PROJE#2

Bu proje için, mekaniği diğer ekte verilmiş olan RPG'nin simülasyonu, C veya C++ programlama dillerinden birisi kullanılarak gerçeklenecektir. Simülasyonun alacağı parametreler ve yapacağı işlemler aşağıdaki sıralama ile verilebilir.

1. Simülasyon, ismi ve sınıfı verilen karakteri, oyun kurallarına (bkz. diğer ek!) bağlı kalarak oluşturur.
2. Simülasyon, boyutu verilen labirenti oluşturur, oyun kurallarına bağlı kalarak zorunlu nesneleri labirente yerleştirir.
3. Simülasyon, derste verilen yön bulma algoritmasını (veya bir miktar modifiye edilmiş halini) kullanarak karaktere oyunu oynatır.
4. Oyunun adımlardan oluştuğu (turn based) göz önüne alınırsa, simülasyon her bir adımda gerçekleşen aksiyonları ekrana basar.

Proje teknik açıdan ele alınırsa, teslimlerden beklenen temel içerik şu şekilde verilebilir.

- Yığın (yol bulmada ve labirent oluşturmada kullanılır)
- Kuyruk (nerede kullanılacağı ileride belirtilmiştir)
- Rastgele sayı üretimi (sürekli zar atıyoruz)

1. Karakter Oluşturma

RPG oyun sistemlerinde karakter oluşturmak için farklı yöntemler kullanılabilir. Karakter sınıflarına sabit özellikler atanması, oyuncunun elindeki puanları özelliklere istediği gibi dağıtması veya özelliklerin zar atılarak belirlenmesi bu yöntemlere örnek olarak verilebilir. Ekte verilen basit oyun sistemi sadece 12 ve 20 yüzlü zarlar kullanmakta ve bu zarları sabit bir taban sayısının üstüne eklemektedir. Örnek olarak X karakter sınıfının özellikleri şu şekilde belirlensin;

```
Strength: 5 + 2d12
Intelligence: 7 + 1d20
Dexterity: Avg(Str+Int) + 1d12
```

Verilen değerlere göre, karakterin "strength" özelliği 12 yüzlü zarın iki kere atılmasıyla (2d12) elde edilen sayının 5 ile toplanmasıyla elde edilir. Aynı şekilde "intelligence" 20 yüzlü zarın bir kere atılmasının ardından elde edilen sayının 7 ile toplanmasıyla bulunur. "dexterity" özelliği ise elde edilen "strength" ve "intelligence" değerlerinin aritmetik ortalamasına 12 yüzlü zarın bir kere atılmasıyla elde edilen sayının eklenmesiyle bulunur (örnekler sadece örnektir).

Karaktere has özelliklerin tutulması için, bu özelliklere uygun veri tipleri ile tanımlanmış değişkenlerin bir struct içerisinde tutulması düşünülebilir (çünkü bunlar aynı varlığa ilişkin özellikler). Örnek;

```
struct adventurer
{
      char *name;
      int strength;
      int intelligence;
      …
};
```

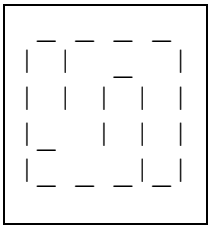
2. Labirent Oluşturma

Labirent oluşturmak için şu çözümler kullanılabilir,

- Depth First Search mantığına dayalı yaklaşım
- Kruskal Algoritmasına dayalı yaklaşım
- Prim Algoritmasına dayalı yaklaşım

Bu labirent oluşturma algoritmaları genel hatlarıyla birbirine benzemektedir ve herhangi birini gerçeklemek kesinlikle zor değildir. http://en.wikipedia.org/wiki/Maze_generation_algorithm adresinden pseudocode'lar ve Web'deki mevcut implementasyonlar bulunabilir.

Labirent oluşturma işleminden sonra aşağıdakine benzer formatta bir harita bulunacaktır.



Yukarıdaki harita 4x4 boyutunda olup, görüldüğü gibi bütün hücrelerin birbiri ile bağlantısı vardır. Bu durumda haritaya konması gerekli olan şeyler (karakter, portal, elmaslar) rastgele yerleştirilebilir. Yaratıklar haritaya yerleştirilmeyecek, karakter bir kareye gittiğinde zarlar atılarak karşısına yaratık çıkıp çıkmayacağı belirlenecektir. Görüldüğü gibi harita 4x4'lük dizide tutulabilir. Dizinin elemanları şu struct ile verilebilir (işinize geldiği gibi struct yapıları kurabilirsiniz).

```
struct hucre
{
	int sag; // sag=0 sagda duvar yok; sag=1 sagda duvar var…
	int sol;
	int asagi;
	int yukari;
	…
};

struct hucre lab[4][4]; // iste labirent!
```

3. Simülasyonun Oyunu Oynatması

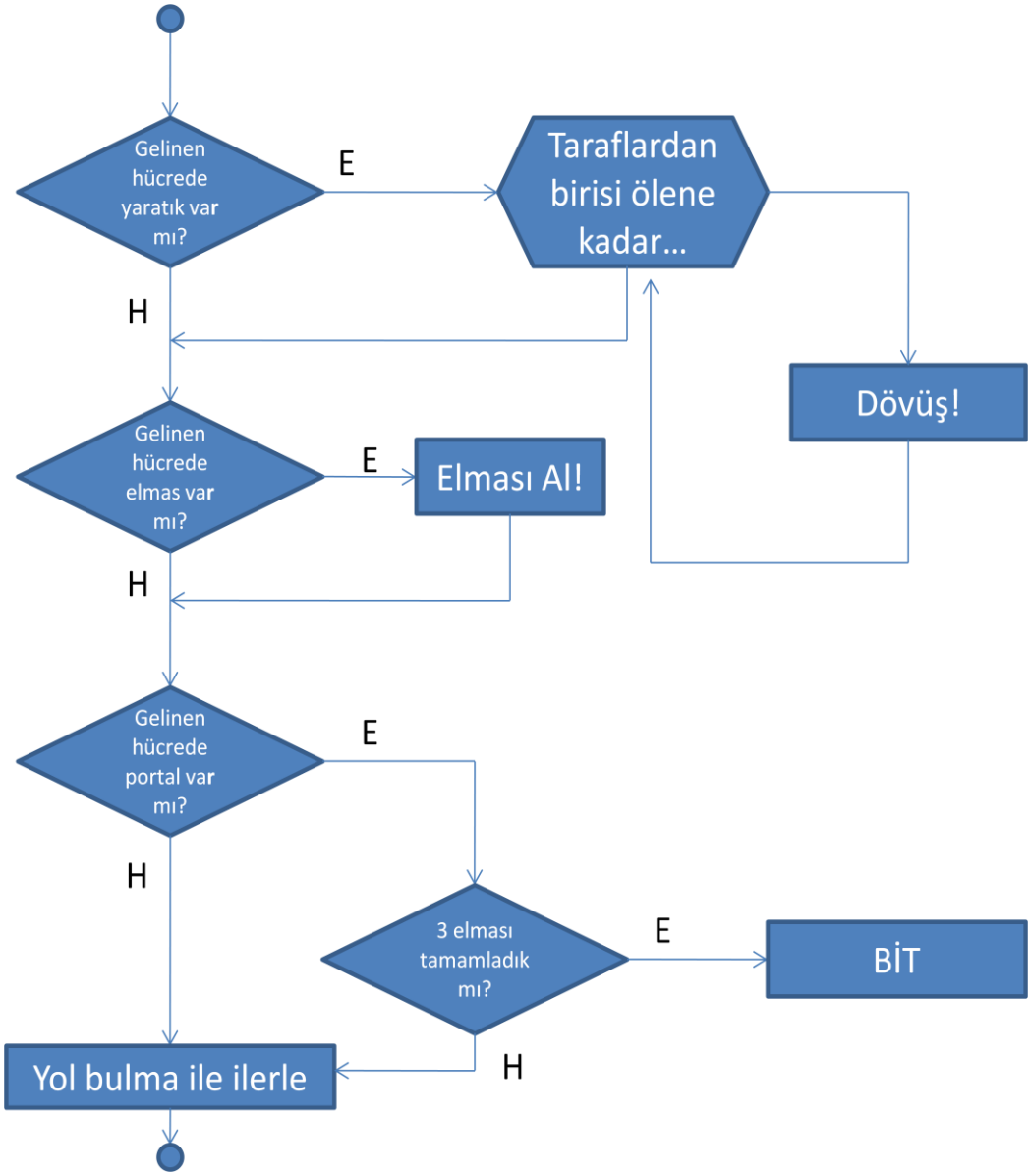
Bu kısım çoğunlukla derste verilen yol bulma algoritması kullanılarak gerçeklenecektir. Oyun senaryosu ve kuralları diğer ekte verilmiştir. Derste verilen yol bulma algoritması, senaryonun gerçeklenmesi için bir miktar değiştirilmelidir.

Derste verilen yol bulma algoritması bir başlangıç noktasından bitiş noktasına gitmek için kullanılmaktadır. Dolayısıyla 3 elmasın aranması, 3 elmasın bulunduktan sonra portala gidilmesi için koda biraz ekleme yapılmalıdır.

Oyunun sonunda, öldürülen yaratıklar kullanıcıya hangi tur öldürüldükleri bilgisi ile ekrana basılacaktır. **Bu işlem için kuyruk yapısı kullanılabilir (öldürülen yaratık mezarlık kuyruğuna atılıyor…).** Ölen yaratıkların bazı özellikleri de zar atılarak belirlendiğinden bu özellikler de düğümlerde tanımlanmış değişkenler üzerinde tutulabilir (yaratığın hp'si, strength'i vs…).

4. Simülasyon Adımlarının Ekrana Basılması

Karakter oyunda her tur 4 farklı hamleden sadece birini yapabilir: (1) labirentte ilerler, (2) bir yaratıkla dövüşür, (3) elması alır, (4) elmasları portala yerleştirir ve labirentten çıkar. Simülasyon, karakter için her tur şu akış şemasını izleyen işlemleri gerçekleştirir (dikdörtgenlere rastladığımızda 1 tur geçmiş oluyor).



Elmaslar ve portal labirente oyunun başında sabit olarak yerleştirilir. Gelinen hücrede yaratık olup olmadığı, eğer varsa nasıl bir şey olduğu, bir hücreye gelindiğinde zar atılarak belirlenir.

5. Örnek Simülasyon Çıktısı

4x4'lük bir harita için, örnek oyun çıktısı aşağıdaki gibi verilebilir (tur tur ekrana ne olup bittiği basılıyor). Farklı çıktılar, ekstra özellikler ve karakterin sahip olacağı daha gelişmiş yapay zeka implementasyonları ekstra notla ödüllendirilecektir (bu projeden alabileceğiniz maksimum not 200'dür.).

```
☺: Karakter (aycan_the_slayer)
R: Kırmızı elmas
G: Yeşil elmas
B: Mavi elmas
P: Portal
```

Adım 1:

```
 _ _ _ _
|☺|  P  |
| | | |R|
|_  |G| |
|B _ _|_|

Oyun başlıyor!
```

Adım 2:

```
 _ _ _ _
| |  P  |
|☺| | |R|
|_  |G| |
|B _ _|_|

Gidilen yön: AŞAĞI
Hiçbir şey olmadı…
```

Adım 3:

```
 _ _ _ _
| |  P  |
| | | |R|
|☺  |G| |
|B _ _|_|

Gidilen yön: AŞAĞI
Yaratık Çıktı: Malkavian
İlk Atak: aycan_the_slayer
```

Adım 4:

```
 _ _ _ _
| |  P  |
| | | |R|
|☺  |G| |
|B _ _|_|

aycan_the_slayer Saldırdı
-15 hp
```

Adım 5:

```
 _ _ _ _
| |  P  |
| | | |R|
|☺  |G| |
|B _ _|_|

Malkavian Saldırdı
-7 hp
```

Adım 6:

```
 _ _ _ _
| |  P  |
| | | |R|
|☺  |G| |
|B _ _|_|

aycan_the_slayer Saldırdı
-22 hp
Malkavian ex oldu.
```

6. Notlar

- Çıktılara ve verilen akışa uymak zorunlu değildir. Olabildiğince oyun mekaniğine uymaya çalışın. Oyun mekaniği, projenin kolay olması için çok çok ve çok cılız tutulmuştur.
- Ekstra efor sarf edilmiş ödevlerin notu 100'ü geçebilir.
- Kopya, yüksek öğrenimde karşılaşılabilecek en büyük suçlardan biridir. Hazırladığınız ödev ve projeleri güvenmediğiniz kişilerle paylaşmayın.
- Proje ile ilgili sorularınızı mail yoluyla ya da yüz yüze sormaktan çekinmeyin.

Kolay Gelsin.