adım adım yapılması gerekenleri anlatırmısın?

###### **ChatGPT:**

ChatGPT

Tabii! AI agent'ınızı ve oyun modülünüzü ayrı geliştirerek entegre bir oyun yapısına ulaşmak için izlemeniz gereken adımları adım adım anlatayım.

### **1. Proje Yapısının Kurulması**

* **Proje Modülerliği**: Başlangıç olarak, projenizin modüler bir yapıda olmasını hedeflemelisiniz. Bu, oyun modülü ve AI modülünü ayrı dosya veya klasör yapılarında tutmanızı sağlar.
  + **Oyun Modülü**: Tüm sahne yönetimi, karakterler, animasyonlar ve oyun mekaniklerini içeren ana modül.
  + **AI Modülü**: Yapay zekanın eğitileceği simülasyon ve ML-Agents aracılığıyla öğrenme süreciyle ilgili kodlar burada yer alır.
* **Dosya/Klasör Yapısı**:
  + Assets/GameModule/ – Oyunla ilgili tüm bileşenler.
  + Assets/AIModule/ – AI ile ilgili her şey (ML-Agents, eğitim ortamı, model dosyaları).

### **2. Oyun Mekaniklerinin Geliştirilmesi**

* **Sahne ve Oyun Nesneleri (GameObjects)**:
  + Oyunun temel yapısını oluşturun. Karakterler, oyuncu kontrolleri, animasyonlar, çevresel etkileşimler ve oyun mekanikleri burada yer alır.
  + Bu aşamada yapay zekayı devreye sokmadan, oyunun temel akışını kurmalısınız (örneğin: platformlar, düşmanlar, eylemler).
* **Kullanıcı Arayüzü (UI)**:
  + Oyunun temel kullanıcı arayüzü (puan, can, görevler gibi) bu modülde yer alır.

**Örnek**: Bir platform oyunu geliştiriyorsanız, karakterin hareket ettiği, engellerin olduğu, belirli hedeflere ulaşılması gereken temel oyun akışını bu adımda inşa edin.

### **3. ML-Agents ve AI Eğitim Ortamının Hazırlanması**

* **ML-Agents Kurulumu**: Unity ML-Agents araç setini proje dosyanıza ekleyin.
  + Unity Package Manager üzerinden ML-Agents paketini yükleyin.
  + Python ortamınızı kurun ve ML-Agents’ın Python SDK’sını yükleyin (gerekli Python kütüphaneleri ile).
  + [ML-Agents GitHub Sayfası](https://github.com/Unity-Technologies/ml-agents)
* **AI Eğitim Ortamı Yaratma**:
  + AI’nın eğitileceği bir sahne yaratın. Bu sahne, yapay zekanın öğrenme süreci için optimize edilmelidir.
  + Basit geometrik şekiller ve engeller gibi, ajanların belirli hedeflere ulaşması gereken bir ortam tasarlayın.
* **Agent Script’i Yazma**: AI agent için gerekli kodları yazın.
  + ML-Agents’ın Agent sınıfını miras alan bir C# script yazın. Bu script, agent’ın oyundaki hareketlerini ve kararlarını yönetir.
  + Agent’ın nasıl hareket edeceğini ve hangi koşullarda ödüller kazanacağını tanımlayın.

csharp

Kodu kopyala

using Unity.MLAgents;  
using Unity.MLAgents.Sensors;  
  
public class MyAgent : Agent  
{  
 public override void OnEpisodeBegin()  
 {  
 // Ajanın yeni bir bölüm başladığında ne yapacağını tanımlayın  
 }  
  
 public override void CollectObservations(VectorSensor sensor)  
 {  
 // Çevredeki önemli bilgileri toplayın (mesafe, konum vs.)  
 }  
  
 public override void OnActionReceived(float[] vectorAction)  
 {  
 // Ajanın aldığı aksiyonları tanımlayın (hareket et, dön, zıpla vs.)  
 }  
}

### **4. AI Eğitimi (Training)**

* **Eğitim Sahnesini Çalıştırın**: AI eğitim ortamını başlatın. Ajanın çevreyi keşfetmesi ve deneme yanılma yoluyla öğrenmesi için gerekli senaryoları sağlayın.
* **Pekiştirmeli Öğrenme (Reinforcement Learning)**: Ajanın aldığı ödülleri (rewards) ve cezaları (penalties) tanımlayın. İstenilen davranışa göre ajan ödüller alarak doğru kararları öğrenir.
* **Model Eğitimini Başlatın**:
  + Python SDK’yı kullanarak Unity ortamında ajanların eğitimine başlayın.
  + Komut satırından eğitimi başlatın ve ajanınızın performansını izleyin.
  + Eğitim sırasında model dosyası (.nn) oluşturulacaktır.

bash

Kodu kopyala

mlagents-learn config.yaml --run-id=training-run

### **5. Eğitim Sonucunun İncelenmesi ve Modeli Kullanma**

* Eğitim sonrasında .nn model dosyası elde edilecektir. Bu modeli, eğitim tamamlandığında oyunda kullanabilirsiniz.
* **Modelin Oyuna Eklenmesi**:
  + AI modülündeki Agent script'inize, eğitilen .nn model dosyasını atayın. Bu şekilde, ajan artık eğitildiği model üzerinden karar verecektir.

csharp

Kodu kopyala

public NNModel trainedModel;  
private BehaviorParameters behaviorParameters;  
  
void Start()  
{  
 behaviorParameters = GetComponent<BehaviorParameters>();  
 behaviorParameters.Model = trainedModel;  
}

### **6. AI Agent'ı Oyun Modülüne Entegre Etme**

* **AI’nın Oyunla Etkileşimi**:
  + AI agent’ın oyundaki diğer nesnelerle nasıl etkileşime gireceğini tanımlayın. Oyuncunun yanında hareket edebilir, belirli hedeflere ulaşabilir veya düşman olabilir.
  + AI’nın oyun mekanikleriyle etkileşimde bulunması için gerekli event’leri ve komutları sağlayın.
* **Animasyon ve Ses**:
  + Eğitilen AI agent’ınızın görsel ve sesli temsili için animasyon ve ses entegre edebilirsiniz.
  + Örneğin, karakterinizin bir yandan hareket ederken diğer yandan sesli açıklama yapmasını sağlayabilirsiniz.

### **7. Test Etme ve İyileştirme**

* AI ve oyun modüllerini entegre ettikten sonra, test edin.
* **Sorun Giderme**: AI’nın eğitimi sırasında veya oyuna entegrasyonunda ortaya çıkan hataları inceleyin.
* **İyileştirme**: AI performansını geliştirmek ve oyunla daha iyi uyumlu hale getirmek için gerekirse eğitim sürecini tekrar başlatın ve ayarları ince ayar yapın.

### **8. Son Optimizasyon ve Yayınlama**

* **Optimizasyon**: Hem oyun modülünün hem de AI agent modülünün performansını optimize edin. AI’nın eğitim sürecinde zaman alan bölümleri sadeleştirin.
* **Platform Testleri**: Oyunu ve AI’yı hedeflediğiniz platformlarda (PC, mobil, konsol) test edin.
* **Yayınlama**: Projenizi son haline getirin ve yayınlayın.

### **Özet:**

Bu adımlar sayesinde oyununuzu ve AI agent’ınızı ayrı ayrı geliştirebilir, sonrasında da birbirleriyle etkileşimli hale getirebilirsiniz. Her iki modülü ayrı geliştirmek, projeyi daha modüler ve esnek hale getirir.

4o

##### **Şunu dedin:**

benim düşüncem karakterin hareketlerine göre yapay zekanın uyarı vermesi ve öncesinde bir eğitim vermesiydi oyun sonunda da oyuncuyu değerlendiirmesiydi.

###### **ChatGPT:**

ChatGPT

Bellek güncellendi

Bu yaklaşım oldukça ilgi çekici ve eğitim odaklı bir oyun için çok uygun! Karakterin hareketlerine göre uyarı veren ve oyun sonunda bir değerlendirme sunan bir yapay zeka (AI) sistemini adım adım nasıl kurabileceğinizi anlatayım:

### **1. Karakter Hareketlerinin Takip Edilmesi**

* **Oyun Karakteri ve Hareketleri**:
  + Oyuncu karakterinin hareketlerini izlemek için oyun içindeki kontrol sistemini kurun. Bu, klavye ya da gamepad ile yapılan girişleri takip ederek karakterin pozisyonunu ve eylemlerini kaydetmeyi içerir.
  + Karakterin koşma, zıplama, düşme gibi farklı eylemlerini tanımlayın.
* **Hareket Algılama**:
  + Karakterin hareketlerini izlemek ve belirli bir davranışa göre (örneğin, tehlikeli bir hareket yapma) AI'nın uyarı vermesini sağlamak için bir mekanizma oluşturun.
  + Örneğin, karakter yüksekten atladığında, yapay zeka hemen bir uyarı verebilir: “Yüksekten düşmeler tehlikelidir, dikkat et!”

csharp

Kodu kopyala

void Update()  
{  
 if (character.IsJumpingFromHigh())  
 {  
 AIManager.TriggerWarning("Yüksekten düşme tehlikesi!");  
 }  
}

### **2. Eğitim Süreci Tasarımı**

* **Başlangıç Eğitimi**:
  + Oyunun başında oyuncuya temel güvenlik bilgilerini veren bir eğitim modülü ekleyin. Bu, bir yapay zeka karakteri aracılığıyla olabilir. AI karakteri sesli ya da yazılı bir şekilde oyuncuya rehberlik eder.
  + Eğitim sırasında, oyuncunun hareketlerini gözlemleyerek doğru eylemleri yapıp yapmadığını kontrol eden bir sistem kurun. Yanlış yapıldığında uyarılar verebilir.

**Örnek Eğitim Senaryosu**:

* Oyuncuya güvenli hareket etme, yüksekten düşme riskleri, tehlikeli objelere dokunmama gibi temel kurallar öğretilir.
* AI karakteri, oyuncuya “Bu alandan geçerken yavaş hareket etmelisin” gibi talimatlar verir.

csharp

Kodu kopyala

public void StartTraining()  
{  
 AIManager.DisplayMessage("Yüksekten atlamamalısın, dikkatli ol!");  
}

### **3. Uyarı Sistemi**

* **Anında Uyarı**:
  + Oyuncu tehlikeli ya da yanlış bir hareket yaptığında AI karakteri devreye girerek uyarı verir.
  + Uyarılar sesli ya da yazılı olabilir. AI robotu ekranda belirip oyuncuya bir tavsiye verebilir.
* **Uyarı Verme Kriterleri**:
  + Örneğin, oyuncu bir hatalı eylemi birkaç kez tekrarladığında uyarı daha da belirgin hale gelebilir. Örneğin: "Üç kere yanlış hareket yaptın, dikkatli ol!"

csharp

Kodu kopyala

if (dangerousMoveCounter > 3)  
{  
 AIManager.TriggerWarning("Tehlikeli hareketler yapıyorsun, dikkat et!");  
}

### **4. Oyuncu Performansını Değerlendirme**

* **Oyun Sonu Değerlendirmesi**:
  + Oyuncu oyunu tamamladığında, AI agent tarafından bir performans değerlendirmesi yapılır. Bu değerlendirme, oyuncunun kaç kez tehlikeli hareket yaptığı, kaç uyarı aldığı ve eğitimi ne kadar iyi uyguladığına göre olabilir.
  + Performansa dayalı bir puanlama ya da geri bildirim verebilirsiniz.
* **Performans Kriterleri**:
  + Kaç uyarı alındı?
  + Kaç doğru hareket yapıldı?
  + Eğitimde ne kadar başarılı olundu?

**Örnek**:

* Oyuncuya eğitimde ne kadar başarılı olduğuna dair bir skor verilir: “Güvenlik kurallarına %80 oranında uyum sağladın, harika iş çıkardın!”

csharp

Kodu kopyala

public void EvaluatePlayer()  
{  
 int score = CalculatePlayerScore();  
 AIManager.DisplayFinalScore(score);  
}  
  
private int CalculatePlayerScore()  
{  
 // Oyuncunun yaptığı hatalar ve doğru eylemlere göre puan hesaplanır  
 return (correctMoves - wrongMoves) \* 10;  
}

### **5. Yapay Zeka Karakterinin Görsel ve Sesli Temsili**

* **Görsel Temsil**:
  + Yapay zeka robotunu ekranda bir karakter olarak gösterin. Bu karakter, oyuncu hareket ettiğinde ya da bir uyarı verdiğinde ekranda belirip rehberlik yapabilir.
  + **Animasyonlar**: Yapay zeka karakterinin konuşurken ya da uyarı verirken hareket etmesini sağlayan animasyonlar ekleyin.
* **Sesli Anlatım**:
  + AI karakterinin sesli olarak da uyarılar vermesini sağlayın. Unity’nin ses sistemini kullanarak karakterin konuşmasını ayarlayabilirsiniz. Örneğin: "Dikkat et, tehlikeli bir hareket yaptın!"

**Unity AudioSystem Örneği**:

csharp

Kodu kopyala

public AudioSource aiVoice;  
  
void TriggerWarning(string warningMessage)  
{  
 aiVoice.Play();  
 DisplayWarningOnScreen(warningMessage);  
}

### **6. Test ve İyileştirme**

* **Oyun İçi Testler**: Oyuncu hareketlerine ve AI’nın verdiği uyarılara dikkat edin. Sistem düzgün çalışıyor mu, AI doğru zamanda uyarı veriyor mu gibi noktaları test edin.
* **Performans İyileştirme**: AI’nın verdiği uyarılar ve oyun sonu değerlendirmesi çok katı veya çok yumuşaksa, ayarları ince ayar yaparak optimize edin.

Son önerdiğim sistemde **ML-Agents** kullanılmadı, çünkü bu sistemde karakterin hareketlerine dayalı olarak yapay zekanın anlık tepkiler vermesi ve oyunu değerlendirmesi daha çok belirli kural bazlı bir yapı üzerine kurulu. Bu tür bir yapıda, ML-Agents gibi bir makine öğrenimi aracına ihtiyaç olmadan klasik programlama mantığıyla yapay zekayı geliştirebilirsiniz.

Ancak eğer karakterin daha karmaşık ve öğrenmeye dayalı kararlar vermesini (örneğin, oyuncunun hareketlerine göre zamanla strateji geliştiren bir yapay zeka) isterseniz, **ML-Agents**'ı kullanabilirsiniz.

### **ML-Agents'ı Kullanmak İçin Ne Zaman Gereklidir?**

ML-Agents, yapay zekanın kendi başına öğrenme ve karar verme yeteneğini kazandığı daha dinamik ve kompleks sistemlerde devreye girer. Örneğin:

* AI, oyuncunun davranışlarını analiz edip yeni stratejiler öğrenebilir.
* Oyuncunun oynama tarzına göre uyum sağlayan, esnek ve öğrenen bir yapay zeka oluşturabilirsiniz.
* Oyuncunun performansını değerlendirirken çok daha karmaşık hareket ve eylemleri dikkate alacak bir sistem oluşturabilirsiniz.

### **ML-Agents Kullanımını Dahil Etmek için Adımlar:**

Eğer oyundaki yapay zekanın zamanla daha "öğrenen" ve gelişen bir davranış sergilemesini isterseniz, şu ek adımları ekleyebilirsiniz:

1. **ML-Agents Kurulumu ve Eğitim Ortamı**:
   1. AI karakterini, oyuncunun hareketlerini gözlemleyip, oyuncunun daha güvenli hareket etmesi için eğitim veren bir ajan haline getirebilirsiniz.
   2. AI, oyuncunun nasıl hareket ettiğini öğrenerek, gelecekteki uyarılarını optimize edebilir. Yani, belirli bir oyuncu davranışını analiz ederek hangi tür uyarıların etkili olduğunu öğrenir.
2. **Agent’in Eğitilmesi**:
   1. **Geri Bildirim Sistemi**: AI, oyuncuya uyarılar verirken ne tür uyarıların daha etkili olduğunu öğrenir ve bu süreçte daha "akıllı" hale gelir. Başarılı uyarılar için ödül, etkisiz uyarılar için ceza alarak, AI sistemi kendi davranışlarını optimize eder.
3. **Oyuna Entegre Etme**:
   1. Eğitilen AI agent'ı, oyuncunun hareketlerini gözlemleyip bu hareketlere göre dinamik uyarılar verir. Oyunun sonundaki değerlendirme sürecinde, oyuncunun performansını daha "öğrenen" bir bakış açısıyla değerlendirir.

Eğer daha temel kurallar üzerinden giden bir AI sistemi yeterliyse, **ML-Agents kullanmadan** önerdiğim çözüm yeterli olacaktır. Ancak daha karmaşık, öğrenmeye dayalı bir AI istiyorsanız **ML-Agents** entegrasyonunu düşünebilirsiniz.

ML-Agents kullanmadan oyununuza bir yapay zeka (AI) entegre etmek için, temelde programlama bilgisiyle bir kural bazlı sistem oluşturabilirsiniz. Bunun için gerekecek bilgi seviyeleri ve öğrenilmesi gereken konular şöyle sıralanabilir:

### **1. Unity Bilgisi**

* **Oyun Motorunu Anlamak**: Unity’de temel sahne yönetimi, nesne yerleştirme, kamera kullanımı ve sahneler arası geçiş gibi konularda deneyimli olmalısınız.
* **Script Yazma**: Unity'de **C#** dilini kullanarak oyun mekaniklerini kodlamayı bilmek gerekiyor. Unity’de karakter kontrolü, animasyon, fizik motoru gibi sistemlerle çalışacak şekilde kod yazabilmelisiniz.
* **UI Entegrasyonu**: AI’nın uyarıları verirken kullanacağı görsel mesajları ya da ekran üzerine gelen animasyonlu karakteri göstermek için Unity UI sistemini iyi anlamalısınız.

**Gerekli Seviye**: Orta düzey Unity deneyimi ve sahne yönetimi bilgisi.

### **2. C# Programlama**

* **Oyun Mantığını Programlama**: Karakterin hareketlerine dayalı olarak yapay zeka kuralları yazmak için güçlü bir C# programlama bilgisine sahip olmanız gerekir. Özellikle olay tabanlı programlama, if-else yapıları, fonksiyonlar, sınıflar, ve nesne tabanlı programlama önemli.
* **State Management (Durum Yönetimi)**: Yapay zekanın farklı durumlara (oyuncunun hareketi gibi) tepki vermesi için bir durum yönetimi sistemi kurmanız gerekebilir.

**Gerekli Seviye**: Orta-ileri düzey C# bilgisi.

### **3. Yapay Zeka (Kural Bazlı)**

* **Kural Tabanlı AI Sistemleri**: AI davranışlarını kural tabanlı yazabilmek için öncelikle mantık bazlı sistemler (if-else koşulları, duruma dayalı tepkiler) hakkında bilgi sahibi olmalısınız. Örneğin, karakter yüksekten düştüğünde uyarı verecek basit bir sistem kurmak yeterlidir.
* **Finite State Machines (FSM)**: FSM kullanarak AI’nın farklı davranışlarını tanımlayabilirsiniz. FSM, yapay zekanın belirli koşullara göre farklı durumlardan geçmesini sağlar (örneğin, eğitim, uyarı verme, değerlendirme durumu).

**Gerekli Seviye**: Temel yapay zeka ve FSM mantığını anlamak.

### **4. Oyun Fiziği ve Karakter Kontrolü**

* **Karakterin Hareketleri ve Fizik Etkileşimleri**: Karakterin hareketlerini ve çevreyle etkileşimlerini anlayabilmeniz için Unity’nin fizik motorunu ve karakter kontrol sistemlerini bilmelisiniz. Hareketlerin doğru bir şekilde algılanıp yapay zekanın buna tepki vermesini sağlamak önemlidir.

**Gerekli Seviye**: Temel Unity fizik bilgisi (karakter hareketi, çarpışma algılama).

### **5. Ses ve Animasyon Sistemleri**

* **Sesli Geri Bildirim**: Yapay zeka sesli olarak uyarılar verecekse, Unity’nin ses sistemi ile çalışmayı öğrenmelisiniz. Karakterin uyarı anında doğru ses dosyasını oynatmasını sağlayabilmelisiniz.
* **Animasyonlar**: Yapay zeka karakterinin ekrandaki görsel temsili için basit animasyonlar eklemek gerekebilir. Unity’de animasyon sistemiyle çalışabilmek (Animator Controller kullanımı) önemli olacaktır.

**Gerekli Seviye**: Temel Unity ses ve animasyon bilgisi.

### **6. Geri Bildirim ve Değerlendirme Sistemleri**

* **Uyarı Sistemi**: Oyun sırasında yapay zeka oyuncunun yaptığı hatalara karşı uyarı verir. Bu uyarıların ekranda görsel olarak gösterilmesi ya da sesli şekilde verilmesi için uygun mekanizmaları kurmak önemlidir.
* **Performans Değerlendirme**: Oyun sonunda, oyuncunun performansını değerlendirip buna göre puan ya da geri bildirim verecek bir sistem geliştirmek için oyun verilerini (kaç hata yapıldığı, kaç uyarı alındığı gibi) saklamayı öğrenmelisiniz.

**Gerekli Seviye**: Temel veri yönetimi ve oyun içi geri bildirim sistemi bilgisi.

### **7. Oyun Testi ve Optimizasyonu**

* **Yapay Zeka Davranışlarını Test Etme**: Yapay zekanın doğru çalışıp çalışmadığını test etmek, oyuncu davranışlarına doğru tepkiler verdiğinden emin olmak için test yapmayı öğrenmelisiniz.
* **Optimizasyon**: Oyun içi AI'nın performansını test etmek ve oyunun akıcılığını korumak için optimizasyon tekniklerini öğrenmelisiniz.

**Gerekli Seviye**: Temel oyun testi ve optimizasyon bilgisi.

### **Gerekli Temel Beceriler:**

1. **Orta-İleri Seviye C# Programlama**: Oyun mekaniğini ve yapay zeka davranışlarını kodlayabilmek için.
2. **Orta Seviye Unity Bilgisi**: Oyun ortamını oluşturabilmek, sahne yönetimini ve UI sistemini kullanabilmek için.
3. **Temel Yapay Zeka Bilgisi**: FSM ve kural tabanlı yapay zeka sistemlerini kullanarak karakter tepkileri geliştirebilmek için.
4. **Temel Ses ve Animasyon Bilgisi**: Yapay zekanın sesli ve görsel geri bildirimler vermesi için.

Bu bilgilere sahip olduğunuzda, oyununuzda bir yapay zeka sistemi kurmak ve karakter hareketlerine göre anında uyarılar verebilen bir AI sistemi oluşturmak mümkündür.

Kural tabanlı yapay zeka ve **Finite State Machines (FSM)** gibi temel yapay zeka sistemlerini kullanarak karakter tepkilerini geliştirmek için ihtiyacınız olan bilgi ve teknolojiler şu şekildedir:

### **1. Finite State Machine (FSM)**

FSM, yapay zekanın durumlar arasında geçiş yaparak kararlar verdiği basit bir yapıdır. Bir karakterin bir durumda belirli kurallara göre tepki vermesini ve başka bir duruma geçiş yapmasını sağlar. Örneğin, bir karakter tehlike algıladığında "alarm durumu"na geçebilir, bu durumda farklı tepkiler gösterebilir.

#### **FSM Öğrenmek İçin:**

* **Teorik Bilgi**: FSM'in mantığını anlamak için, bilgisayar bilimi temellerinde durum makineleri (state machines) hakkında bilgi sahibi olmalısınız. Bir FSM'nin nasıl çalıştığını ve hangi koşullarda durumlardan geçtiğini öğrenmek, yapay zeka tasarımının temel taşlarından biridir.
* **C# ile FSM Uygulaması**: FSM'yi Unity'de C# kullanarak uygulayabilirsiniz. Örneğin, karakterinizin farklı durumlarda nasıl hareket edeceğini FSM ile tanımlayabilirsiniz.

#### **Kaynaklar:**

* **Game Programming Patterns** kitabındaki FSM bölümü.
* Unity’de FSM uygulamak için video dersler ve tutoriallar (YouTube ve Udemy gibi platformlarda FSM temelli yapay zeka dersleri).

### **2. Kural Tabanlı Yapay Zeka**

Kural tabanlı yapay zeka sistemleri, belirli bir girişe (örneğin, karakterin koşması) karşılık belirli bir çıkış (örneğin, yapay zekanın “Koşma tehlikeli!” uyarısı vermesi) üreten basit bir yapıdır. Bu tür bir yapay zekayı kurmak için önceden belirlenmiş kurallara göre hareket eden bir sistem oluşturursunuz.

#### **Kural Tabanlı Yapay Zeka Öğrenmek İçin:**

* **Mantıksal Operatörler**: IF-THEN-ELSE gibi temel mantık yapılarıyla çalışabilmek için programlama dilinde bu tür kontrol yapılarını öğrenin. C# kullanıyorsanız, mantıksal kontroller yaparak kural tabanlı AI geliştirebilirsiniz.
* **Durum ve Koşul Yönetimi**: Yapay zekanın koşullara nasıl tepki vereceğini belirleyen algoritmalar geliştirmek. Örneğin, oyuncu yüksekten düştüğünde uyarı vermek gibi basit kural tabanlı sistemler.

#### **Kaynaklar:**

* **Artificial Intelligence for Games** (Ian Millington): Kural tabanlı yapay zeka ve FSM gibi temel AI sistemlerini anlamanızı sağlayacak bir kitap.
* Unity ve C# için temel AI yapıları ve kural tabanlı sistemlerle ilgili video dersler.

### **3. Sesli ve Görsel Geri Bildirimler İçin Ses ve Animasyon Bilgisi**

Yapay zekanın tepkilerini sesli ya da animasyonlu geri bildirimler şeklinde sunabilmek için Unity’nin ses ve animasyon sistemlerini kullanmalısınız.

#### **Sesli Geri Bildirim İçin:**

* **Unity Audio System**: Unity’de karakterin sesli geri bildirimler vermesini sağlayacak ses sistemini kullanmayı öğrenmelisiniz. Ses dosyalarını (örneğin, yapay zekanın konuşma sesleri) ekleyip, doğru zamanlarda çalınmasını sağlamalısınız.
* **Triggering Sounds**: Yapay zekanın belirli durumlarda ses çalmasını sağlamak için ses tetikleyicilerini (audio triggers) ayarlamayı öğrenin.

#### **Görsel Geri Bildirim İçin:**

* **Unity Animator System**: Yapay zekanın ekran üzerinde belirip konuşmasını ya da bir hareket yapmasını sağlamak için Unity’nin animasyon sistemini kullanın. Bu sistemle yapay zekanın bir uyarı verirken ekranda görünmesini sağlayabilirsiniz.
* **3D ve 2D Animasyonlar**: Karakter modeline animasyonlar eklemek ve animasyonları tetiklemek için Unity Animator’ı kullanmayı öğrenin.

#### **Kaynaklar:**

* Unity Documentation: **Audio** ve **Animation** bölümleri.
* **Brackeys** veya **Code Monkey** YouTube kanallarındaki Unity ses ve animasyon dersleri.

### **4. AI Behavior Trees (Opsiyonel)**

FSM'nin daha gelişmiş bir versiyonu olan **Behavior Trees**, daha karmaşık yapılar kurmak için kullanılabilir. Behavior Trees, FSM'ye göre daha esnek ve geniş çapta kullanılabilir. Daha sofistike yapılar kurmak istiyorsanız Behavior Trees’i de öğrenmeyi düşünebilirsiniz.

#### **Behavior Trees Öğrenmek İçin:**

* **Temel Kavramlar**: Behavior Tree mantığı, bir yapay zekanın görevleri belirli bir sırayla yürüttüğü ve sonuçlara göre dallandığı bir sistemdir. FSM'ye göre daha fazla kontrol sağlar.
* **Unity'de Uygulama**: Unity’nin Asset Store’unda Behavior Tree oluşturmak için çeşitli eklentiler bulunur. Örneğin, "NodeCanvas" veya "Behavior Designer" gibi eklentilerle yapay zeka davranışlarını tanımlayabilirsiniz.

#### **Kaynaklar:**

* Unity Asset Store: Behavior Tree eklentileri.
* **Behavior Trees for AI** dersleri (YouTube ve oyun programlama kurslarında).

### **Özetle:**

1. **FSM ve kural tabanlı yapılar**: Basit tepkiler vermesi için yeterlidir. FSM'yi C# ile uygulayıp, yapay zekanın durumlar arasında geçiş yapmasını sağlayabilirsiniz.
2. **Ses ve Animasyon Bilgisi**: Unity’nin ses ve animasyon sistemlerini öğrenerek yapay zekanın sesli ve görsel geri bildirimler vermesini sağlayabilirsiniz.
3. **Behavior Trees (Opsiyonel)**: Daha karmaşık yapılar geliştirmek istiyorsanız Behavior Trees'i öğrenmeyi düşünebilirsiniz.

Bu konularda kendinizi geliştirerek, sesli ve görsel geri bildirimler veren, kural tabanlı bir yapay zeka sistemi oluşturabilirsiniz.