return mainMenu( invalidInput: 1);
}
```

“mainMenu” fonksiyonu oyunun başlangıcında “ASCII Art” kullanarak ekrana “Panel de Pon” yazar. Daha sonra kullanıcıdan oyun modu seçmesini ister. Fonksiyon ayrıca “invalidInput” adında bir parametre alır. Hatalı girdi girildiği durumlarda fonksiyon kendisini bu parametreyi 1 yaparak çağırır ve bu sayede kullanıcıya hata mesajı gösterir. Doğru girdi girildiğinde ise bu girdiye göre eğer oyun modu seçildiyse 1, kontrol modu seçildiyse 0 döndürür.

Oyun modu belirlendikten sonra program ana akıştan devam eder.

```c
    clear(gamemode);
    setBoardSize(&boardX, &boardY, gamemode,  invalidInput: 0);
```

Oyun modu seçilmişse ekran tamamen temizlenir. Oyun alanının boyutunun ayarlanması için “setBoardSize” fonksiyonu çağırılır. Bu fonksiyona oyun alanı boyutlarını tutan değişkenlerin referansları verilir, bu sayede fonksiyon asıl değişkenlerin değerini düzenler.

```
void setBoardSize(int* x, int* y, int gamemode, int invalidInput) {
    if (invalidInput == 1) {
        printf( format: "%sHatali giris yaptiniz. Lutfen tekrar giris yapin.%s\n\n", RED, RESET);
    }
    else if (invalidInput == 2) {
        printf( format: "%sBelirlediginiz boyutta oynanabilir alan olusmuyor. Lutfen daha buyuk oyun alani belirleyin.%s\n", RED,
               RESET);
    }
    else {
        printf( format: "\n\n");
    }

    printf( format: "Oyun alani belirleyin:\n");
    printf( format: "> Yukseklik (Max 20):");
    scanf( format: "%d", y);

    if (*y ≤ 0 || *y > 20) {
        clear(gamemode);
        return setBoardSize(x, y, gamemode,  invalidInput: 1);
    }

    if (*y % 2 && gamemode) {
        printf( format: "Yukseklik olarak tek sayi girdiniz. Oyun alaninin ilk %d satiri dolu baslayacak.\n", *y / 2);
    }

    printf( format: "> Genislik (Max 20):");
    scanf( format: "%d", x);

    if (*x ≤ 0 || *x > 20) {
        clear(gamemode);
        return setBoardSize(x, y, gamemode,  invalidInput: 1);
    }

    if (*x ≤ 2 || *y ≤ 1) {
        clear(gamemode);
        return setBoardSize(x, y, gamemode,  invalidInput: 2);
    }
}
```

"setBoardSize" fonksiyonu oyun alanı boyutu değişkenlerinin referansları ile beraber iki parametre daha alır. "gamemode" parametresi ekranın temizleneceği durumlarda oyun moduna göre karar verilmesi için alınır. "invalidInput" parametresi ise "mainMenu" fonksiyonundaki gibi hata mesajı gösterilmesi için kullanılır.
Fonksiyon girilen oyun alanının boyutunun maksimum boyutu (20x20) aşıp aşmadığını ve girilen oyun alanının oyunu oynamak için uygun olmadığını kontrol eder. Oyunun oynanabilir olabilmesi için en az 3 sütun ve 2 satır olması gerektiği varsayılmıştır.
Fonksiyon "invalidInput" değişkeninin değerine göre 2 farklı hata mesajı gösterir. Oyun alanı belirlendikten sonra fonksiyon herhangi bir değişken döndürmez, oyun alanı boyutu değişkenlerinin referanslarını aldığı için direkt olarak bu değişkenlerin değerlerini düzenler.

Oyun alanının boyutu belirlendikten sonra program ana akıştan devam eder.

```
389     if (gamemode) {
390         randomFillBoard(board, boardX, boardY, chars);
391     }
392     else {
393         controlModeFillBoard(board, boardX, boardY);
394     }
```

Oyun moduna göre oyun alanı doldurmak için kullanılacak fonksiyon seçilir.

```
396     while (gameCanContinue(board, boardX) && choice ≠ 9) {
```

Oyunun döngüsü iki koşula bağımlıdır.

1. Oyun kurallarına göre oyun devam edebiliyor olmalı.
2. Kullanıcı oyunu kendi isteğiyle erkenden bitirmemiş olmalı. (Bu durumda choice = 9)

```
286  int gameCanContinue(int board[20][20], int x) {
287      int i;
288      int st = 1;
289
290      for (i = 0; i < x; i++) {
291          if (board[0][i] != 0) st = 0;
292      }
293
294      return st;
295  }
```

Oyun kurallarına göre devam edebilirlik kontrolü “gameCanContinue” fonksiyonuyla yapılmaktadır, bu fonksiyon parametre olarak oyun tahtasını ve oyun tahtasının genişliğini almaktadır. Fonksiyon “for” döngüsü aracılığıyla oyun alanının en üst satırındaki tüm kareleri kontrol eder, herhangi bir karede element bulursa 0 döndürür, bulamazsa 1 döndürür. 0 dönmesi durumunda ana akıştaki “while” döngüsünün koşulu sağlanmaz ve oyun bitirilir.

Koşullar sağlandığı durumda oyun akışı “while” döngüsü içinde devam eder.

```
396      while (gameCanContinue(board, boardX) && choice != 9) {
397          clear(gamemode);
398          if (choice == 2 && step == 2) {
399              drawBoard(board, boardX, boardY,  highlight: 1,  highlightX: chx1,  highlightY: chy1);
400          }
401          else {
402              drawBoard(board, boardX, boardY,  highlight: 0,  highlightX: 0,  highlightY: 0);
403          }
404          printf( format: "\n");
405          drawStats( swp: swapCount,  exp: explosionCount);
406          printf( format: "\n");
407
408          if (showInfo) {
409              if (showInt) printf( format: "%s%d %s%s\n\n", BLUE, messageInt, message, RESET);
410              else printf( format: "%s%s%s\n\n", BLUE, message, RESET);
411          }
412          else if (showError) {
413              printf( format: "%s%s%s\n\n", RED, message, RESET);
414          }
415          else {
416              printf( format: "\n\n");
417          }
```

Döngünün her turunda önce ekran oyun moduna göre temizlenir (kontrol modunda gamemode = 0 olacağından “clear” fonksiyonuna “fullClear” parametresi 0 olarak gönderilir ve ekran temizlenmez).
choice = 2 ve step = 2 kontrolü ile yer değiştirme işleminde seçilen ilk elementin oyun tahtası yazdırılırken vurgulanması sağlanır, choice = 2 yer değiştirmesi yapıldığını ve step = 2 ikinci elementin seçim aşamasında bulunulduğunu belirtir. Diğer durumlarda oyun alanı herhangi bir vurgu olmadan yazdırılır.
“drawStats” fonksiyonu ile oyuncu istatistikleri oyun tahtasının altına yazdırılır.
Herhangi bir bilgilendirme veya hata mesajı varsa rengi ile beraber (bilgilendirme için mavi, hata mesajları için kırmızı) ekrana yazdırılır. Eğer mesaj yazdırılmayacaksa mesaj yazdırılan bölge boş bırakılır, bu sayede arayüz kayması önlenir.

```
419             if (step == 0) {
420                 printf( format: "Islem secin:\n");
421                 printf( format: "1) Patlatma\n");
422                 printf( format: "2) Yer Degistirme\n");
423                 printf( format: "9) Oyunu Bitir\n");
424                 printf( format: ">  ");
425                 scanf( format: "%d", &choice);
426                 printf( format: "\n");
427 
428                 step = 1;
429                 showInfo = 0;
430                 showError = 0;
431                 showInt = 0;
432             }
```

“step” değişkeni oyun başında 0’a eşittir. Bu durumda kullanıcıdan bir işlem seçmesi istenir. Seçilen değer “choice” değişkenine atanır. “Oyunu Bitir” seçeneği seçildiği zaman “choice” değişkeni 9’a eşitlenir, ve daha öncesinde bahsedildiği gibi oyunun bitmesini sağlar. Girilen girdinin validasyonu burada yapılmaz. Girdi alındıktan sonra “step” değişkeni 1’e eşitlenir ve herhangi bir mesaj gösterilmeyeceği için ilgili değişkenler 0’a eşitlenir.

```
433         else if (choice == 1) {
434             printf( format: "Patlatilacak elementin koordinatini giriniz (Format: y,x Ornek: 6,7): \n");
435             printf( format: "> ");
436             scanres = scanf( format: "%d,%d", &chy1, &chx1);
437 
438             // koordinatlar 1den basliyor, arrayler 0dan basliyor
439             chx1--;
440             chy1--;
```

“step” değişkeni 0’a eşit değil ise bir işlem seçilmiştir. Patlatma işlemi seçildiği durumlarda kullanıcıdan patlatılacak elementin koordinatı istenir. Arayüzdeki koordinatlar 1’den başladığı için daha sonra bu değerler 1 eksiltilir, bu sayede doğrudan dizi indisi olarak kullanılabilirler.

```
442             if (chx1 < 0 || chx1 ⩾ boardX || chy1 < 0 || chy1 ⩾ boardY || scanres ≠ 2) {
443                 message = "Hatali giris yaptiniz.";
444                 showInfo = 0;
445                 showInt = 0;
446                 showError = 1;
447             }
448             else if (board[chy1][chx1] == 0) {
449                 message = "Verilen koordinatta element yok. Patlatma yapilmadi.";
450                 showInfo = 1;
451                 showInt = 0;
452                 showError = 0;
453                 step = 0;
454             }
```

Girilen koordinatların geçerli olup olmadığı kontrol edilir (girilen koordinat oyun alanında olmalı, ve “scanf” fonksiyonunun döndürdüğü değer 2 değer alındığı için 2 olmalı). Geçersiz koordinat girilirse “message” değişkenine hata mesajı girilir ve “showError” değişkeni aracılığıyla gösterilir. “step” değişkeni değiştirilmez, bu sayede kullanıcıdan tekrardan koordinat girdisi istenir. Eğer girilen koordinatta element yoksa “message” değişkenine bilgilendirme mesajı girilir ve “showInfo” değişkeni aracılığıyla gösterilir. “step” değişkeni sıfırlanır ve kullanıcıya yeni işlem seçtirilir.

```
else {
    int result = explode(board, boardX, boardY,  x: chx1,  y: chy1);
    explosionCount += result;

    if (result) {
        messageInt = result;
        message = "element patlatildi.";

        showInfo = 1;
        showInt = 1;
        showError = 0;

        step = 0;
    }
    else {
        message = "Patlatma icin yeterli element yok.";
        showInfo = 0;
        showInt = 0;
        showError = 1;

        step = 0;
    }
}
```

Eğer koordinatların uygun olduğu görülürse “explode” fonksiyonu çağırılır. Fonksiyon toplam patlatılan element sayısı döndürür ve bu kullanıcının toplam patlattığı element sayısına eklenir. Turda patlatılan element sayısı 0 ise kullanıcıya hata mesajı gösterilir, değilse kullanıcıya kaç element patlattığına dair bilgilendirme mesajı gösterilir. Daha sonra kullanıcının yeni bir işlem seçebilmesi için “step” değeri sıfırlanır.

```
479         else if (choice == 2 && step == 1) {
480             printf( format: "Yer degistirme icin ilk elementin koordinatini giriniz (Format: y,x Ornek: 6,7): \n");
481             printf( format: ">  ");
482             scanres = scanf( format: "%d,%d", &chy1, &chx1);
483             chx1--;
484             chy1--;
485 
486             if (chx1 < 0 || chx1 ≥ boardX || chy1 < 0 || chy1 ≥ boardY || scanres ≠ 2) {
487                 message = "Hatali giris yaptiniz.";
488                 showInfo = 0;
489                 showInt = 0;
490                 showError = 1;
491             }
492             else {
493                 showInfo = 0;
494                 showInt = 0;
495                 showError = 0;
496                 step = 2;
497             }
498         }
```

Kullanıcı yer değiştirme işlemi seçmişse kullanıcıdan ilk elementin koordinatı istenir. Alınan koordinat değerleri dizi indisinde kullanılabilmesi için 1 azaltılır. Koordinatın uygunluğu kontrol edilir, uygun değilse hata mesajı gösterilir. “step” değişkeni tekrar koordinat girdisi alma amaçlı değiştirilmez. Koordinat uygunsa mesaj gösterilmez, “step” değişkeni 2’ye eşitlenir.

```
499         else if (choice == 2 && step == 2) {
500             printf( format: "Yer degistirme icin ikinci elementin koordinatini giriniz (Format: y,x Ornek: 6,7): \n");
501             printf( format: ">  %d,", chy1 + 1);
502             scanres = scanf( format: "%d", &chx2);
503             chx2--;
```

Yer değiştirme işlemi seçiliyken “step” değişkeni 2’ye eşitlendiği için bir sonraki turda ikinci koordinat istenir. Sadece yan yana olan elementler yer değiştirebileceği için kullanıcının Y ekseni koordinatı girmesine izin verilmez, sadece X ekseni koordinatı alınır. Alınan değer dizi indisinde kullanılabilmesi için 1 azaltılır.

```
505                 if (chx2 < 0 || chx2 ≥ boardX || scanres ≠ 1) {
506                     message = "Hatali giris yaptiniz.";
507                     showInfo = 0;
508                     showInt = 0;
509                     showError = 1;
510                 }
```

Girilen koordinatın geçerli olup olmadığı kontrol edilir, geçerli değilse hata mesajı gösterilir. “step” değişkeni değiştirilmeyerek tekrar koordinat istenir.

```
511             else if (!board[chy1][chx1] || !board[chy1][chx2]) {
512                 message = "Sadece farkli renkte iki element yer degistirebilir. Element bos alana gecirilemez.";
513                 showInfo = 0;
514                 showInt = 0;
515                 showError = 1;
516                 step = 0;
517             }
518             else if (board[chy1][chx1] == board[chy1][chx2]) {
519                 message = "Ayni renkte iki element yer degistiremez.";
520                 showInfo = 0;
521                 showInt = 0;
522                 showError = 1;
523                 step = 0;
524             }
525             else if (chx1 + 1 ≠ chx2 && chx1 - 1 ≠ chx2) {
526                 message = "Sadece yan yana olan elementler yer degistirebilir.";
527                 showInfo = 0;
528                 showInt = 0;
529                 showError = 1;
530                 step = 0;
531             }
```

Girilen koordinatların oyun kurallarına uygunluğu kontrol edilir, eğer oyun kurallarına uygun bir yer değiştirme işlemi girilmemişse hata mesajı gösterilir. “step” değişkeni sıfırlanarak kullanıcının tekrar işlem seçmesi istenir.

```
532             else {
533                 int temp = board[chy1][chx1];
534                 board[chy1][chx1] = board[chy1][chx2];
535                 board[chy1][chx2] = temp;
536 
537                 swapCount++;
538 
539                 message = "Yer degisimi yapildi.";
540                 showInfo = 1;
541                 showInt = 0;
542                 showError = 0;
543                 step = 3;
544             }
```

Geçerli koordinatlar girilmişse ve kurallara uygun bir yer değiştirme işlemi yapılıyorsa geçici bir değişken aracılığıyla verilen koordinatlardaki elemanların yeri değiştirilir. Kullanıcıya bilgilendirme mesajı gösterilir. “step” değeri 3’e eşitlenir.

```
546         else if (choice == 2 && step == 3) {
547             wait( seconds: 2);
548             addRandomLine(board, boardX, boardY, chars);
549             step = 0;
550         }
```

“step” değişkeni 3’e eşit iken kullanıcı 2 saniye bekletilir, oyun alanına yeni bir satır eklenir, daha sonra “step” değişkeni sıfırlanarak kullanıcının yeni bir işlem yapmasına izin verilir (yeni eklenen satır oyunu bitirirse yeni bir işlem yapılamaz). Kullanılan 2 fonksiyon sonraki sayfada anlatılmıştır.

```
112  void wait(int seconds) {
113      long end = time(NULL) + seconds;
114      while (time(NULL) < end) {
115      }
116  }
```

“wait” fonksiyonu kaç saniye bekleneceğini parametre olarak alır. Fonksiyon ilk çağırıldığı andaki saniye cinsinden tarih-saate bu parametreyi ekler ve “end” değişkeninde saklar. Anlık saniye cinsinden tarih-saat, “end” değişkenini geçene kadar bekler.

```
351  void addRandomLine(int board[20][20], int boardX, int boardY, int chars[5]) {
352      int i, j;
353
354      for (i = 0; i < boardX; i++) {
355          for (j = 1; j < boardY; j++) {
356              board[j - 1][i] = board[j][i];
357          }
358          board[boardY - 1][i] = chars[rand() % 5];
359      }
360  }
```

“addRandomLine” fonksiyonu parametre olarak oyun alanını, alanın boyutlarını ve oyun elementlerini alır. Oyun alanındaki tüm elementleri bir satır yukarı kaydırır ve en alt satırdaki tüm kareleri rastgele elementler ile doldurur.

“while” döngüsündeki koşullardan birinin sağlanmaması ile oyun biter.

```
559      }
560
561      clear(gamemode);
562      drawBoard(board, boardX, boardY,  highlight: 0,  highlightX: 0,  highlightY: 0);
563      printf( format: "\n");
564      drawStats( swp: swapCount,  exp: explosionCount);
565      printf( format: "\n\n");
566      printf( format: "%sOyun bitti.%s", GREEN, RESET);
567      printf( format: "\n\n");
568
569      return 0;
570  }
```

Oyun bitiminde oyun moduna göre ekran son bir kez temizlenir, oyun alanı ve istatistikler ekrana yazdırılır, kullanıcıya “Oyun bitti.” mesajı gösterilir ve oyun kapanır.

Video Linki: https://www.youtube.com/watch?v=9wf1LoPit_4