Bu kısa raporda oyunun algoritması yerine algoritmanın gömülü sistem yardımıyla gerçekleştirilme aşamasını anlatacağım.

Oyun için 2 boyutlu matrisi harita olarak baz almayı düşündüm.Bu haritanın büyüklüğü oranında LED'i kontrol etmem gerekti.Her bir LED için bir adet output olarak set edilmiş pini ayarlamam gerekti.En başta 6x6 lık bir harita yapmayı tasarlamıştım.Fakat bazı pinler debugger,bazıları da USB bağlantısına atandığı,bazı pinler ise direkt olarak birbirine bağlı olduğu ve onları ayırmak için gereken komutlar olayı karmaşıklaştıracağı 4x4 alana kadar küçülttüm.Bu pinlerle ilgili bilgiler `http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C6_MicrocontrollerPorts.htm` adresinde mevcuttur.

16 adet pini A portundan 6, B portundan 8 ve F portundan 2 olmak üzere seçtim.Oyun haritasındaki pin dağılımı aşağıdaki gibidir.

	0	0	0	0
0	PB5	PF1	PF2	PB2
0	PB0	PB1	PB6	PB7
0	PB4	PA5	PB3	PA4
0	PA6	PA7	PA2	PA3

Pinlerin ucu ile LED'ler arasına 470 ohmluk dirençler koyarak akımı küçültmek gerekiyor.

A ve B portunun pinlerini aktive etmek için bu portların clocklarını açmak ve çeşitli registerlerini tüm pinler output olacak şekilde ayarlamak gerekti.Aşağıda A portunun register adresleri ve initialize fonksiyonunu örnek olarak koydum.

Listing 1: A Portu Registerlerinin Memory Adresleri

```
#define PORTA_DATA  (*((volatile unsigned long *)0x400043FC))
#define PORTA_DIR   (*((volatile unsigned long *)0x40004400))
#define PORTA_DEN   (*((volatile unsigned long *)0x4000451C))
#define PORTA_AFSEL (*((volatile unsigned long *)0x40004420))
#define PORTA_AMSEL (*((volatile unsigned long *)0x40004528))
#define PORTA_PCTL  (*((volatile unsigned long *)0x4000452C))
```

Listing 2: A Portu Pinlerinin Output Ayarlanması

```
void PortA_Init(void){

volatile unsigned long delay;
SYSCTL_RCGC2_R |= 0x00000001;
delay = SYSCTL_RCGC2_R;

PORTA_DIR = 0xFF;
PORTA_AMSEL = 0x00;
PORTA_AFSEL = 0x00;
PORTA_PCTL = 0x00000000;

PORTA_DEN = 0xFF;
}
```

Port pinlerini set etmek için şimdiye kadar olan lablarda pinin registerdaki sırasına göre değişik sayılarla or'luyoruk.Örneğin launchpadin üzerindeki mavi ledi yakmak için DATA registerinin içeriğini 0x04 sayısıyla OR'larız.Fakat her bir pin için özel bir sayı yeri bulmak yerine 0xFF sayısını kullanabiliriz.Bunun için de bit-specific-adressing metodunu kullandım.

Bu modla beraber , her pine atanan özel adresin içeriğini 0xFF ile orlayarak ya da 0x00 ile and'leyerek sadece o pinin LED'i yakıp söndürmesini sağlayabiliriz.Bu adresleme moduyla ilgili bilgi yukarda verdiğim linkte var.Aşağıda A portunun herbir pininin özel adresini dizide tutulmasını sağlayan kod parçasını veriyorum.

```
PORTA_ARRAY[0] = ((volatile unsigned long *)0x40004004); //PA0
PORTA_ARRAY[1] = ((volatile unsigned long *)0x40004008); //PA1
PORTA_ARRAY[2] = ((volatile unsigned long *)0x40004010); //PA2
PORTA_ARRAY[3] = ((volatile unsigned long *)0x40004020); //PA3
PORTA_ARRAY[4] = ((volatile unsigned long *)0x40004040); //PA4
PORTA_ARRAY[5] = ((volatile unsigned long *)0x40004080); //PA5
PORTA_ARRAY[6] = ((volatile unsigned long *)0x40004100); //PA6
PORTA_ARRAY[7] = ((volatile unsigned long *)0x40004200); //PA7
```

Bu noktadan sonra iki boyutlu bir matris yaratıp pinleri doğru yerlere yerleştirdikten sonra oyun algoritması çalışıyor.