
Hazırlık Ekranı

Oyun hazırlık ekranı ve oyun UI ekranı üstüste duruyor ve Start'a basıldığında GamePrepScreen inaktif edilerek altta duran oyun ekranı gösteriliyor.

GamePrepScreen ekranında genişliği ve yüksekliği Slider ile belirliyoruz ve sliderın değişmesi ile gerçekleşen OnValueChange kısmını UseInterfaceManager scriptinde yazdığımız WidthSliderChange() methoduyla bağlıyoruz.

WidthSliderChange() Methodunda ise sadece tek sayılara izin vererek simetrik görünümü koruyoruz.

Yükseklik seçiminde ise yine slider kullanıyoruz ve burada tek sayıları engellemiyoruz çünkü yükseklik iç içe for döngüsünün içeride olan for'u.

Örneğin genişliği 7 yüksekliği 12 seçersek 7 kere 12'li hexagon grubu oluşturacak ve bu durumda simetri bozulmuyor.

Renk sayısı seçimi için yine Slider kullanıyoruz. OnValueChange değerini ise UserInterfaceManager scriptindeki ColorCountSliderChange() methoduyla ilişkilendiriyoruz.

Bu methodun içerisinde 'childCount' değerini ve 'newCount' değerini tutuyoruz. 'childCount' değerine göre ColorSelection nesnesinin altında bulunan Selection nesnesine daha önce eklediğimiz 7 adet hexagon nesnesinin active olma durumuna karar veriyoruz.

Oynanmak istenilen değerler seçildikten sonra Start butonuna tıklanmasıyla beraber OnClickEvent'inde StartGameButton() methodu çalışıyor.

StartGameButton methodu GamePrepscreen nesnesini inaktif yapıyor ve yükseklik genişlik gibi seçtiğimiz değerleri gerekli methodlara gönderiyor.

Default butonu ise SuperClassta tanımladığımız sabit varsayılan değerleri Slidera atayan default methoduna bağlı.

Back butonu ise sahne yükleyen bir methoda bağlı. Biz Back butonunu 'Main Menu'isimli sahneye yönlendireceğiz.

StartGameButton() sonrası

Oyun hazırlık ekranı inaktif edilir. GridManager scriptindeki gridWidth, gridHeight ve colorList doldurulur InitializeGrid() methodu çalıştırılır. Böylece altta bulunan oyun ekranı görünür hale gelir. InitializeGrid() methodunda 'missingCell' listesinin count'u oluşturalacak hexagon sayısıdır. MissingCells listesindeki değerler ise hangi elemanın hangi kolonda bulundugunu bilmemizi sağlar.

Seçilen genişliğe göre oluşacak sütun sayısı ise 'gameGrid' countudur. Her sütun için gameGrid'e boş bir List<hexagon> eklenir. Daha sonra grid'in doldurulması için ProduceHexagons coroutine'i başlatılır. Bu coroutine'e 'missingCells' listesi ve ColoredGridProducer() methodunun return degeri gönderilir. Seçilen genişliğe göre ilk hexagonun başlangıç noktası değişeceği için GetGridStartCoordinateX() fonksiyonunda transform.position.x değeri bulunur. Başlangıç noktası belirlendikten sonra hexagon üretimine başlanır.

ProduceHexagons'a gönderdiğimiz 'missingCell' listesindeki her eleman için 'Stepper' üzerinde olup olmadığı kontrol edilir transform.position.y değerleri belirlenir. Bulunan x,y pozisyonları StartPosition'a atılır. Daha sonra SuperClassta belirlenen HEX_START_POSITION noktasında 'hexPrefab' instantiate edilir. ColoredGridProducer() methodunun return degeri olan 'returnValue' yanyana gelince patlamayacak şekilde ayarlanmış renk bilgisini tutar. Instatiate edildikten sonra rengini alır ve pozisyon değişikliği işlemine geçilir. Oluşturulan her hexagon newObj'ye atılır. newObj nesnesindeki her hexagonun sahip oldugu 'Hexagon' scripti GetComponent<> ile alınır ve 'newHex' içerisinde tutulur. 'newHex' ile Hexagon'a bağlı scripte ulaşıp ChangeWorldPosition()'a buldugumuz 'startPosition' gönderilir. Bu method ile önceden oluşturduğumuz hexagonları instantiate ettiğimiz noktadan 'startPosition' noktasına gitmesi sağlarız. StartPosition değeri LerpPosition'a atanır ve 'lerp' değişkenine true değeri verilir. Hexagon scriptinde bulunan Update() her frame'de çağırıldıkça ve 'lerp' değişkeni true değer aldığında Hexagon yer değişikliğinin zaman içerisinde animasyonlu bir şekilde gerçekleşmesi sağlanır. Daha sonra yer değişikliğinin tamamlanması kontrol edilip 'lerp' = false yapılır. ChangeGridPosition() methodunda ise her hexagonun x ve y değişkenleri doldurulur. Bu x,y değerleri hangi sütun ve satırda bulundugu belirtir.(Hayali ızgara üzerindeki pozisyonlarını belirtir.)

Buraya Kadar oyun ekranı oluşturuldu artık hexagonlar sahnemizde olması istenilen yere getirildi. Şimdi oynanış mekaniklerini açıklayacağım.

Öncelikle dokunma mekaniklerinin doğru çalışması için yapılan dokunma hareketinin bazı kuralları karşılaması gerekiyor. Bu kurallar şu şekilde; Hexagon üretilmiş olup gridde ve world'de yerleşmiş olmalı, oyun bitmemiş olmalı, hexagon dönmüyor olmalı ve patlama tamamlanmış olmalı. Bu kurallara uyan hexagonlara yapılan dokunuşlar bizim için uygundur. InputAvailabile() bu kurallara uyulmasını kontrol eder ve karşılığında bool değer gönderir. InputManager() scriptinde her frame'de dokunma sayısı ve dokunmanın uygun bir dokunuş olup olmadıgı kontrol edilir. Eğer var ise dokunduğumuz objenin collider'ı alınır.

CheckSelection() methodu parmak ile dokunarak alınan collider'ın null olup olmadığını ve nesnesinin etiketini kontrol eder. Daha sonra parmağın ekrandan kaldırılmasını kontrol eder ve collider'ı Select()'e gönderir. Select() methodu ise dokunduğumuz hexagonun önceden seçtiğimiz hexagon olup olmadıgını kontrol edip selectedPosition.x ve selectedPosition.y değerlerini doldurur. Eğer aynı hexagona birden fazla kez dokunulursa 'SelectionStatus' değeri 1 arttırılır.

'SelectionStatus' integer bir değerdir ve biz bu değeri hangi komşuların seçileceğini belirlemek için kullanacağız. Hexagona yapılan ilk dokunuşta 'SelectionStatus' = 0 ' a eşitlenir ve ConstructOutline() methodu çağırılır. Hangi grubun seçildiğini kullanıcıya belirtmek için Outline oluşturacağız. GetNeighbours() methodu sayesinde her hexagon kendi komşularını bilir. SelectionStatus'un değerine göre seçilen hexagonun komşuların grid koordinatları bir switch-case yapısıyla belirlenir. Daha sonra bu koordinatlar ile selectedGroup listesine hexagonlar eklenir. Artık Outline oluşturabiliriz. 'selectedGroup' listesindeki her hexagon için 3 adet gameObject oluşturulur. bunlardan biri hexagonun kendisini tutarken diğer ikisi outline'nın dışını oluşturacak ve içini dolduracak nesnelerdir. Outline nesnesini hexagon'a göre biraz daha büyük yapıp hexagonun olduğu yerde oluşturuyoruz. Büyük yapmamızın sebebi seçili hexagon grubunun çevresini beyaz bir çizgi ile işaretlemiş görüntüsü yaratmak. Daha sonra 'selectedGroup' listesindeki her hexagon için bir outline inner objesi oluşturuyoruz. Outline Inner objeleri ile outline'ın üzerinde listedeki hexagonun aynısını collider'sız bir şekilde yaratarak kullanıcıya o grubu seçtiğini belirten görüntüyü sağlıyoruz. Şimdi bu seçtiğimiz grubu saat yönüne veya tersi yönde döndürelim. Bunun için InputManager scriptine gitmemiz gerekiyor.

Döndürme Mekanikleri ve sonrası

Saat yönüne veya tersi yönde dönüşü tetiklemek için dokunma hareketinin başlangıç ve bitiş pozisyonunu (son pozisyon yada şuanki pozisyon da diyebiliriz.) değerlendirmemiz gerekiyor. TouchDetection() ile parmağın ekrana dokunmaya başlaması tespit edilir ve 'touchStartPosition' doldurulur. Parmağın ekrandan kalkmadan önceki hareketini izleyip 'touchCurrentPosition' değerini dolduruyoruz.

Parmağın son pozisyonu ile ilk pozisyonun 'x' ve 'y' koordinaları arasındaki mesafe incelenir. Mesafenin hangi eksende daha büyük olduğuna göre dönüş hareketinin x ekseninde mi yoksa y ekseninde mi tetiklendiğine karar veririlir. Eğer 'x' ekseninde tetiklendiyse x eksenindeki mesafe farkının pozitif mi negatif mi olduğuna göre kaydırma hareketinin sağa mı sola mı yapıldığı belirlenir. eğer 'y' ekseninde tetiklenseydi aşağı doğrumu yoksa yukarı doğrumu kaydırma yapıldığı da distanceY'nin pozitif veya negatif olmasına göre belirlenecektir. Eğer tetikleme 'y' ekseninde olduysa hexagonun sağından mı yukarı yada aşağı kaydırıldı yoksa solundan mı şeklinde kontrol edilip 'clocWise' değerine 1 yada 0 atanır. Daha sonra bu 'clockWise' değeri if ile kontrol edilir. Kontrole göre hexagonlar arası yer değiştirme işlemini ya her hexagonu bir sonraki hexagonun pozisyonuna gönderir yada bir önceki hexagonun pozisyonuna gönderir.

Artık sıra yapılan dönüş sırasında hexagonların nasıl patlayacağına geldi. 'clockWise' değeri GridManager scriptindeki Rotate()'e gönderilir. Rotate() methodunda bir coroutine başlatılır. Bu coroutine'de seçilen hexagon grubunun dönüyor olma durumunu belirler. SwapHexagons() methodu çalışır ve seçilen grubunun elemanlarının dünyadaki ve griddeki pozisyonları belirlenir. Belirlenen bu pozisyonlar seçilen hexagonlara bağlı olan hexagon scriptindeki Rotate() methoduna gönderilir ve bu hexagonlar dönüş saat yönünde ise bir sonraki hexagonun yerine geçer saat yönü tersi ise bir öncekinin yerine geçer. Bu yer değiştirme işlemi Hexagon scriptindeki Update() her çağırıldığında(yani her frame'de) Math.Lerp yardımı ile animasyonlu bir şekilde gerçekleşir. Yer değiştirme tamamlandıgında 'lerp' false değer alır ve koşuldan çıkılır. Bu dönüş sırasında her hangi bir patlama yaşanır ise RotationCheckCoroutine()'deki for döngüsünden 'break' komutu ile çıkılır. Daha sonra patlayan hexagonların yerine yeni hexagonlar yerleştirmek için ProduceHexagons()'a ExplodeHexagons() methodunun return ettiği 'missingColumns' gönderilir. 'missingColumns' listesi hangi kolonda hexagon patladığını bilmemizi sağlar. Bu bilgi ile patlama olan kolonlardaki bütün elemanları gezip transform.position değerini(yani dünyadaki pozisyonu) ve griddeki pozisyonunu güncelliyoruz. 'missingColumns' listesindeki kolon numaralarına göre yeni hexagonalar instatiate edilir ve olması gereken posizyona gönderilir. Eğer bir patlama gerçekleşirse Score kazanılır. Patlayacak

hexagonar listesindeki her hexagon Remove() edilmeden önce Scrore()'a 1 değerini göderir. Score() gelen 1 değerini alır ve 'blownHexagons' sayacına ekler. Bu sayaç 5 ile çarpılıp Score'un gösterildiği text'e yazılır. Her 1000 score'da bomba oluşacağı için 'bombCount' sayacını çarpan olarak kullanıyoruz. Oluşturduğumuz Outline'ı yok etmek için DestructOutline() methodunu kullanıyoruz. Sahnedeki Outlines objesinin altında child objeler var ise bunları yok ediyoruz.

Serhat Karabağ