SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

ADI: **Emirhan**
SOYADI: **Güngör**
NUMARA:2301090309

ADI: **Mehmet Emin**
SOYADI: **Bozdoğan**
NUMARA:23010903051

ÖDEV: Bir Oyun Tasarımı ve Geliştirilmesi

AKADEMİSYEN: **Dr. Öğr. Üyesi Selman HIZAL**
Arş. Gör. Furkan Atban

Ödev Tanıtımı

1. Başlangıç Ayarları

- SFML kütüphanesiyle bir pencere oluşturulur.
- Oyun için gerekli olan sınıflar ve objeler tanımlanır:
    - **Top**: Oyuncunun kontrol ettiği oyun objesi.
    - **Engel**: Oyuncunun çarpmaktan kaçınması gereken nesne.
    - **Yıldız**: Oyuncunun toplayarak puan kazandığı nesne.
- Rastgelelik için rand() fonksiyonu ayarlanır.
- Saatler ve oyun süreleri tanımlanır.

2. Ana Döngü

Oyun çalışırken aşağıdaki işlemler sürekli olarak gerçekleştirilir:

a. Kullanıcı Girdileri

- Oyuncunun yön tuşlarıyla topu hareket ettirme komutları alınır ve yön bilgisi (**Yon**) belirlenir.

b. Topun Hareketi

- Top, belirlenen yöne doğru hareket ettirilir.
- Topun oyun alanı sınırları dışına çıkıp çıkmadığı kontrol edilir. Çıkarsa oyun sonlanır.

c. Puan Güncellemesi

- Oyunun başlangıcından itibaren geçen süre, oyuncunun puanı olarak atanır.

d. Engellerin Oluşturulması

- Zaman aralıklarına göre engeller oluşturulur:
    - Oyun başında tekli engeller.
    - Oyun ilerledikçe çiftli ve üçlü engeller.
- Engeller sürekli aşağı doğru hareket eder.

- Ekran dışına çıkan engeller listeden silinir.

e. Çarpışma Kontrolü (Engellerle)

- Top ile herhangi bir engel çarpışırsa oyun sonlanır.

f. Yıldızların Oluşturulması

- Zaman aralıklarına göre yıldızlar oluşturulur.
- Yıldızlar sürekli aşağı doğru hareket eder.
- Ekran dışına çıkan yıldızlar listeden silinir.

g. Çarpışma Kontrolü (Yıldızlarla)

- Top bir yıldızla çarpışırsa yıldız puanı 30 artırılır.
- Çarpışan yıldız, listeden silinir.

h. Renk Değişimi

- Oyunun arka plan rengi, belirli bir süre sonunda beyaz ve siyah arasında değişir.

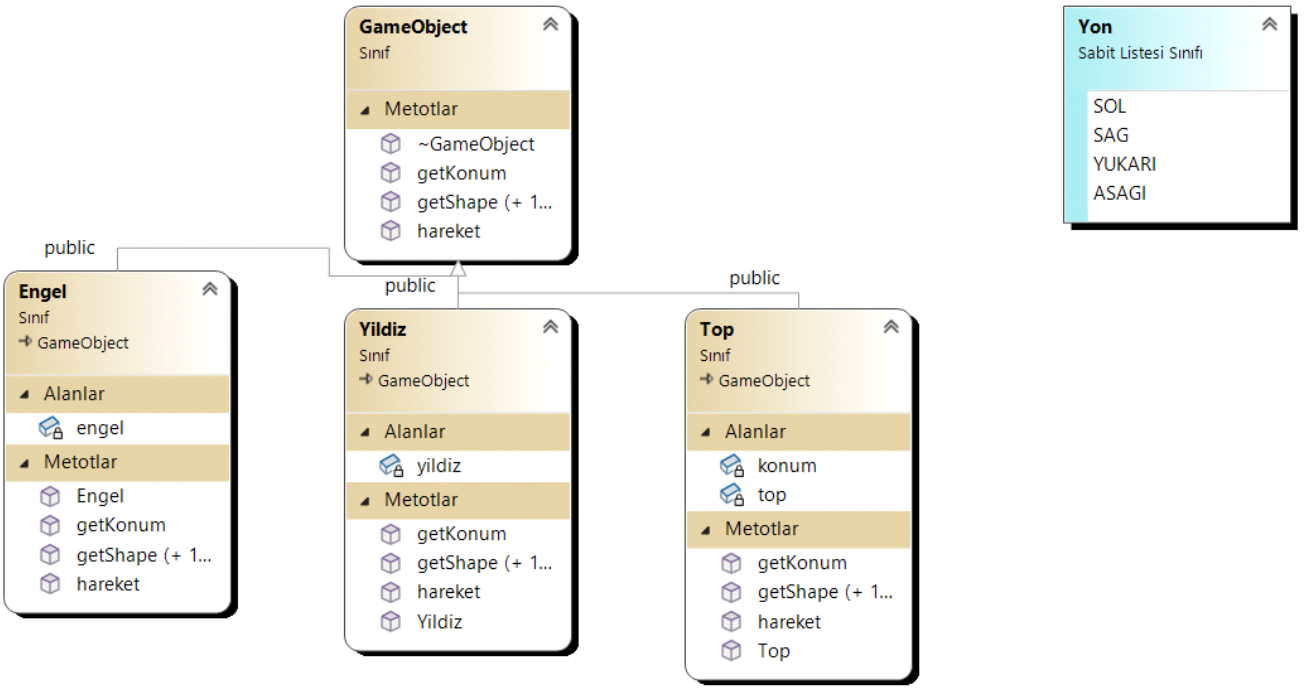
i. Görsel Güncelleme

- Pencere temizlenir ve oyun objeleri güncel konumlarıyla çizilir.

3. Oyun Sonu

- Eğer bir hata meydana gelirse (örneğin topun engellere çarpması veya alan dışına çıkması):
    - Oyun biter.
    - Oyuncunun toplam puanı ve yıldız puanı ekrana yazdırılır.

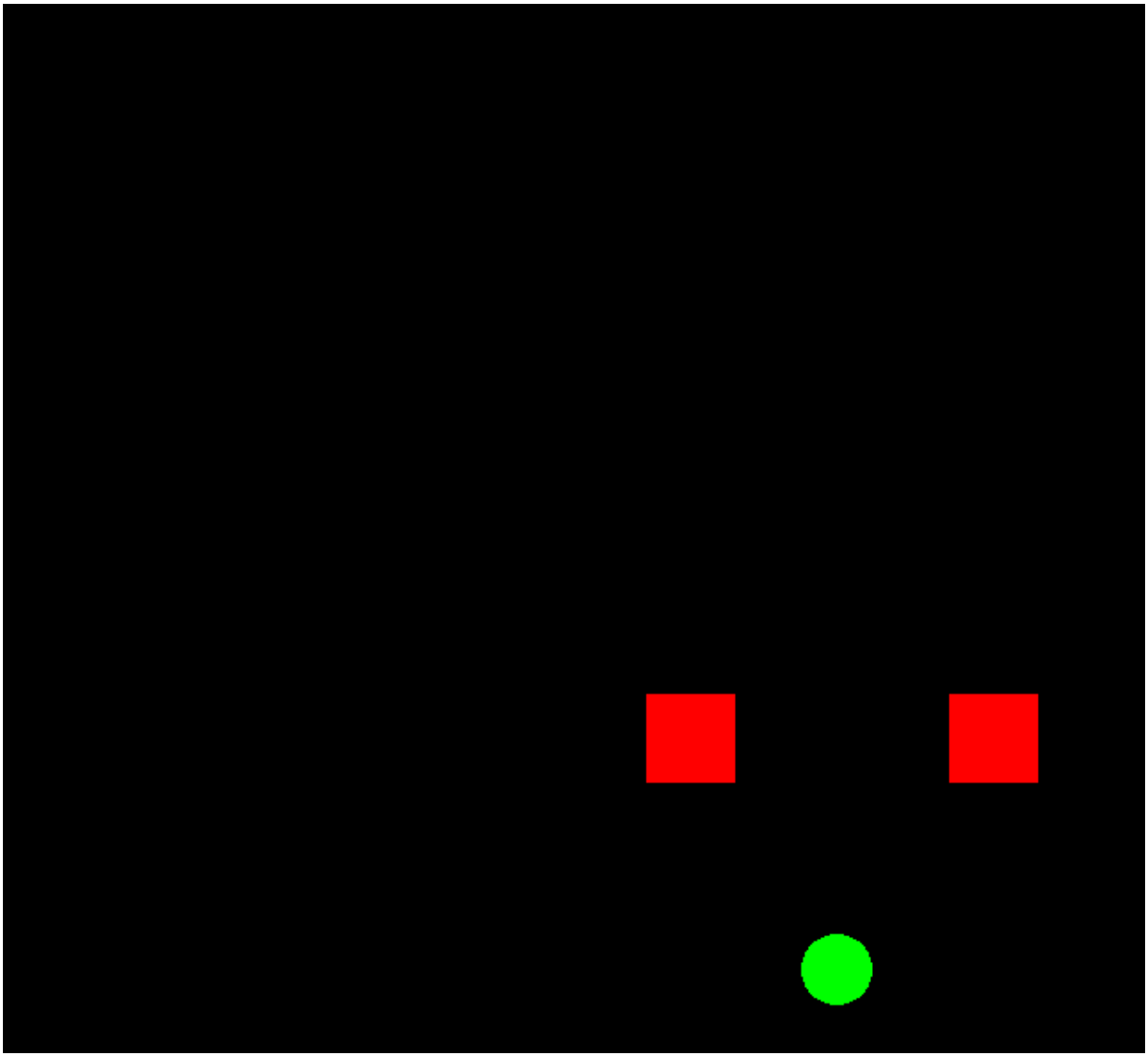
UML Diyagramı



Oyun İçi Görüntüler

Microsoft Visual Studio Hata /

```
Oyun Bitti! Top bir engele carpti!
Puaniniz: 50
Yildiz Puaniniz: 30
```



Inheritance İlişkisi(Kalıtım İlişkisi)

GameObject (Temel Sınıf)

- **GameObject**, diğer tüm oyun nesneleri için bir temel (abstract) sınıf olarak tanımlanmış.
- Bu sınıf, tüm alt sınıfların sahip olması gereken ortak özellikler ve davranışları (metotları) tanımlar:
    - **Metotlar**:
        - hareket(Yon yon) (sanal/metot): Alt sınıflar kendi spesifik hareket mantıklarını sağlayacak.
        - getShape() (sanal/metot): Her alt sınıf bir şekil döndürmek zorunda.
        - getKonum() (sanal/metot): Konumu almak için gerekli.

- **Destructor**:
    - ~GameObject() sanal destructor, alt sınıfların uygun şekilde yok edilmesini sağlar.

Bu sınıf **abstract** bir sınıf olarak tasarlandığı için doğrudan bir nesne oluşturulamaz. Sadece diğer sınıflar bu sınıfı **inherit (miras)** ederek kullanabilir.

Top

- **Kalıtım ilişkisi**:

cpp

Kodu kopyala

class Top : public GameObject

- **Özellikler**:
    - sf::CircleShape top: Bu, topun görsel temsilini tutar.
    - sf::Vector2f konum: Topun pozisyonunu saklar.
- **Metotlar**:
    - hareket(Yon yon) metodu: Topun oyun alanında nasıl hareket edeceğini belirler (sağa, sola, yukarı, aşağı).
    - getShape(): Topun grafiği olan CircleShape nesnesini döndürür.
    - getKonum(): Topun mevcut pozisyonunu döndürür.

Engel

- **Kalıtım ilişkisi**:

cpp

Kodu kopyala

class Engel : public GameObject

- **Özellikler**:
    - sf::RectangleShape engel: Engel nesnesinin görsel temsilini tutar.
- **Metotlar**:
    - hareket(Yon yon) metodu: Engel nesnesinin hareket mantığını sağlar (aşağı doğru hareket eder).
    - getShape(): Engel nesnesinin RectangleShape nesnesini döndürür.
    - getKonum(): Engel nesnesinin pozisyonunu döndürür.

Yildiz

- **Kalıtım ilişkisi**:

cpp

Kodu kopyala

class Yildiz : public GameObject

- **Özellikler**:
    - sf::CircleShape yildiz: Yıldızın görsel temsilini tutar.
- **Metotlar**:
    - hareket(Yon yon) metodu: Yıldız nesnesinin hareket mantığını sağlar (aşağı doğru hareket eder).
    - getShape(): Yıldız nesnesinin CircleShape nesnesini döndürür.
    - getKonum(): Yıldız nesnesinin pozisyonunu döndürür.

Enkapsülasyon

top ve konum üyeleri private olarak tanımlanmıştır, bu da doğrudan erişimi engeller.

Bu verilere erişim ve değişiklikler, sadece public yöntemler (hareket, getShape, getKonum) aracılığıyla yapılabilir.

engel üyesi private olarak tanımlanmıştır ve sadece public yöntemler aracılığıyla erişilebilir.

yildiz üyesi private olarak tanımlanmıştır ve sadece public yöntemler aracılığıyla erişilebilir.

Soyutlama

soyutlama, oyun nesnelerinin ortak özelliklerinin ve davranışlarının bir üst sınıf olan GameObject sınıfında toplanmasıyla yapılır. Top, Engel ve Yildiz gibi sınıflar, GameObject sınıfını miras alarak kendi özel özelliklerini tanımlar.

Soyutlama avantajları:

- Tekrar Kullanılabilirlik
- Esneklik

Polimorfizm

hareket(Yon yon): GameObject sınıfı soyut bir hareket metodu tanımlar, ancak her alt sınıf bu metodu kendine özgü şekilde implement eder. Örneğin, Top sınıfı, topu hareket ettirirken, Engel ve Yildiz sınıfları kendi nesnelerini hareket ettirir. Bu metotların hepsi aynı isme sahip olmasına rağmen farklı davranışlar gösterir.

getShape(): Her alt sınıf, GameObject sınıfındaki getShape() metodunu kendi özel türü olan sf::CircleShape veya sf::RectangleShape ile implement eder. Bu sayede, tüm nesneler aynı arayüzü (interface) kullanır, ancak farklı türde şekiller döndürür.

Kodlar

Oyun.hpp

```
#ifndef OYUN_HPP
#define OYUN_HPP

#include <SFML/Graphics.hpp>
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <memory>

// Hareket yönü enum sınıfı
enum class Yon {
    SOL,    // Sol yön
    SAG,    // Sağ yön
    YUKARI, // Yukarı yön
    ASAGI   // Aşağı yön
};
```

```
// Temel oyun objesi sınıfı
class GameObject {
public:
    // Saf sanal fonksiyonlar (abstract methods)
    virtual void hareket(Yon yon) = 0; // Hareket fonksiyonu
    virtual sf::Shape& getShape() = 0; // Şekil döndüren fonksiyon
    virtual const sf::Shape& getShape() const = 0; // Şekil döndüren fonksiyon (const versiyon)
    virtual sf::Vector2f getKonum() const = 0; // Konum döndüren fonksiyon
    virtual ~GameObject() = default; // Sanal yıkıcı
};

// Top sınıfı
class Top : public GameObject {
public:
    // Yapıcı fonksiyon
    Top(float yaricap, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
    sf::CircleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::CircleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;

private:
    sf::CircleShape top; // Topun şekli
    sf::Vector2f konum;  // Topun konumu
```

```
};

// Engel sınıfı
class Engel : public GameObject {
public:
    // Yapıcı fonksiyon
    Engel(float genislik, float yukseklik, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
    sf::RectangleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::RectangleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;

private:
    sf::RectangleShape engel; // Engel şekli
};

// Yıldız sınıfı
class Yildiz : public GameObject {
public:
    // Yapıcı fonksiyon
    Yildiz(float yaricap, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
```

```
    sf::CircleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::CircleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;

private:
    sf::CircleShape yildiz; // Yıldız şekli
};

// Çarpışma kontrol fonksiyonu
bool carpismaKontrol(const GameObject& obj1, const GameObject& obj2);

#endif // OYUN_HPP
```

Odev.cpp

```
#include "Oyun.hpp"

// Top sınıfı implementasyonu
Top::Top(float yaricap, sf::Color renk) {
    // Topun yarıçapını ve rengini ayarla
    top.setRadius(yaricap);
    top.setFillColor(renk);
    // Topun başlangıç konumunu ayarla
    top.setPosition(400, 300);
    // Konumu güncelle
    konum = top.getPosition();
}

void Top::hareket(Yon yon) {
    // Topun hareket yönüne göre konumunu güncelle
```

```
    switch (yon) {
    case Yon::SAG:
        konum.x += 5.0f;
        break;
    case Yon::SOL:
        konum.x -= 5.0f;
        break;
    case Yon::YUKARI:
        konum.y -= 5.0f;
        break;
    case Yon::ASAGI:
        konum.y += 5.0f;
        break;
    }
    // Topun yeni konumunu ayarla
    top.setPosition(konum);
}

sf::CircleShape& Top::getShape() {
    // Topun şekil referansını döndür
    return top;
}

const sf::CircleShape& Top::getShape() const {
    // Topun şekil referansını döndür (const versiyon)
    return top;
}

sf::Vector2f Top::getKonum() const {
    // Topun konumunu döndür
```

```
    return konum;
}

// Engel sınıfı implementasyonu
Engel::Engel(float genislik, float yukseklik, sf::Color renk) {
    // Engel boyutlarını ve rengini ayarla
    engel.setSize(sf::Vector2f(genislik, yukseklik));
    engel.setFillColor(renk);
    // Engel başlangıç konumunu rastgele ayarla
    engel.setPosition(static_cast<float>(rand() % 750), 0.0f);
}

void Engel::hareket(Yon yon) {
    // Engeli aşağı doğru hareket ettir
    engel.move(0, 5);
}

sf::RectangleShape& Engel::getShape() {
    // Engel şekil referansını döndür
    return engel;
}

const sf::RectangleShape& Engel::getShape() const {
    // Engel şekil referansını döndür (const versiyon)
    return engel;
}

sf::Vector2f Engel::getKonum() const {
    // Engel konumunu döndür
    return engel.getPosition();
```

```
}

// Yıldız sınıfı implementasyonu
Yildiz::Yildiz(float yaricap, sf::Color renk) {
    // Yıldızın yarıçapını ve rengini ayarla
    yildiz.setRadius(yaricap);
    yildiz.setFillColor(renk);
    // Yıldızın başlangıç konumunu rastgele ayarla
    yildiz.setPosition(static_cast<float>(rand() % 750), 0.0f);
}

void Yildiz::hareket(Yon yon) {
    // Yıldızı aşağı doğru hareket ettir
    yildiz.move(0, 5);
}

sf::CircleShape& Yildiz::getShape() {
    // Yıldız şekil referansını döndür
    return yildiz;
}

const sf::CircleShape& Yildiz::getShape() const {
    // Yıldız şekil referansını döndür (const versiyon)
    return yildiz;
}

sf::Vector2f Yildiz::getKonum() const {
    // Yıldız konumunu döndür
    return yildiz.getPosition();
}
```

```
// Çarpışma kontrol fonksiyonu
bool carpismaKontrol(const GameObject& obj1, const GameObject& obj2) {
    // İki oyun objesinin çarpışıp çarpışmadığını kontrol et
    return obj1.getShape().getGlobalBounds().intersects(obj2.getShape().getGlobalBounds());
}

// === Ana Program ===
int main() {
    int puan = 0; // Oyun puanı
    int yildizPuan = 0; // Yıldız puanı

    try {
        // Pencere oluştur
        sf::RenderWindow pencere(sf::VideoMode(800, 600), "Top ve Engel Oyunu");

        // Saatler oluştur
        sf::Clock saat;
        sf::Clock oyunSuresi;
        sf::Clock engelSaat;
        sf::Clock yildizSaat;
        sf::Clock renkDegisimSaat;

        bool beyazEkran = true; // Ekran rengi durumu

        // Top objesi oluştur
        std::unique_ptr<GameObject> top = std::make_unique<Top>(20, sf::Color::Green);
        // Engel ve yıldız vektörleri oluştur
        std::vector<std::unique_ptr<GameObject>> engeller;
        std::vector<std::unique_ptr<GameObject>> yildizlar;
```

```
    float engelOlusturmaSuresi = 1.0f; // Engel oluşturma süresi
    Yon topYonu = Yon::SAG; // Topun başlangıç yönü

    while (pencere.isOpen()) {
        sf::Event olay;
        while (pencere.pollEvent(olay)) {
            if (olay.type == sf::Event::Closed)
                pencere.close();
        }

        // Klavye girdilerini kontrol et
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) topYonu = Yon::SOL;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) topYonu = Yon::SAG;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) topYonu = Yon::YUKARI;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) topYonu = Yon::ASAGI;

        // Topu hareket ettir
        if (saat.getElapsedTime().asSeconds() >= 0.01f) {
            top->hareket(topYonu);

            // Topun oyun alanı dışına çıkıp çıkmadığını kontrol et
            if (top->getKonum().x < 0 || top->getKonum().x > 800 || top->getKonum().y < 0 ||
top->getKonum().y > 600) {
                throw std::runtime_error("Oyun Bitti! Top oyun alani disina cikti!");
            }
            saat.restart();
        }

        // Puanı oyun süresine eşitle
        puan = static_cast<int>(oyunSuresi.getElapsedTime().asSeconds());
```

```
        // Engel oluşturma süresi dolduysa yeni engeller oluştur
        if (engelSaat.getElapsedTime().asSeconds() >= engelOlusturmaSuresi) {
            if (puan >= 90) {
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
            }
            else if (puan >= 30) {
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
            }
            else {
                engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
            }
            engelSaat.restart();
        }

        // Engelleri hareket ettir
        for (auto& engel : engeller) {
            engel->hareket(Yon::ASAGI);
        }

        // Oyun alanı dışına çıkan engelleri sil
        engeller.erase(std::remove_if(engeller.begin(), engeller.end(),
            [](const std::unique_ptr<GameObject>& engel) { return engel->getKonum().y >
600; }),
            engeller.end());

        // Top ile engellerin çarpışmasını kontrol et
        for (const auto& engel : engeller) {
```

```
            if (carpismaKontrol(*top, *engel)) {
                throw std::runtime_error("Oyun Bitti! Top bir engele carpti!");
            }
        }

        // Yıldız oluşturma süresi dolduysa yeni yıldızlar oluştur
        if (yildizSaat.getElapsedTime().asSeconds() >= 30.0f) {
            yildizlar.push_back(std::make_unique<Yildiz>(10, sf::Color::Blue));
            yildizSaat.restart();
        }

        // Yıldızları hareket ettir
        for (auto& yildiz : yildizlar) {
            yildiz->hareket(Yon::ASAGI);
        }

        // Oyun alanı dışına çıkan yıldızları sil
        yildizlar.erase(std::remove_if(yildizlar.begin(), yildizlar.end(),
            [](const std::unique_ptr<GameObject>& yildiz) { return yildiz->getKonum().y >
600; }),
            yildizlar.end());

        // Top ile yıldızların çarpışmasını kontrol et
        for (auto it = yildizlar.begin(); it != yildizlar.end();) {
            if (carpismaKontrol(*top, **it)) {
                yildizPuan += 30; // Yıldıza çarpınca yıldız puanını artır
                it = yildizlar.erase(it);
            }
            else {
                ++it;
            }
```

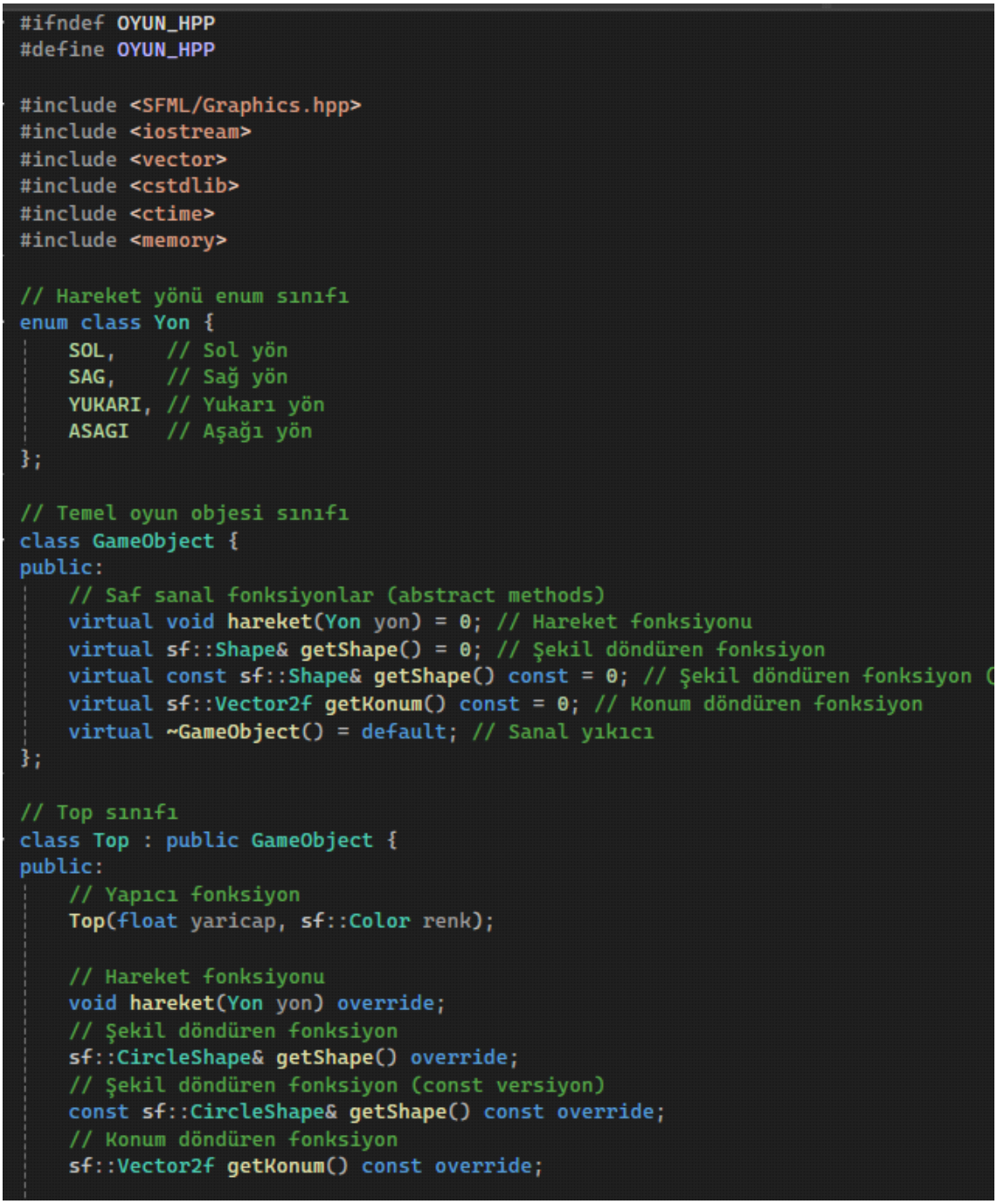
```
        }

        // Ekran rengini değiştirme süresi dolduysa ekran rengini değiştir
        if (renkDegisimSaat.getElapsedTime().asSeconds() >= 60.0f) {
            beyazEkran = !beyazEkran;
            renkDegisimSaat.restart();
        }

        // Ekranı temizle ve yeni rengi uygula
        pencere.clear(beyazEkran ? sf::Color::White : sf::Color::Black);
        // Topu çiz
        pencere.draw(top->getShape());
        // Engelleri çiz
        for (const auto& engel : engeller) {
            pencere.draw(engel->getShape());
        }
        // Yıldızları çiz
        for (const auto& yildiz : yildizlar) {
            pencere.draw(yildiz->getShape());
        }
        // Ekranı güncelle
        pencere.display();
    }
}
catch (const std::exception& e) {
    // Hata mesajını ve puanları yazdır
    std::cerr << e.what() << std::endl;
    std::cerr << "Puaniniz: " << puan << std::endl;
    std::cerr << "Yildiz Puaniniz: " << yildizPuan << std::endl;
}
```

```
    return 0;
}
```

Kodların Resmi



```hpp
#ifndef OYUN_HPP
#define OYUN_HPP

#include <SFML/Graphics.hpp>
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <memory>

// Hareket yönü enum sınıfı
enum class Yon {
    SOL,    // Sol yön
    SAG,    // Sağ yön
    YUKARI, // Yukarı yön
    ASAGI   // Aşağı yön
};

// Temel oyun objesi sınıfı
class GameObject {
public:
    // Saf sanal fonksiyonlar (abstract methods)
    virtual void hareket(Yon yon) = 0; // Hareket fonksiyonu
    virtual sf::Shape& getShape() = 0; // Şekil döndüren fonksiyon
    virtual const sf::Shape& getShape() const = 0; // Şekil döndüren fonksiyon (
    virtual sf::Vector2f getKonum() const = 0; // Konum döndüren fonksiyon
    virtual ~GameObject() = default; // Sanal yıkıcı
};

// Top sınıfı
class Top : public GameObject {
public:
    // Yapıcı fonksiyon
    Top(float yaricap, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
    sf::CircleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::CircleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;
```

```
private:
    sf::CircleShape top; // Topun şekli
    sf::Vector2f konum;  // Topun konumu
};

// Engel sınıfı
class Engel : public GameObject {
public:
    // Yapıcı fonksiyon
    Engel(float genislik, float yukseklik, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
    sf::RectangleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::RectangleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;

private:
    sf::RectangleShape engel; // Engel şekli
};

// Yıldız sınıfı
class Yildiz : public GameObject {
public:
    // Yapıcı fonksiyon
    Yildiz(float yaricap, sf::Color renk);

    // Hareket fonksiyonu
    void hareket(Yon yon) override;
    // Şekil döndüren fonksiyon
    sf::CircleShape& getShape() override;
    // Şekil döndüren fonksiyon (const versiyon)
    const sf::CircleShape& getShape() const override;
    // Konum döndüren fonksiyon
    sf::Vector2f getKonum() const override;

private:
    sf::CircleShape yildiz; // Yıldız şekli
};
```

```cpp
// Çarpışma kontrol fonksiyonu
bool carpismaKontrol(const GameObject& obj1, const GameObject& obj2);

#endif // OYUN_HPP
```

```cpp
#include "Oyun.hpp"

// Top sınıfı implementasyonu
Top::Top(float yaricap, sf::Color renk) {
    // Topun yarıçapını ve rengini ayarla
    top.setRadius(yaricap);
    top.setFillColor(renk);
    // Topun başlangıç konumunu ayarla
    top.setPosition(400, 300);
    // Konumu güncelle
    konum = top.getPosition();
}

void Top::hareket(Yon yon) {
    // Topun hareket yönüne göre konumunu güncelle
    switch (yon) {
    case Yon::SAG:
        konum.x += 5.0f;
        break;
    case Yon::SOL:
        konum.x -= 5.0f;
        break;
    case Yon::YUKARI:
        konum.y -= 5.0f;
        break;
    case Yon::ASAGI:
        konum.y += 5.0f;
        break;
    }
    // Topun yeni konumunu ayarla
    top.setPosition(konum);
}

sf::CircleShape& Top::getShape() {
    // Topun şekil referansını döndür
    return top;
}

const sf::CircleShape& Top::getShape() const {
    // Topun şekil referansını döndür (const versiyon)
    return top;
}
```

```cpp
sf::Vector2f Top::getKonum() const {
    // Topun konumunu döndür
    return konum;
}

// Engel sınıfı implementasyonu
Engel::Engel(float genislik, float yukseklik, sf::Color renk) {
    // Engel boyutlarını ve rengini ayarla
    engel.setSize(sf::Vector2f(genislik, yukseklik));
    engel.setFillColor(renk);
    // Engel başlangıç konumunu rastgele ayarla
    engel.setPosition(static_cast<float>(rand() % 750), 0.0f);
}

void Engel::hareket(Yon yon) {
    // Engeli aşağı doğru hareket ettir
    engel.move(0, 5);
}

sf::RectangleShape& Engel::getShape() {
    // Engel şekil referansını döndür
    return engel;
}

const sf::RectangleShape& Engel::getShape() const {
    // Engel şekil referansını döndür (const versiyon)
    return engel;
}

sf::Vector2f Engel::getKonum() const {
    // Engel konumunu döndür
    return engel.getPosition();
}

// Yıldız sınıfı implementasyonu
Yildiz::Yildiz(float yaricap, sf::Color renk) {
    // Yıldızın yarıçapını ve rengini ayarla
    yildiz.setRadius(yaricap);
    yildiz.setFillColor(renk);
    // Yıldızın başlangıç konumunu rastgele ayarla
    yildiz.setPosition(static_cast<float>(rand() % 750), 0.0f);
}
```

```cpp
void Yildiz::hareket(Yon yon) {
    // Yıldızı aşağı doğru hareket ettir
    yildiz.move(0, 5);
}

sf::CircleShape& Yildiz::getShape() {
    // Yıldız şekil referansını döndür
    return yildiz;
}

const sf::CircleShape& Yildiz::getShape() const {
    // Yıldız şekil referansını döndür (const versiyon)
    return yildiz;
}

sf::Vector2f Yildiz::getKonum() const {
    // Yıldız konumunu döndür
    return yildiz.getPosition();
}

// Çarpışma kontrol fonksiyonu
bool carpismaKontrol(const GameObject& obj1, const GameObject& obj2) {
    // İki oyun objesinin çarpışıp çarpışmadığını kontrol et
    return obj1.getShape().getGlobalBounds().intersects(obj2.getShape().getGloba
}

// === Ana Program ===
int main() {
    int puan = 0; // Oyun puanı
    int yildizPuan = 0; // Yıldız puanı

    try {
        // Pencere oluştur
        sf::RenderWindow pencere(sf::VideoMode(800, 600), "Top ve Engel Oyunu");

        // Saatler oluştur
        sf::Clock saat;
        sf::Clock oyunSuresi;
        sf::Clock engelSaat;
        sf::Clock yildizSaat;
        sf::Clock renkDegisimSaat;

        bool beyazEkran = true; // Ekran rengi durumu
```

```cpp
// Top objesi oluştur
std::unique_ptr<GameObject> top = std::make_unique<Top>(20, sf::Color::Green);
// Engel ve yıldız vektörleri oluştur
std::vector<std::unique_ptr<GameObject>> engeller;
std::vector<std::unique_ptr<GameObject>> yildizlar;

float engelOlusturmaSuresi = 1.0f; // Engel oluşturma süresi
Yon topYonu = Yon::SAG; // Topun başlangıç yönü

while (pencere.isOpen()) {
    sf::Event olay;
    while (pencere.pollEvent(olay)) {
        if (olay.type == sf::Event::Closed)
            pencere.close();
    }

    // Klavye girdilerini kontrol et
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) topYonu = Yon::SOL;
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) topYonu = Yon::SAG;
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) topYonu = Yon::YUKARI;
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) topYonu = Yon::ASAGI;

    // Topu hareket ettir
    if (saat.getElapsedTime().asSeconds() >= 0.01f) {
        top->hareket(topYonu);

        // Topun oyun alanı dışına çıkıp çıkmadığını kontrol et
        if (top->getKonum().x < 0 || top->getKonum().x > 800 || top->getKonum().y < 0 || top->getKonum().y > 600) {
            throw std::runtime_error("Oyun Bitti! Top oyun alani disina cikti!");
        }
        saat.restart();
    }

    // Puanı oyun süresine eşitle
    puan = static_cast<int>(oyunSuresi.getElapsedTime().asSeconds());

    // Engel oluşturma süresi dolduysa yeni engeller oluştur
    if (engelSaat.getElapsedTime().asSeconds() >= engelOlusturmaSuresi) {
        if (puan >= 90) {
            engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
            engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
            engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
        }
```

```
    else if (puan >= 30) {
        engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
        engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
    }
    else {
        engeller.push_back(std::make_unique<Engel>(50, 50, sf::Color::Red));
    }
    engelSaat.restart();
}

// Engelleri hareket ettir
for (auto& engel : engeller) {
    engel->hareket(Yon::ASAGI);
}

// Oyun alanı dışına çıkan engelleri sil
engeller.erase(std::remove_if(engeller.begin(), engeller.end(),
    [](const std::unique_ptr<GameObject>& engel) { return engel->getKonum().y > 600; }),
    engeller.end());

// Top ile engellerin çarpışmasını kontrol et
for (const auto& engel : engeller) {
    if (carpismaKontrol(*top, *engel)) {
        throw std::runtime_error("Oyun Bitti! Top bir engele carpti!");
    }
}

// Yıldız oluşturma süresi dolduysa yeni yıldızlar oluştur
if (yildizSaat.getElapsedTime().asSeconds() >= 30.0f) {
    yildizlar.push_back(std::make_unique<Yildiz>(10, sf::Color::Blue));
    yildizSaat.restart();
}

// Yıldızları hareket ettir
for (auto& yildiz : yildizlar) {
    yildiz->hareket(Yon::ASAGI);
}

// Oyun alanı dışına çıkan yıldızları sil
yildizlar.erase(std::remove_if(yildizlar.begin(), yildizlar.end(),
    [](const std::unique_ptr<GameObject>& yildiz) { return yildiz->getKonum().y > 600; }),
    yildizlar.end());
```

```
            // Top ile yıldızların çarpışmasını kontrol et
            for (auto it = yildizlar.begin(); it != yildizlar.end();) {
                if (carpismaKontrol(*top, **it)) {
                    yildizPuan += 30; // Yıldıza çarpınca yıldız puanını artır
                    it = yildizlar.erase(it);
                }
                else {
                    ++it;
                }
            }

            // Ekran rengini değiştirme süresi dolduysa ekran rengini değiştir
            if (renkDegisimSaat.getElapsedTime().asSeconds() >= 60.0f) {
                beyazEkran = !beyazEkran;
                renkDegisimSaat.restart();
            }

            // Ekranı temizle ve yeni rengi uygula
            pencere.clear(beyazEkran ? sf::Color::White : sf::Color::Black);
            // Topu çiz
            pencere.draw(top->getShape());
            // Engelleri çiz
            for (const auto& engel : engeller) {
                pencere.draw(engel->getShape());
            }
            // Yıldızları çiz
            for (const auto& yildiz : yildizlar) {
                pencere.draw(yildiz->getShape());
            }
            // Ekranı güncelle
            pencere.display();
        }
    }
    catch (const std::exception& e) {
        // Hata mesajını ve puanları yazdır
        std::cerr << e.what() << std::endl;
        std::cerr << "Puaniniz: " << puan << std::endl;
        std::cerr << "Yildiz Puaniniz: " << yildizPuan << std::endl;
    }

    return 0;
}
```

BAŞVURULAN KAYNAKLAR

https://youtu.be/HUONtLEAL8g?si=yuJLwaYUYLNHe2be

https://youtu.be/lnIUa1V9o-U?si=S2MCvZRUnflBkJqh

https://youtu.be/GmujGT06BpU?si=pNvAZAC3Eorew16F

https://youtu.be/4vm_E4WPtGo?si=6gUHTLzewJttXZVa

https://youtu.be/eoK0yJyFWV4?si=aIOycwMuqP_3tssf

https://youtu.be/Pz13dafHwSk?si=Y9iIAFSnlEgdxSAX

https://youtu.be/ufNhA249PoA?si=pPBuLISmpChTyFnf

https://youtu.be/qWqB_Oj8nB8?si=1lHnt6yzVjcO6aGF

https://youtu.be/OQMl-u0vF3Q?si=TcZz6j9etoQrqvGq