



DeepL

Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

ASTEROID DASH

PROJE MÜHENDİSLİĞİ
PROGRAMLAMA
ÖDEV



Hacettepe Üniversitesi - Bilgisayar Mühendisliği

BBM203 Yazılım Pratiği I - Güz 2024



ASTEROID ÇİZGİ

Konular: Bağılı Listeler, Dinamik Bellek Tahsisi, Matrisler, Dosya I/O

Kurs Eğitmenleri: Doç Dr. Adnan ÖZSOY, Yrd Doç Dr. Engin DEMİR, Doç Dr. Hacer YALIM KELEŞ

Doç: M. Aslı TAŞGETİREN, S. Meryem TAŞYÜREK, **Dr. Öğr. Üyesi Selma DİLEK***, Alperen ÇAKIN*

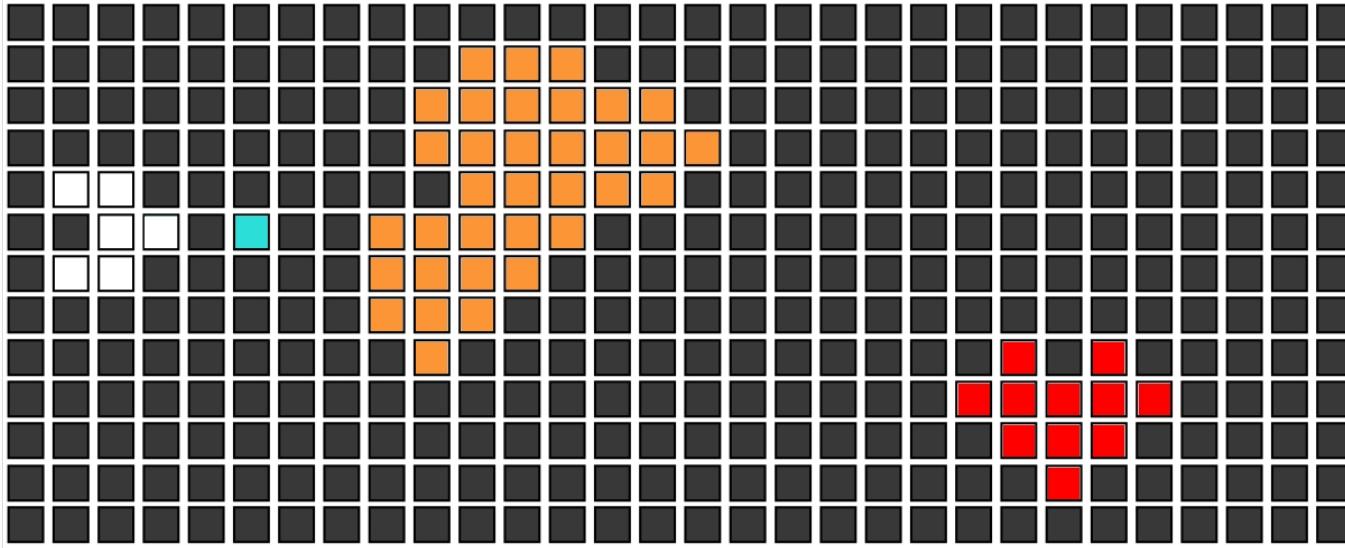
Programlama Dili: C++11 - **Bu başlangıç kodunu KULLANMALISINIZ**

Son Tarih: **Cuma, 15/11/2024 (23:59:59)**

Asteroid Dash

Kitleler İçin Bir Sonraki Viral Asteroid Parçalayan Skor Vurma Oyunu

Kodlama becerilerinizin hızlı tempolu bir uzay macerasında test edileceği **Asteroid Dash**'e hoş geldiniz! Bu görevde, oyuncuların tehlikeli bir asteroit kuşağında gezindiği arcade tarzı bir oyun geliştireceksiniz. Kaçınılması gereken asteroidler, toplanması gereken güçlendiriciler ve stratejik atış mekanizmaları ile bu görev, C++'da dinamik diziler, bellek yönetimi ve nesne yönelimli tasarım anlayışınızı zorlayacak. **HUBBM** ve **HUAIN**'ın seçkin programcıları olarak, oyuncuların uzay araçlarını gelen asteroit dalgaları arasında yönlendirmeleri gereken yüksek oktanlı bir oyun olan **Asteroid Dash'in** arkasındaki motoru oluşturmakla görevlendirildiniz. Uzay nesnelerini, çarpışma algılamayı ve zamana dayalı oynanışı başarıyla yönetmek için ihtiyaç duyacağınız nesne odaklı programlama ve dinamik bellek yönetiminin güçlü araçları emrinizde.



Asteroid Dash, asteroitlerin sağdan belirdiği ve oyuncunun uzay aracına doğru sola doğru hareket ettiği ızgara tabanlı bir uzay ortamında gerçekleşir. Uzay aracı kendini savunmak için ateş edebilir, ancak sınırlı cephe ile oyuncular gelen asteroitlerden kaçarken kaynaklarını taktiksel olarak yönetmelidir. Cephaneyi dolduran veya canları geri getiren güçlendirmeler oyuna dinamik bir unsur katıyor. Oyundaki her adım ızgarayı günceller, asteroitleri hareket ettirir, hasar gördükçe döndürür ve oyuncunun eylemlerini yansıtır. Bu ödev, bağıntılı listeleri ve dosya I/O, sınıf tasarımı ve bellek yönetimi gibi temel C++ kavramlarını vurgulayarak öğrencilere gerçek zamanlı, olay güdümlü bir oyun geliştirme konusunda pratik deneyim sağlar. **Asteroit kuşağında hayatı kalabilecek ve mücadelenin üstesinden gelebilecek misiniz?**



ASTEROİD ÇİZGİ



1 Giriş Verilerinin Okunması ve Oyunun Başlatılması

Bu bölümde, sağlanan girdiler aracılığıyla oyunun başlatılmasını özetliyoruz. Tam kredi için çok önemli olduğundan dinamik bellek ayırma gereksinimlerine çok dikkat edin.

1.1 Girdi Dosyaları ve Komut Satırı Bağımsız Değişkenleri

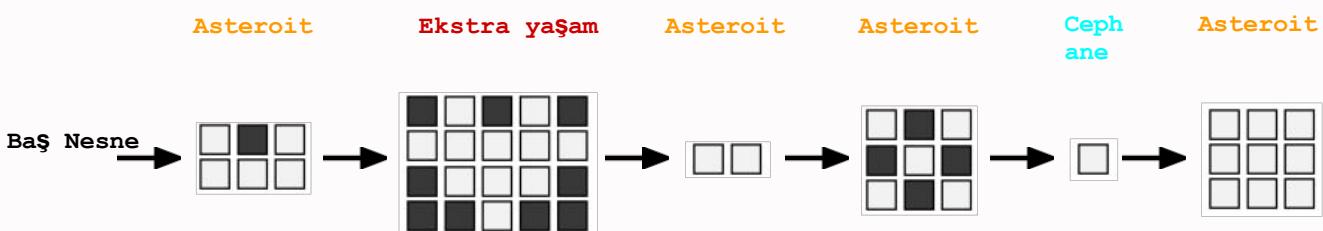
ilk komut satırı argümanı aracılığıyla DAT formatında 2D satır \times renk matrisi olarak yapılandırılmış oyunun **uzay_gridi** hakkında ayrıntılar içeren bir girdi dosyası sağlanacaktır. Göreviniz, **AsteroidDash** sınıfı içinde **space_grid** özelliğini ayarlamak için programınız içinde bu dosyayı ayırtırmaktır. Space_grid.dat olarak verilen 10 satır ve 20 sütunlu örnek bir girdi dosyasının içeriği sağda gösterilmektedir. Dosya, her bir rakamın tek bir boşlukla ayrıldığı ve her satırın bir satırsonu karakteriyle bittiği rakam satırlarından oluşur.

DAT formatında olan ve *ikinci komut satırı argümanı* olarak verilen *ikinci* girdi dosyası, *CelestialObject* sınıfının örnekleri olarak temsil edilen oyunun **göksel nesnelerini** içerir; bunlar, asteroidler için köşeli parantez [] ve güçlendiriciler için küme parantezleri { } içinde verilen $2D \text{ yükseklik} \times \text{genişlik}$ şekil matrislerine sahiptir. Her gök cismi, seklinin altında özelliklerini belirten meta veri satırları da içerir:

- **s**: uzay izgarasında nesnenin sol üst köşesinin görüneceği **başlangıç satırını** belirtir. Nesneler her zaman izgaranın en sağ (son) sütunundan girecektir.
 - **t**: nesnenin oyuna girmeye başlaması gereken **tik** veya zaman **adımını** belirtir sağdan ızgara.
 - **e**: (sadece güçlendirmeler için) güçlendirmenin oyuncu üzerindeki etkisini tanımlar: ekstra bir can eklemek için can veya oyuncunun cephanelerini yenilemek için **cephane**.

Programınız bu dosyayı **AsteroidDash** sınıfında oyunun göksel nesnelerinin **bağlantılı** bir **listesini** oluşturmak için kullanmalıdır. Örnek bir dosya olan `celestial_objects.dat` dosyasından bir alıntı sağda gösterilmektedir. Gök cisimleri oyunda görünümleri gereken sıraya göre (uzay izgarasının sağ tarafından sola doğru hareket ederek) ilk döndürülmemiş halleriyle yerleştirilir. **Birden fazla nesnenin uzay izgarasına aynı anda girebileceğini unutmayın, ancak görevi basitleştirmek için, herhangi biri vurulduktan sonra dönce bile asla birbirleriyle çarpışacak şekilde konumlandırılmayacaklardır.**

Tek tek nesnelerin *yükseklik* ve genişliklerinin değişim的能力ini ve tüm gök nesneleri ve dönüşleri için dinamik bellek tahsisi gereklidir. Bu örnek girdideki gök cisimlerinin bir örneği aşağıda verilmiştir:



DAT formatında olan ve *üçüncü komut satırı argümanı* olarak verilen girdi dosyası, **Player** sınıfının bir örneği olarak temsil edilen **oyuncunun uzay aracı** hakkında



ASTEROID ÇİZGİ



- İlk satır, uzay aracının sol üst köşesinin görüneceği başlangıç satırını ve sütununu belirterek, oyuncunun uzay aracının uzay ızgarası üzerindeki ilk **konumunu** sağlar.
- Sonraki satırlar uzay aracının **2B şeklini** 1 ve 0'lardan oluşan bir matris olarak tanımlar; burada her 1 uzay aracının dolu (dolu) bir hücreni ve her 0 boş bir hücreyi temsil eder.

Bu dosya, bir **Oyuncu** örneği oluşturmak ve oyuncunun uzay aracını uzay ızgarasındaki ilk konumuna ayarlamak için programınız tarafından okunmalıdır.

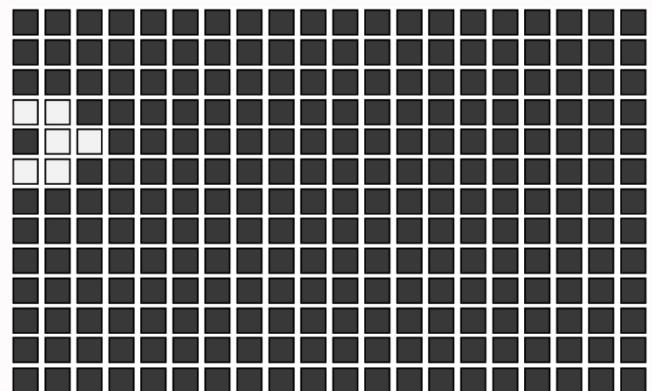
Oyunun **komutlarını** içeren ve dizeler olarak temsil edilen girdi dosyası, **dördüncü komut satırı argümanı** olarak DAT formatında sağlanacaktır. Programınız, **GameController** sınıfı içinde uygulanacak bir işlem olan oynanışı kolaylaştırmak için bu dosyanın içeriğini yorumlamakla görevlendirilmiştir. Commands.dat adlı tipik bir girdi dosyasından bir alıntı referans için sağda gösterilmektedir.

```
MOVE_UP
MOVE_DOWN
MOVE_RIGHT
MOVE_LEFT
SHOOT
NOP
```

Beşinci komut satırı bağımsız değişkeni, **Liderlik tablosu** verilerinin depolandığı bir metin dosyasının dosya adını belirler (daha fazla ayrıntı atama talimatlarının sonraki bölümlerde yer alacaktır). Son olarak, **altıncı komut satırı bağımsız değişkeni**, liderlik tablosu tanımlama amaçları için **mevcut oyuncunun adını** belirler.

1.2 Izgaranın Başlatılması

Oyunun **space_grid**'i dinamik olarak tahsis edilen 2D tamsayı matrisi olarak yönetilir; burada **sıfırlar** (0'lar) boş hücreleri (siyah kareler olarak gösterilir) **ve birler** (1'ler) dolu hücreleri (beyaz kareler olarak gösterilir) temsil eder. Oyuncunun uzay aracı, **sol üst köşesi belirtilen başlangıç satırına ve sütununa yerleştirilecek şekilde** şecline göre ızgara **üzerinde** konumlandırılmalıdır. Izgaranın boyutlarının oyuncunun uzay aracını sınırları aşmadan yerleştirmek için yeterli olduğunu varsayıbilirsiniz.

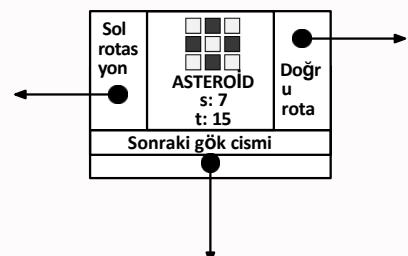


1.3 Göksel Nesne Listesinin Başlatılması

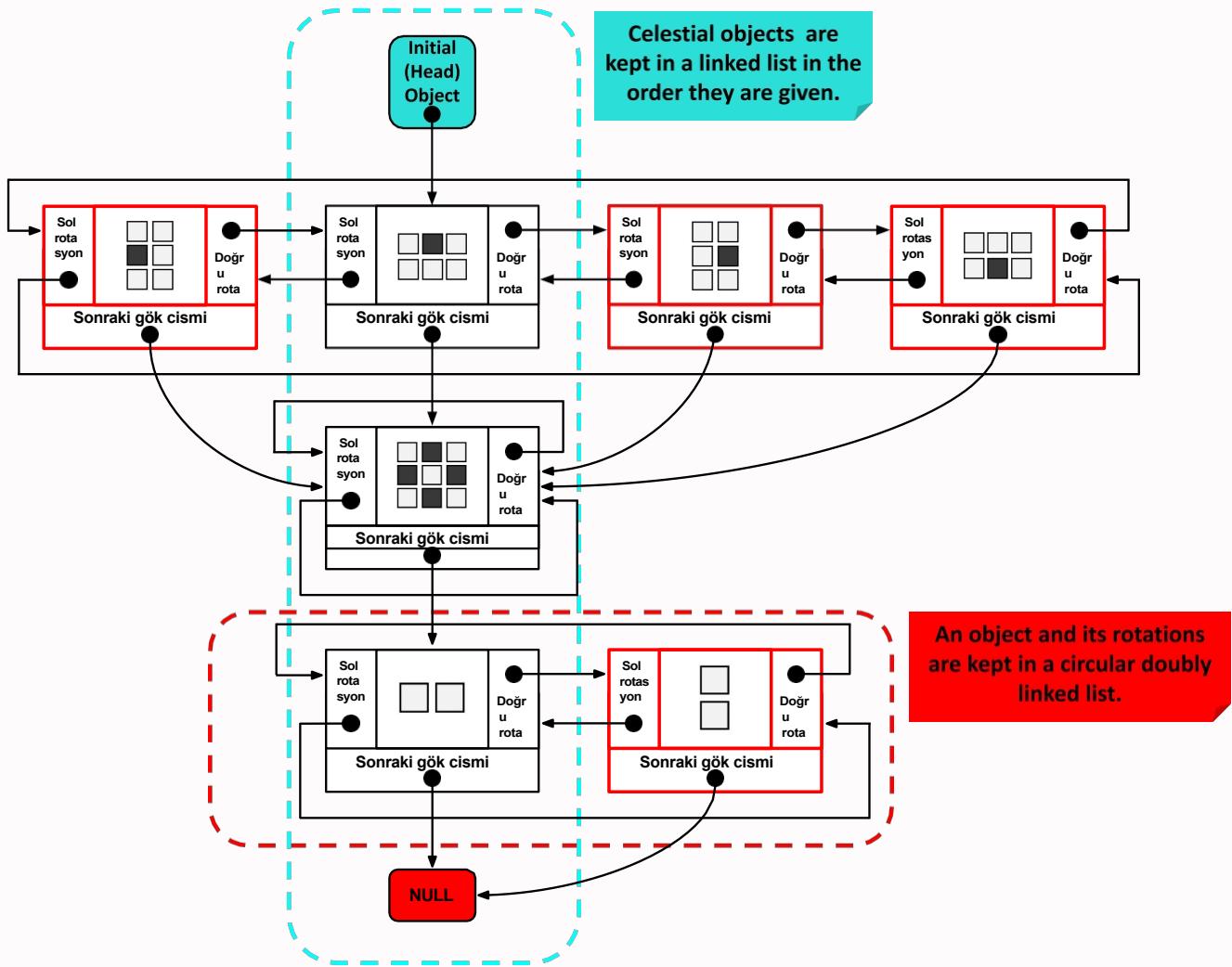
Oyunda, **göksel nesneler** **AsteroidDash** sınıfındaki **celestial_objects_list_head** işaretçisinden erişilebilen **çok seviyeli bir bağlantılı listede düğümler** (**CelestialObject** sınıfının **örnekleri**) olarak temsil edilecektir. İlgili girdi dosyasından sıralı gök nesnelerini okuduktan sonra, ilk adımınız her nesnenin şekeini, **nesne_türünü**, **başlangıç_sırasını** ve **görünme_zamanını** tanımlamak, onu stantiate etmek ve ardından oyunun bağlantılı gök nesneleri listesine entegre etmektir.

nesneleri. Bu entegrasyon, listenin bir önceki göksel nesnesinin **next_celestial_object** işaretçisini doğru bir şekilde atayarak elde edilir. **Ayrıca, her bir nesnenin olası tüm dönüşülerini hesaplamamanız ve bunları dairesel çift bağlantılı bir listede saklamamanız gereklidir.**

Her göksel nesnenin **sağ_dönüş** işaretçisi, dairesel bir şekilde geri dönerek saat yönündeki dönüşlerine sıralı erişime izin vermelidir. Benzer şekilde, **left_rotation** işaretçisi de saat yönünün tersine dönüşlerine dairesel bir düzende ardışık erişim sahiplamalıdır. **Hem orijinal göksel nesnenin**



ASTEROİD ÇİZGİ



Her gök cisminin hiç veya birden fazla dönüş durumuna sahip olabileceğini unutmayın (bir asteroidin bir mermiyle çarpışması nedeniyle döndüğü durumlar için gereklidir). Dönüşler, `right_rotation` ve `left_rotation` işaretçileri kullanılarak dairesel çift bağlantılı bir yapıda düzenlenir.

2 Uygulanacak Temel İşlevler

Bu bölümde, oyun kurallarını ve uygulama için gerekli işlevleri ele alıyoruz. Dinamik bellek tahsisine gerekliliklerine sıkı sıkıya bağlı kalmak tam kredi için çok önemlidir.

2.1 Komutları Okuma ve İşleme

`AsteroidDash`, ilgili girdi dosyasında her biri yeni bir satırda boşluksuz olarak listelenen yedi komuta sahiptir. Komutlar dinamik olarak ve karşılaşıldıkları anda yorumlanmalıdır (her komut bir oyun adımında yürütülür), gerçek zamanlı oyun koşullarını taklit etmek için bellekte saklanmamalıdır. Komutlar aşağıdaki gibidir:

- `PRINT_GRID`: Mevcut oyun adımında gerekli güncellemlerden sonra uzay izgarasının durumunu yazdırır.
- `MOVE_UP`: Oyuncu uzay aracını mümkünse bir boşluk yukarı hareket ettirir.



- **MOVE_DOWN**: Oyuncu uzay aracını mümkünse bir boşluk aşağı hareket etterir.
- **MOVE_RIGHT**: Oyuncu uzay aracını mümkünse bir boşluk sağa hareket etterir.
- **MOVE_LEFT**: Oyuncu uzay aracını mümkünse bir boşluk sola hareket etterir.
- **ATIŞ**: Bir mermi ateşleyin.
- **NOP**: Bir sonraki tıkta işlem yok: hiçbir şey yapmayın.

Bilinmeyen komutlar bir hata mesajı yazdırılarak işlenir, ancak işleme bir sonraki komutla devam edilir. Sağdaki örneğe bakın.

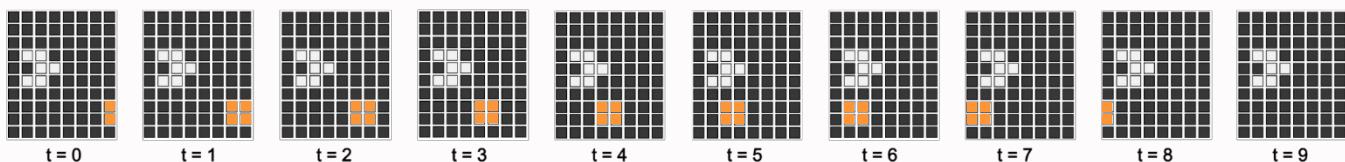
Bilinmeyen komut: GIMME_POINTS

2.2 Oyun Kuralları

Bu oyun klasik *Shoot 'em up* oyunlarından ilham alsa da, kural setinde farklılık göstermektedir. Bu nedenle, oyun kurallarını titizlikle gözden geçirmek ve belirtilen her bir temel gereksinime ve ayrıntıya çok dikkat ederek atama görevlerini yerine getirmek çok önemlidir.

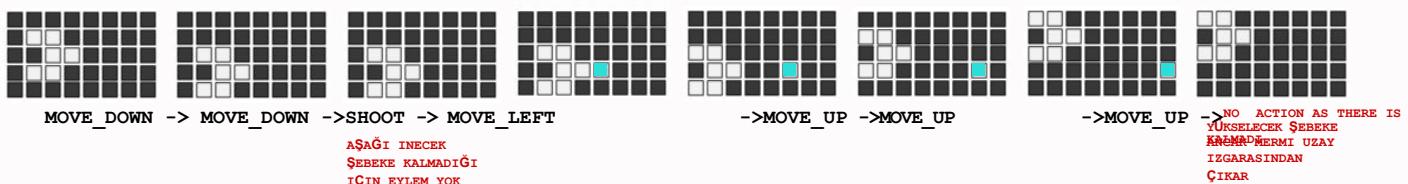
2.2.1 Oyuncu ve Göksel Nesnelerin Hareketleri ve Çarpışma Algılama

Oyuncunun **uzay** aracı ve göksel nesnelerin hareket ederken **space_grid** ile etkileşime girer ve oyuncu eylemlerine veya oyun olaylarına yanıt verir. Oyuncunun uzay aracı, oyuncu veri dosyasında belirtildiği gibi izgaraya belirli bir başlangıç konumundan girer. Her zaman adımda, gök cisimleri izgaradan çıkışa ya da oyuncuya çarpışana kadar bir hücre sola doğru ilerler. Aynı anda, oyuncu uzay aracının hareketlerini kontrol edebilir ve gelen nesnelere mermi fırlatmak gibi eylemleri başlatabilir. Aşağıdaki şekil, oyun süresi (tik) ilerledikçe gök cisimlerinin hareketini göstermektedir. Burada, ilk gök cisminin $t = 0$ oyun zamanında uzay izgarasına girdiğini ve oyuncunun (3, 1) hücresinden başlatıldığını ve bu on oyun zamanı boyunca hiç hareket etmediğini varsayıyoruz.



Şekil 1: Oyun süresi ilerledikçe gök cisimlerinin izgara üzerindeki hareketi.

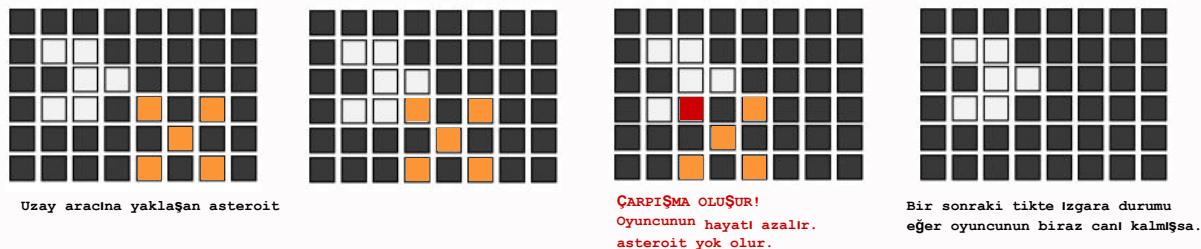
Oyuncu Hareketi ve Eylemleri: Oyuncu uzay aracını izgara sınırları içinde yukarı, aşağı, sola veya sağa hareket ettirebilir, ancak **her tik başına** yalnızca **bir hücre**. Izgara kenarının ötesine geçme girişimleri geçersiz kılır ve oyuncunun konumu değişmeden kalır. **SHOOT** komutu verildiğinde, uzay aracının orta sırasından, özellikle de uzay aracının hemen sağındaki sütundan bir mermi fırlatılır. Bu mermi, bir nesneyle çarpışana veya izgaradan çıkışa kadar sürekli olarak sağa doğru hareket eder. Uzay aracının yüksekliğinin her zaman tek sayı olacağını varsayıbilirsiniz, bu da merkez sırasının hesaplanması basitleştirir.



ASTEROİD ÇİZGİ

Çarpışma Tespiti: Oyun, gök cisimleri ile oyuncunun uzay aracı veya mermileri arasındaki çarpışmaları sürekli olarak kontrol etmelidir. Örneğin, gök cisimleri ilerlediğinde, oyuncunun uzay aracıyla veya bir mermi isabetiyle çarpışma uygun şekilde ele alınmalıdır.

Asteroidlerin Oyuncu Uzay Aracı ile Çarpışması: Bir asteroit oyuncunun uzay aracıyla çarpışırsa, oyuncunun kalan canı azalmalı ve asteroit uzay izgarasından kaldırılmalıdır. Eğer bu çarpışma oyuncunun son canını tüketirse, oyuncu da uzay izgarasından kaybolmalıdır. Şekil 2 ilk durumu göstermektedir. Oyun bitti senaryolarının ele alınmasıyla ilgili talimatlar için Bölüm 2.2.4'e bakın.

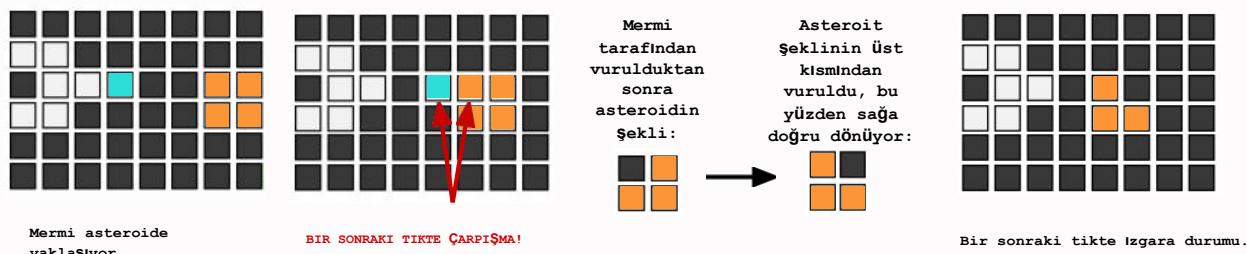


Şekil 2: Asteroit-oyuncu çarpışmalarının gösterimi.

Gök Cisimlerinin Mermilerle Çarpışması: Oyuncu tarafından ateşlenen mermiler, bir asteroitle karşılaşana veya izgaranın kenarına ulaşana kadar düz bir çizgide hareket eder. Mermi bir asteroide çarptığında, asteroidten tek bir hücreyi (temas ettiği ilk hücre) kaldırarak asteroidin şeklini dinamik olarak değiştirir. Bu değişiklik, gök cisminin tüm dönüşleri boyunca bir güncelleme yapılmasını ve bunların sıfırdan oluşturulmasını sağlamalıdır.

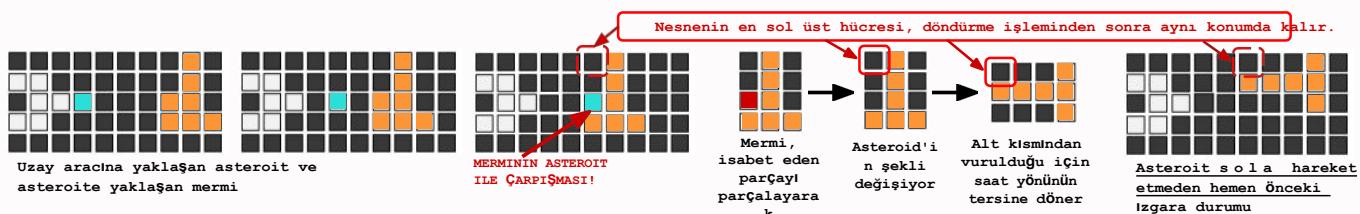
Bir asteroide isabet eden mermiin etkisi çarpma konumuna bağlıdır:

- Eğer mermi asteroidin **üst** kısmına çarparsa, asteroid saat yönünde (sağa) dönmelidir.
- Eğer mermi asteroidin **alt** kısmına çarparsa, saat yönünün tersine (sola) dönmelidir.
- Mermi **orta** bölüme çarparsa (**tek yüksekliğe sahip asteroidler için**), dönme meydana gelmemelidir.



Şekil 3: Asteroit çarpışmalarının gösterimi.

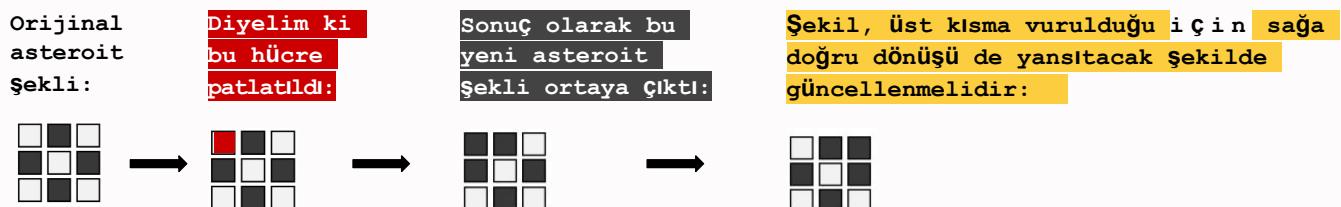
Bir gök nesnesini döndürürken, nesnenin en sol üst hücresini `space_grid` üzerinde aynı konumda tutmak önemlidir. Örneğin, bir gök cismi $n \times m$ boyutlarındaysa ve en sol üst hücresi `grid[i][j]`'de konumlanmışsa, döndürme işleminden sonra ($m \times n$ boyutıyla sonuçlanır), yeni döndürülen gök cisminin en sol üst hücresi bir sonraki tıkta ilerlemeden önce hala `grid[i][j]`'de bulunmalıdır. Bu durum aşağıdaki şekilde gösterilmiştir.



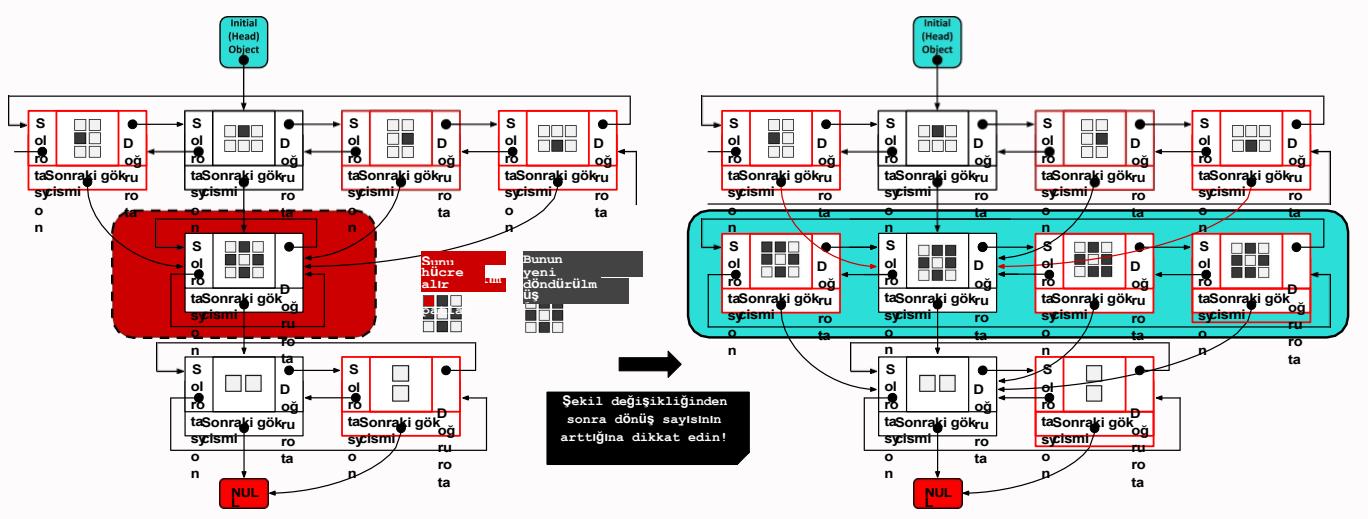
Şekil 4: Kare olmayan gök cinsi şekilleri için döndürme örnekleri.

ASTEROİD ÇİZGİ

Bunu başarmak için, gök cisimleri listesinde yapısal güncellemeler gereklidir. Bir asteroit bir mermi tarafından vurulduğunda, hücrelerinden biri patlayarak şeklini değiştirir. Bu değişiklik ilk olarak etkilenen **CelestialObject** örneğinin şekline uygulanmalıdır. **Daha sonra, nesnenin rotasyonlarının güncellenmiş şekilde göre yeniden hesaplanması gereklidir (ESKİ ROTASYONLARIN YER ALDIĞI HAFIZAYI BOŞALTMAKI UNUTMAYIN!).** Bazı durumlarda, bu değişiklik nesnenin toplam dönme sayısını etkileyebilir. Şekil 5 ve 6 bu süreci ve oyun içindeki gök cisimleri listesi üzerindeki etkisini göstermektedir.



Şekil 5: Bir mermi tarafından vurulduktan sonra nesnenin şeklinin ve dönüşlerinin güncellenmesi - adımlar.



Şekil 6: Bir mermi tarafından vurulduktan sonra nesnenin şeklinin ve rotasyonlarının güncellenmesi - bağlı liste asteroit için yeni rotasyonlarla değiştir.

Şimdi Şekil 7'de gösterildiği gibi ikinci bir merminin asteroide çarptığını düşünün.

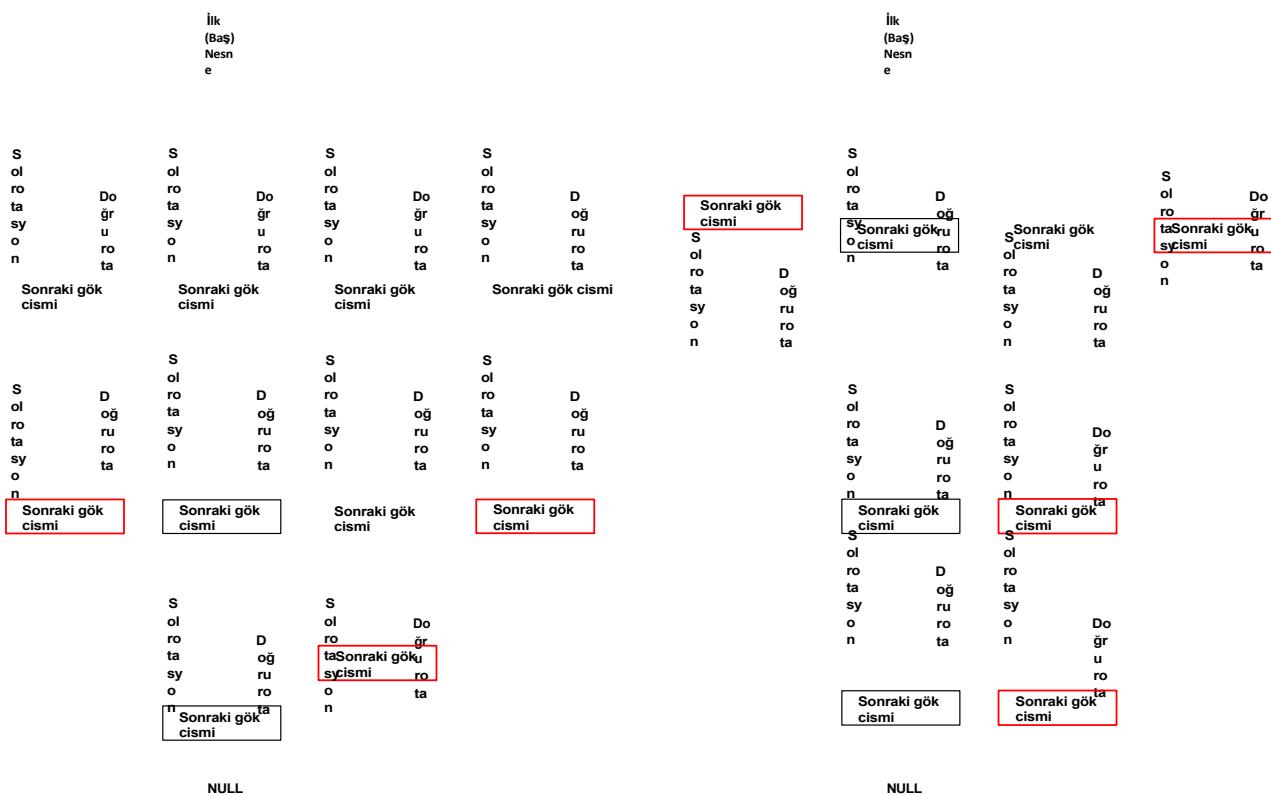


Şekil 7: Bir mermi tarafından ikinci kez vurulduktan sonra nesnenin şeklinin ve dönüşlerinin güncellenmesi.

Bağlı listedeki güncelleme de Şekil 8'de gösterildiği gibi bu değişikliği yansıtmalıdır.

Şimdi Şekil 9'da gösterildiği gibi üçüncü bir merminin asteroide çarptığını varsayıyalım. **Birden fazla merminin etkisi asteroidin parçalarının birbirinden ayrılmamasına neden olsa bile, asteroidin genel boyutlarını koruyarak tek bir varlık olarak davranışına devam edeceğini unutmamalıdır.**

ASTEROİD ÇİZGİ



Şekil 8: Bir mermi tarafından tekrar vurulduktan sonra nesnenin şeklinin ve dönüşlerinin güncellenmesi - bağlantılı liste asteroit için yeni dönüşlerle değişir.

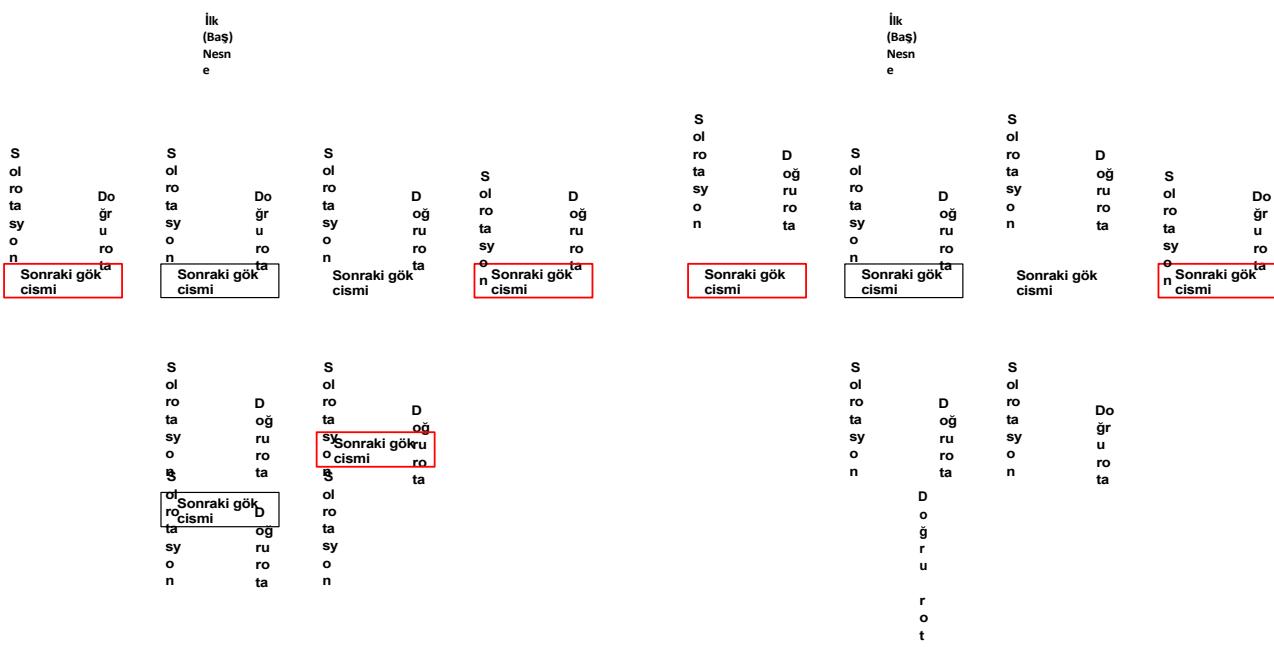
Üçüncü
vuruştan
Önceki
asteroit
Şekli:

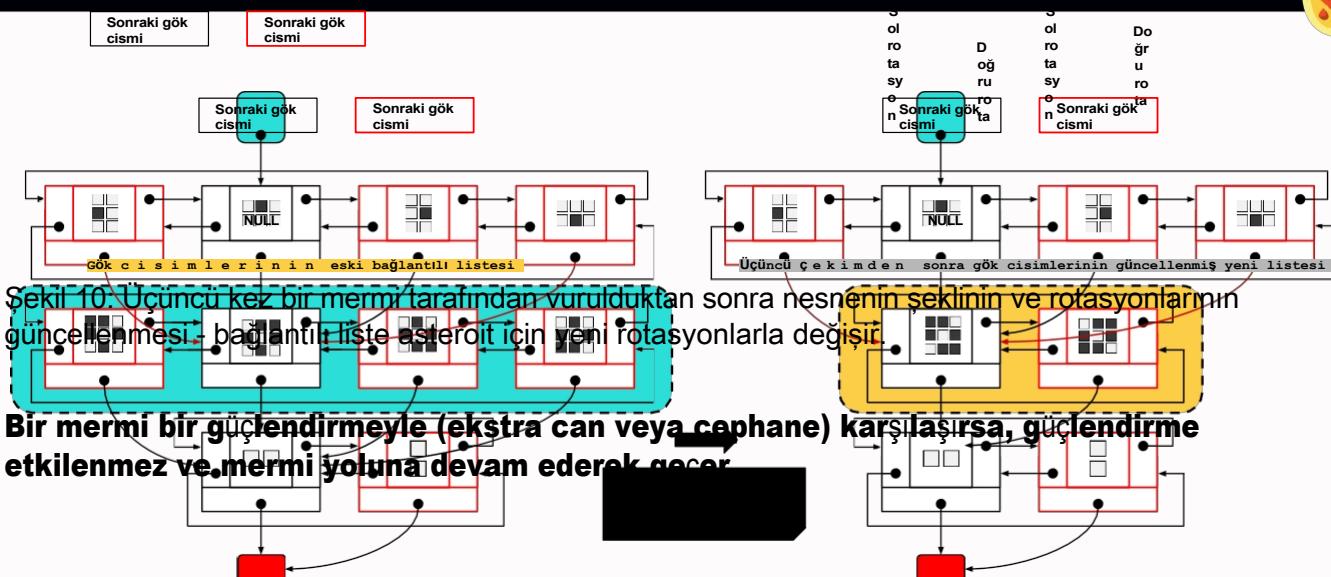
**Şimdi bu
hücrenin
patlatıldığını
varsayılmı:**

**Sonuç olarak bu
yeni asteroit
Şekli ortaya çıktı:**

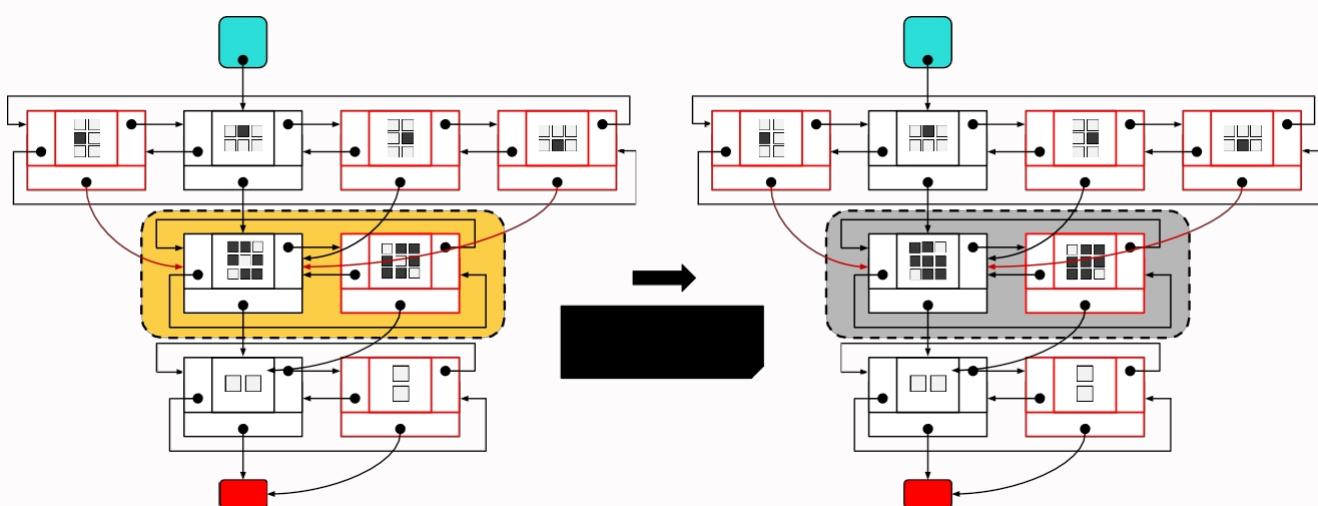
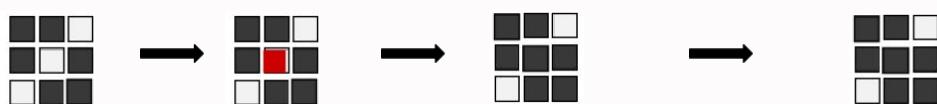
**Şekil bu kez
döndürülmeyecektir çünkü
Şeklinin ortasından
vurulmuştur:**

Şekil 9: Üçüncü kez bir mermi tarafından vurulduktan sonra nesnenin şeklinin ve dönüşlerinin güncellenmesi.





Hacettepe Bilgisayar Mühendisliği - BBM203 Yazılım Pratiği I - Güz 2024
Programlama Ödevi 2 - Son Tarih: 15/11/2024 23:59:59



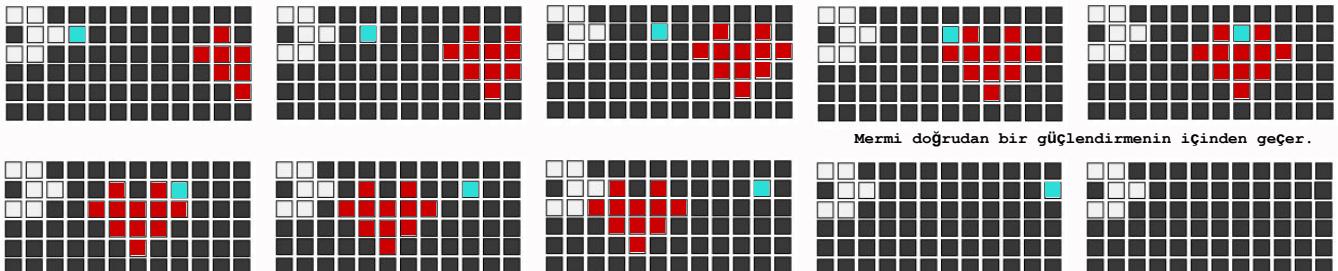


Güç-Up Etkileşimleri:

Güçlendiriciler, gök cisimleri girdi dosyasında tanımlandığı gibi, toplandıklarında (uzay aracıyla çarpışıklarında) oyuncuya belirli faydalalar sağlar:

- **e:life** - Can Artışı - Oyuncunun toplam canını bir artırır (can sayısında sınır yoktur).
- **e:ammo** - Cephane Doldurma - Oyuncunun cephanesini **max_ammo** değerine kadar doldurur.

Bir güçlendirme oyuncu tarafından toplandığında, çarpışmanın ardından uzay ızgarasından kaybolan bir asteroide benzer şekilde davranışır. Şekil. 11 güçlendirmelerle olan bu etkileşimleri göstermektedir.

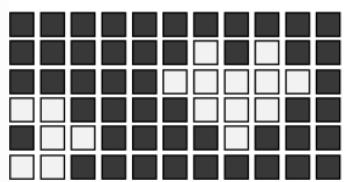


Şekil 11: Güçlendirmelerle etkileşim.

Izgarayı Yazdırma:

PRINT_GRID komutu ilk olarak oyun süresini tık olarak oyuncunun kalan canını, cephanesini ve mevcut skorunu göstermelidir. Daha sonra, tüm zamanların en yüksek skorunu göstermelidir. Eğer bu oynanan ilk oyunsa veya oyuncunun mevcut skoru mevcut rekoru geçerse, yeni skor kaydedilmeli ve tüm zamanların en yüksek skoru olarak gösterilmelidir. Bu işlemden sonra ızgara, tüm aktif nesneleri, oyuncunun uzay aracını, mermileri ve diğer işgal edilmiş hücreleri doğru bir şekilde temsil edecek şekilde satır satır yazdırılmalıdır. **PRINT_GRID** komutunun örnek bir çıktısı sağda gösterilmektedir.

Tick: 29
Lives: 3
Ammo: 7
Score: 29
High Score: 154



2.2.2 Puanlama Sistemi

AsteroidDash'teki puanlama sistemi, oyuncuları gök cisimleriyle etkileşimlerine ve oyun boyunca gerçekleştirdikleri eylemlere göre ödüllendirmek üzere tasarlanmıştır. Puanlama dökümü aşağıdaki gibidir:

Eylem	Puan Etkisi
Mermi Asteroide Çarptı	Bir mermi başarılı bir şekilde bir asteroide çarparak ondan bir hücre çıkardığında, oyuncu 10 puan kazandırır. Eğer vuruş bir asteroidin kalan son hücresini de ortadan kaldırırsa, bu 10 puanları da bonus puanlarından önce verilir (sonraki satırı bakın).
Tüm Asteroidi Yok Etme	Bir asteroidin tüm hücreleri kaldırıldığında, oyuncu 100 puanlık bir bonus alır asteroidin orijinal boyutunun İşgal edilen her hücresi için, tamamen yok edilmeyi ödüllendirmek asteroitlerin.
Her Oyun Kene'sinde Hayatta Kalmak	Oyuncu, bir asteroide çarpmadan hayatı kalladığı her tık için 1 puan kazanır. Bu hayatı kalma bonusu, hasar almadan uzun süreli oyunu teşvik eder.
Yeni Yüksek Skor	Oyuncunun mevcut skoru tüm zamanların en yüksek skorunu aşarsa, bu yeni rekor olur, liderlik tablosunda görüntülenir.

Oyuncunun son **skoru** oyunun sonunda kaydedilir ve tüm zamanların en yüksek on skorundan biri olması halinde uzun süreli takip için **Liderlik Tablosuna** girilir.

zaman damgası (giriş oluşturulduktan sonra time(nullptr) tarafından elde edilir) ve **oyuncu adı** içerir. Daha önce



ASTEROID ÇİZGİ

Her oyun oturumunda sistem, eğer varsa, *beşinci komut satırı argümanı* olarak verilen mevcut bir dosyadan liderlik tablosunu yüklemeye çalışmalıdır. Ancak, bu dosya ilk çalışma sırasında mevcut olmayabilir, bu da yüksek puanların bulunmadığını gösterir. Kodunuz bunu ele almalıdır. Bir oyun oturumu sona erdiğinde, oyunun sona erme nedeni ne olursa olsun, liderlik tablosunun en son skorları yansıtacak şekilde yenilenmesi ve ardından aynı liderlik tablosu dosyasına geri kaydedilmesi gereklidir. Bu liderlik tablosu, tüm oturumlarda en fazla **10** en yüksek puanı tutar ve puanlara göre azalan sırada tutarlı bir şekilde sıralanmalıdır. Ondan daha az yüksek skorun saklandığı durumlar olabileceğini unutmamak önemlidir. Bu tür senaryolarda, on girişi ayırtırma girişimlerinden kaçının. Bunun yerine, bellek hatalarını önlemek için esnek bir yaklaşım benimseyin.

Metin dosyası içeriği aşağıdaki gibi yapılandırılmalıdır:

```
<score> <timestamp> <player_name>
```

Bir örnek olarak:

```
40000 1697910655 AsteroidBuster  
1200 1697910655 StackOverthrower  
...
```

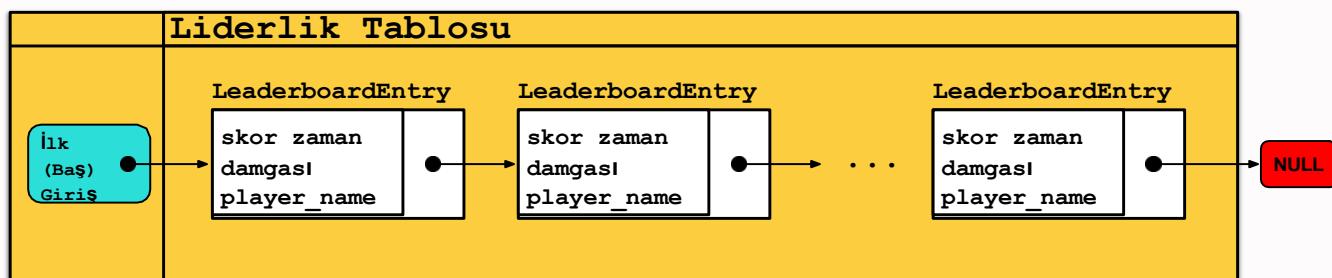
Ayrıca, lider tablosu STDOUT'a aşağıdaki formatta yazdırılmalıdır:

```
Liderlik Tablosu  
-----  
<#order>. <player_name> <score> <timestamp formatted as %H:%M:%S/%d.%m.%Y>
```

Bir örnek olarak:

```
Liderlik Tablosu  
-----  
1. AsteroidBuster 40000 20:50:55/21.10.2024  
2. StackOverthrower 1200 20:50:55/21.10.2024  
...
```

Leaderboard sınıfının **head_leaderboard_entry** adında ilk (en üst) **LeaderboardEntry** için bir işaretçisi vardır. Henüz yüksek skor yoksa NULL olacaktır. Leaderboard, her giriş bir sonraki yüksek skora işaret edecek şekilde **LeaderboardEntry** örneklerinin bağlantılı bir listesi olarak saklanmalıdır.



Lider tablosu, tüm zamanların en yüksek on skorunu tutacak şekilde tasarlanmıştır ve skor sırasını korurken ekleme ve silme işlemlerini verimli bir şekilde yönetebilen dinamik bir uygulama gerektirir. Bu nedenle, yeni yüksek puanları liderlik tablosuna doğru bir şekilde entegre eden işlevler geliştirmekle görevlendirildiniz. Bu, yeni girişler için dinamik olarak bellek ayırmayı ve ilk on sıralamadan çıkarılanlarla ilişkili belleği sorumlu bir şekilde ayırmayı içerir. Bu işlemlerin liderlik tablosunun bütünlüğünü ve oyuncu başarısının gerçek zamanlı yansımmasını korumasını sağlamak çok



ASTEROID ÇİZGİ

2.2.4 Oyun Bitti

Oyun, aşağıdaki sonlandırma koşullarından biri karşılanana kadar ilerlemeye devam edecektir:

- Komutları içeren girdi dosyası tamamen okunur ve yürütülecek başka talimat bırakılmaz.
- Bir asteroitle her çarışma oyuncunun can sayısını sıfıra ulaşana kadar azalttığı için oyuncunun canı tükenir.

Oyun, oyuncu bir asteroide çarpıp son canını kaybettiğinde ya da uygulanacak başka komut kalmadığında sona erer. Oyunun sonunda, son ızgara düzeni, mevcut skor ve diğer ilgili oyun istatistikleri görüntülenir. Ek olarak, oyuncunun skoru yeni bir yüksek skor olarak nitelendirilirse, liderlik tablosu buna göre güncellenir.

Çıktı Formatı: Oyun sonlandırma senaryosuna bağlı olarak, program sonlandırmaya neden olan özel durumu yansıtmak için farklı çıktılar üretecektir. Her çıktı biçimi, aşağıdaki örnek şekilde gösterildiği gibi mevcut skoru, kalan canları ve şebekenin durumunu detaylandırmalıdır.

GAME OVER!
Tick: 29
Lives: 0
Ammo: 7
Score: 26
High Score: 40000
Player: NullNinja

Leaderboard

1. AsteroidBuster 40000 20:50:55/21.10.2024
2. StackOverthrower 1200 20:51:23/20.10.2024
3. NullNinja 26 22:20:50/26.10.2024

GAME FINISHED! No more commands!
Tick: 20
Lives: 3
Ammo: 10
Score: 40
High Score: 40000
Player: SpaceExplorer

Leaderboard

1. AsteroidBuster 40000 20:50:55/21.10.2024
2. StackOverthrower 1200 20:51:23/20.10.2024
3. SpaceExplorer 40 22:21:42/26.10.2024
4. NullNinja 26 22:20:50/26.10.2024

2.3 Atama Uygulama Görevleri ve Gereksinimleri

Bu bölümde, uygulamanız gereken sınıfları ve işlevleri ana hatlarıyla belirtiyoruz. Lütfen kodunuzun açıklık, kapsülleme ve tutarlılığı koruyarak sağlanan başlangıç dosyalarının yapısına bağlı kaldığından emin olun. Şablonda belirtilen işlevlerin veya üye değişkenlerin adlarını veya imzalarını değiştirmeyin. Ancak, gerektiğinde ek fonksiyonlar veya değişkenler ekleyebilirsiniz.

2.3.1 CelestialObject Sınıfı

Bu sınıf, oyun içindeki asteroitler veya güçlendiriciler gibi göksel nesneleri temsil eder.

- Kurucu:

CelestialObject(const vector<vector<bool>& shape, ObjectType type, int start_row, int time_of_appearance)
- Nesneyi şekli, türü, başlangıç satırı ve görünüm zamanı ile başlatın.

- Kopya Oluşturucu:

CelestialObject(const CelestialObject *other)
- Kopyalama nesnesini mevcut bir nesneden başlatın.



ASTEROID ÇİZGİ

- Fonksiyon:

```
void delete_rotations()
```

- Şeklindeki değişikliklerden sonra gök cisminin tüm eski rotasyonlarını silin (bunlar için dinamik olarak ayrılmış belleği boşaltın).

2.3.2 AsteroidDash Sınıfı

Bu sınıf ana oyun mantığını ve yapısını temsil eder.

- Kurucu:

```
AsteroidDash(const string &space_grid_file_name, const string  
    ↳ &celestial_objects_file_name,  
    const string &leaderboard_file_name, const string &player_file_name, const  
    ↳ string &player_name)
```

- Belirtilen girdi dosyalarını kullanarak uzay ızgarasını, oyuncuyu, göksel nesneleri ve liderlik tablosunu yükleyerek oyunu başlatır.

- Fonksiyon:

```
void print_space_grid() const
```

- Boşluğu STDOUT'a yazdırır.

- Fonksiyon:

```
void read_space_grid(const string &input_file)
```

- Girdi dosyasından uzay ızgarasını okur ve kurar, oyunun oynanabilir alanını başlatır.

- Fonksiyon:

```
void read_player(const string &player_file_name, const string &player_name)
```

- Oyuncu bilgilerini bir dosyadan okur ve oyuncuyu ızgara içinde başlatır.

- Fonksiyon:

```
void read_celestial_objects(const string &input_file)
```

- Belirtilen dosyadan göksel nesneleri okur ve oyun için göksel nesnelerin bağlantılı listesini ayarlar.

- Fonksiyon:

```
void update_space_grid()
```

- Her tıkta uzay ızgarasını güncelliyor oyuncunun, gök cisimlerinin ve mermilerin hareketlerini ve çarpışmalarını yönetir.

- Fonksiyon:

```
void shoot()
```

- Oyuncunun mevcut konumundan bir mermi başlatır ve oyunun aktif mermilerine ekler.

- Yok edici:

```
~AsteroidDash()
```

- Göksel nesneler, oyuncu ve diğer oyun bileşenleri için dinamik olarak ayrılmış belleği temizler.



ASTEROİD ÇİZGİ

2.3.3 GameController Sınıfı

Bu sınıf, oyunun yönetilmesinden ve bir girdi dosyasından gelen komutların yorumlanması sorumludur.

- **Kurucu:**

```
GameController(const string &space_grid_file_name, const string  
    ↳ &celestial_objects_file_name,  
    const string &leaderboard_file_name, const string &player_file_name, const  
    ↳ string &player_name)
```

- AsteroidDash oyun örneğini verilen girdi dosyalarıyla başlatır, uzay ızgarasını, göksel nesneleri, oyuncuyu ve liderlik tablosunu ayarlar.

- **Fonksiyon:**

```
void play(const string &commands_file)
```

- Belirtilen dosyadaki komutları okuyarak ve yorumlayarak oyunu yürütür, sağlanan komutlara göre her tıklamada oyun durumunu günceller.

- **Yok edici:**

```
~GameController()
```

- AsteroidDash oyun örneği için dinamik olarak ayrılmış belleği temizler.

2.3.4 Oyuncu Sınıfı

Oyuncu sınıfı, oyuncunun uzay aracını temsil eder ve hareket ve ateş etme dahil olmak üzere oyuncunun eylemlerini yönetir.

- **Kurucu:**

```
Oyuncu(const vector<vector<bool> &shape, int row, int col, const string  
    ↳ &player_name, int max_ammo = 10, int lives = 3)
```

- Oyuncuyu verilen uzay aracı şekli, başlangıç konumu, oyuncu adı, maksimum cephanе ve can ile başlatır.

- **Fonksiyon:**

```
void move_left()
```

- Oyuncunun uzay aracını ızgara sınırları içinde bir hücre sola hareket ettirir.

- **Fonksiyon:**

```
void move_right(int grid_width)
```

- Oyuncunun uzay aracını ızgara sınırları içinde bir hücre sağa hareket ettirir.

- **Fonksiyon:**

```
void move_up()
```

- Oyuncunun uzay aracını ızgara sınırları içinde bir hücre yukarı taşıır.

- **Fonksiyon:**

```
void move_down(int grid_height)
```

- Oyuncunun uzay aracını ızgara sınırları içinde bir hücre aşağı hareket ettirir.

- **Yok edici:**

```
~Oyuncu()
```

- Oyuncu için ayrılmış tüm kaynakları temizler.



ASTEROID ÇİZGİ

2.3.5 LeaderboardEntry Sınıfı

Bu sınıf, liderlik tablosundaki tek bir giriş temsil eder.

- **Kurucu:**

```
LeaderboardEntry(unsigned long score, time_t lastPlayed, const string  
    ↴ &playerName)
```

- Skor, zaman damgası ve oyuncu adı ile bir liderlik tablosu girişini başlatın.

2.3.6 Liderlik Sınıfı

Bu sınıf liderlik tablosu girişlerini yönetir.

- **Fonksiyon:**

```
void insert(LeaderboardEntry *new_entry)
```

- Listeyi puana göre azalan sırada tutarak yeni bir giriş ekleyin. Liderlik tablosunu 10 girişle sınırlayın.

- **Fonksiyon:**

```
void read_from_file(const string& filename)
```

- Liderlik tablosunu bir dosyadan yükleyin.

- **Fonksiyon:**

```
void write_to_file(const string& filename)
```

- Geçerli liderlik tablosunu bir dosyaya yazın.

- **Fonksiyon:**

```
void print_leaderboard()
```

- Liderlik tablosunu konsola yazdırın.

- **Yok edici:**

```
~Leaderboard()
```

- Liderlik tablosu girişleri için dinamik olarak ayrılan belleği temizleyin.

Kullanılması Gereken Başlangıç Kodları

Bu başlangıç (şablon) kodunu kullanmalısınız. Tüm başlıklar ve sınıflar doğrudan **zip** arşivinizin içine yerleştirilmelidir.

Notlandırma Politikası

- Bellek sızıntısı ve hata yok: 10%
 - Bellek sızıntısı yok: 5%
 - Bellek hatası yok: 5%
- Oyunun uygulanması: 80%
 - Uygun oyun ızgarası ve oyuncu başlatma: 5%



ASTEROID ÇİZGİ

- Oyun nesneleri ve bunların rotasyonları için çok düzeyli bağlı liste yapısının ve ilgili işlemlerin doğru şekilde uygulanması: 15%
 - Oyuncu hareketlerinin doğru uygulanması: 5%
 - Çekimin doğru uygulanması ve sonuçta ortaya çıkan nesne şekli ve rotasyon güncellemesi: %15
 - Nesne hareketlerinin ve çarpışmaların doğru uygulanması: 7.5%
 - Güç açma işleminin doğru uygulanması: 5%
 - Puanlama mekanizmasının doğru uygulanması: %7,5
 - Liderlik tablosu bağlantılı listesinin ve ilgili işlemlerin doğru uygulanması: 15%
 - Uygun oyun sonlandırma: 5%
- Çıkış testleri: 10%

Önemli Notlar

- Son başvuru tarihini kaçırmayın: **15.11.2024 Cuma (23:59:59)**.
- Ödeve not verilene kadar tüm çalışmalarınızı saklayın.
- Gönderdiğiniz ödev çözümü sizin özgün, bireysel çalışmanız olmalıdır. Kopya veya benzer ödevlerin her ikisi de kopya olarak değerlendirilecektir.
- Sorularınızı Piazza (<https://piazza.com/hacettepe.edu.tr/fall2024/bbm203>) üzerinden sorabilirsiniz ve Piazza'da tartışılan her şeyden haberdar olmanız gerekmektedir.
- Kodunuzu **Tur⁶ Bo Grader** <https://test-grader.cs.hacettepe.edu.tr/> üzerinden test etmelisiniz.
(teslim olarak sayılmaz!).
- Çalışmanızı <https://submit.cs.hacettepe.edu.tr/> adresi üzerinden aşağıda verilen dosya hiyerarşisi ile göndermeniz gerekmektedir:
 - **b<studentID>.zip**
 - * AsteroidDash.h <DOSYA>
 - * AsteroidDash.cpp <DOSYA>
 - * CelestialObject.h <DOSYA>
 - * CelestialObject.cpp <DOSYA>
 - * GameController.h <DOSYA>
 - * GameController.cpp <DOSYA>
 - * LeaderboardEntry.h <DOSYA>
 - * LeaderboardEntry.cpp <DOSYA>
 - * Leaderboard.h <DOSYA>
 - * Leaderboard.cpp <DOSYA>
 - * Player.h <DOSYA>
 - * Player.cpp <DOSYA>
- **Bu başlangıç kodunu kullanmalısınız.** Tüm sınıflar doğrudan **zip** arşivinize yerleştirilmelidir.
- Bu dosya hiyerarşisi gönderilmeden önce sıkıştırılmalıdır (.rar değil, yalnızca .zip dosyaları desteklenir).



ASTEROID ÇİZGİ



Çalıştırma Yapılandırması

İşte kodunuzun nasıl derleneceğine dair bir örnek (main.cpp yerine test dosyalarımızı kullanacağımıza dikkat edin):

```
$ g++ -std=c++11 -g main.cpp AsteroidDash.h AsteroidDash.cpp CelestialObject.h  
    CelestialObject.cpp GameController.h GameController.cpp LeaderboardEntry.h  
    ↳ LeaderboardEntry.cpp Leaderboard.h Leaderboard.cpp Player.h Player.cpp -o  
    ↳ AsteroidDash
```

Veya kodunuzu derlemek için örnek girdi içinde sağlanan Makefile veya CMakeLists.txt dosyasını kullanabilirsiniz:

```
$ make
```

veya

```
$ mkdir AsteroidDash_build  
$ cmake -S . -B AsteroidDash_build/  
$ make -C AsteroidDash_build/
```

Derleme işleminden sonra programı aşağıdaki şekilde çalıştırabilirsiniz:

```
$ ./AsteroidDash space_grid.dat celestial_objects.dat player.dat commands.dat  
↳ leaderboard.txt AsteroidBuster
```

Akademik Dürüstlük Politikası

Ödevler üzerindeki tüm çalışmalar **bireysel olarak yapılmalıdır**. Verilen ödevleri sınıf arkadaşlarınızla tartışmanız teşvik edilmektedir, ancak bu tartışmalar soyut bir şekilde yürütülmeli. Yani, belirli bir problemin belirli bir çözümüyle ilgili tartışmalara (gerçek kodda veya sözde kodda) **tolerans gösterilmeyecektir**. Kısacası, bir başkasının çalışmasını (internette bulunan çalışmalar **da** dahil olmak üzere) tamamen veya kısmen kendi çalışmanız olarak teslim **etmeniz akademik dürüstliğin ihlali** olarak değerlendirilecektir. Web'de bulunan her şey bir başkası tarafından yazıldığından, önceki koşulun web'de bulunan materyal için de geçerli olduğunu lütfen unutmayın.



Başvurular benzerlik kontrolüne tabi tutulacaktır. Benzerlik kontrolünden geçemeyen başvurulara not verilmeyecek ve akademik dürüstlük ihlali vakası olarak etik kurula bildirilecek, bu da ilgili öğrencilerin uzaklaştırılmasıyla sonuçlanabilecektir.

Minik Ödüllerle Bonus Mücadelesi

İlk üç öğrenci

- ödevi başarıyla tamamlayın (sınavdan 100 puan alın)
- Tur⁶ Bo Grader**) ve
- oyunları için interaktif bir arayüz geliştirenler

minik ödüllerle ödüllendirilecek.

İşte uygulamamız iş başında: [Oynamış Kaydi](#)

