Block Game - Geliştirme Görev Listesi

Bu liste, oyunu oynanabilir en basit versiyonundan (MVP) başlayarak adım adım geliştirmek için tasarlandı. Her "Part", oyunun büyük bir parçasını temsil ediyor.

AŞAMA 1: TEMEL MEKANİKLER VE OYNANABİLİR PROTOTİP (MVP)

Part 1: Proje Kurulumu ve Oyun Alanı

Amaç: Boş bir Unity projesinden, ekranda görünen ve etkileşime hazır bir oyun tahtası oluşturmak.

- **TASK-01: Proje Başlatma ve Yapılandırma**
    - **Açıklama: Yeni bir Unity 2D projesi oluşturulacak ve hedef platform olarak mobil (Android/iOS) seçilecek.**
    - **Uygulama: File > Build Settings üzerinden platform değişikliği yapılacak. Assets altında _Scripts, _Prefabs, _Sprites, _Scenes gibi standart klasör yapısı oluşturulacak.**
- **TASK-02: Prosedürel Grid Sistemi**
    - **Açıklama: Oyunun 8x8'lik tahtası, kod kullanılarak (prosedürel olarak) oluşturulacak. Bu, gelecekteki değişiklikler için esneklik sağlayacaktır.**
    - **Uygulama: GridManager.cs script'i, oyun başladığında bir GridCell prefab'ını çoğaltarak 8x8'lik bir görsel grid oluşturacak.**
- **TASK-03: Mantıksal Grid Altyapısı**
    - **Açıklama: Görsel grid'in yanı sıra, her hücrenin dolu veya boş olma durumunu takip eden bir veri yapısı kurulacak.**
    - **Uygulama: GridManager.cs içine Transform[,] logicGrid adında iki boyutlu bir dizi eklenecek. Bu dizi, blok yerleştirme ve sıra tamamlama kontrolleri için temel oluşturacak.**

Part 2: Blok Mekanikleri

Amaç: Oyuncuya rastgele bloklar sunmak, bu blokları sürüklemesini ve oyun kurallarına göre yerleştirmesini sağlamak.

- **TASK-04: Esnek Blok Veri Yapısı (ScriptableObject)**
    - **Açıklama: Oyundaki tüm blok şekillerini ve rotasyonlarını, kod değiştirmeden yönetebilmek için bir veri yapısı oluşturulacak.**

    - **Uygulama: BlockData adında bir ScriptableObject yaratılacak. Bu obje, bir bloğun hücre pozisyonlarını (List<Vector2Int>) ve görsel prefab'ını tutacak. Farklı şekiller (T, L, I, kare vb.) ve rotasyonları için en az 8-10 adet BlockData asset'i oluşturulacak.**

- **TASK-05: Blok Oluşturma (Spawning)**
    - **Açıklama: Oyuncuya her turda rastgele 3 blok sunan bir sistem geliştirilecek.**
    - **Uygulama: BlockSpawner.cs script'i, BlockData asset'lerinden rastgele üçünü seçerek, ekranın kenarındaki bekleme alanlarında görsel olarak canlandıracak.**
- **TASK-06: Sürükle-Bırak Mekaniği**
    - **Açıklama: Oyuncunun blokları fare veya parmak ile tutup oyun alanına sürükleyebilmesi sağlanacak.**
    - **Uygulama: BlockDragger.cs script'i, OnMouseDown, OnMouseDrag, O nMouseUp olaylarını kullanarak bu işlevselliği yönetecek. Sürüklemenin çalışması için Camera objesine Physics 2D Raycaster ve blok hücrelerine Collider 2D bileşenleri eklenecek.**
- **TASK-07: Blok Yerleştirme Mantığı**
    - **Açıklama: Bırakılan bir bloğun, grid sınırları içinde ve boş hücrelere denk gelip gelmediği kontrol edilecek.**
    - **Uygulama: GridManager'a CanPlaceBlock fonksiyonu eklenecek. BlockDragger, blok bırakıldığında bu fonksiyonu çağıracak. Geçerli ise blok yerleşecek, geçersiz ise başlangıç pozisyonuna dönecek.**

Part 3: Oyun Döngüsü ve Skor

***Amaç: Oyunun ana döngüsünü (sıra tamamlama, puan kazanma, yeni blokların gelmesi, oyun sonu) tamamlamak*.**

- **TASK-08: Sıra/Sütun Tamamlama ve Temizleme**
    - **Açıklama: Bir satır veya sütun tamamen dolduğunda, o hattın temizlenmesi ve oyun alanında yer açılması sağlanacak.**
    - **Uygulama: GridManager'daki CheckForCompletedLines fonksiyonu, dolu hatları tespit edecek. ClearLines fonksiyonu ise bu hatlardaki**

hücreleri hem görsel (Destroy) hem de mantıksal (logicGrid'i güncelleme) olarak temizleyecek.

- **TASK-09: Puanlama ve Combo Sistemi**
    - **Açıklama: Oyuncuyu ödüllendirmek için bir skor sistemi kurulacak.**
    - **Uygulama: ScoreManager.cs, temizlenen her hat için standart bir puan ve aynı anda birden fazla hat temizlendiğinde ekstra "combo" bonusu verecek. Skor, ekrandaki bir TextMeshPro UI elemanı ile gösterilecek.**
- **TASK-10: Oyun Döngüsünün Devamlılığı**
    - **Açıklama: Oyuncu elindeki 3 bloğu yerleştirdiğinde, yeni bir setin otomatik olarak gelmesi sağlanacak.**
    - **Uygulama: BlockSpawner, yerleştirilen blokları bir listede takip edecek. Liste boşaldığında, yeni bir 3'lü blok setini oluşturan SpawnNewBlockSet fonksiyonunu tetikleyecek.**
- **TASK-11: Oyun Sonu (Game Over) Koşulu**
    - **Açıklama: Oyuncunun elindeki bloklardan hiçbirini tahtaya yerleştirecek geçerli bir hamlesi kalmadığında oyun sonlanacak.**
    - **Uygulama: GridManager'daki IsAnyMovePossible fonksiyonu, bir bloğun tahtaya sığıp sığamayacağını kontrol edecek. BlockSpawner, elde kalan bloklar için bu kontrolü yaparak hamle kalmadığında "Oyun Bitti" durumunu tetikleyecek.**
- **TASK-12: Arayüz ve Sahne Yönetimi**
    - **Açıklama: Oyun için bir ana menü ve oyun sonu ekranı oluşturulacak.**
    - **Uygulama: MainMenuScene ve GameScene adında iki sahne oluşturulacak. "Oyna" ve "Tekrar Oyna" butonları, SceneManager.LoadScene komutu ile sahneler arası geçişi sağlayacak. Oyun bittiğinde GameOverPanel aktif edilecek.**
- **TASK-13: Veri Kalıcılığı (En Yüksek Skor)**
    - **Açıklama: Oyuncunun en yüksek skoru, oyun kapatılıp açıldığında dahi saklanacak.**
    - **Uygulama: PlayerPrefs kullanılarak, oyun bittiğinde currentScore ile karşılaştırma yapılacak ve yeni rekor cihaz hafızasına kaydedilecek. Bu skor, hem oyun içinde hem de ana menüde gösterilecek.**

AŞAMA 2: CİLALAMA (POLISHING) VE EK ÖZELLİKLER

Part 4: Görsel ve İşitsel Geri Bildirim

Amaç: Oyuncu aksiyonlarını daha tatmin edici ve canlı hissettirmek.

- **TASK-14: Görsel Efektler ve Animasyonlar (Juiciness)**
    - **Açıklama: Oyuna küçük animasyonlar eklenerek daha dinamik bir his verilecek.**
    - **Uygulama:**
        - **Blok yerleştiğinde hafifçe büyüyp küçülme (scale) animasyonu eklenecek.**
        - **Sıra temizlenirken, hücrelerin kaybolmadan önce parlaması veya küçülerek yok olması gibi bir efekt eklenecek.**
        - **Skor arttığında, skor metninin anlık olarak vurgulanması sağlanacak.**
- **TASK-15: Ses Yönetimi**
    - **Açıklama: Oyuna ses efektleri ve arka plan müziği entegre edilecek.**
    - **Uygulama:**
        - **Bir AudioManager.cs script'i oluşturularak tüm sesler merkezi bir yerden yönetilecek.**
        - **Blok yerleştirme, sıra temizleme, geçersiz hamle ve buton tıklama gibi temel aksiyonlar için ses efektleri eklenecek.**
        - **Oyuna uygun, rahatlatıcı bir arka plan müziği eklenecek.**

Part 5: Kullanıcı Deneyimi (UX) ve Ekstra Özellikler

Amaç: Oyunu daha kullanıcı dostu hale getirmek ve tekrar oynanabilirliği artırmak.

- **TASK-16: "Hayalet Blok" Göstergesi**
    - **Açıklama: Oyuncu bir bloğu sürüklerken, eğer bırakırsa tam olarak nereye yerleşeceğini gösteren yarı saydam bir "gölge" oluşturulacak.**
    - **Uygulama: OnMouseDrag sırasında, bloğun bırakılacağı en yakın grid pozisyonu hesaplanacak ve o pozisyona bloğun yarı saydam bir kopyası anlık olarak çizilecek.**
- **TASK-17: Android Geri Tuşu Desteği**

    - **Açıklama: Cihazın "Geri" tuşuna basıldığında uygulamanın aniden kapanması engellenecek.**
    - **Uygulama: Update() içinde Input.GetKeyDown(KeyCode.Escape) kontrolü yapılacak. Oyun içindeyken bir "Çıkmak istiyor musun?" paneli gösterilecek.**
- **TASK-18: Ayarlar Menüsü**
    - **Açıklama: Oyuncunun ses ve müzik seviyelerini kontrol edebileceği basit bir menü eklenecek.**
    - **Uygulama: AudioManager'daki ses seviyeleri AudioListener.volume veya AudioSource.mute ile kontrol edilecek. Bu ayarlar, kalıcı olması için PlayerPrefs ile kaydedilecek.**