



CPU ve GPU Mimarilerinin Simgesel Savaşı: Piksel Tabanlı Bir GörSEL Simülasyon

Merkezi İşlem Biriminin (Seri) ve Grafik İşlem Biriminin (Paralel) çalışma
prensiplerini görselleştiren bir yüksek lisans projesi.

Hazırlayanlar: Ahmet Gökhan Kocaman (240404006), Bahadır Bekeç (240404008)

Danışman: ELİF PINAR HACIBEYOĞLU

Tarih: 26 Aralık 2025

Teoriyi Anlamak Kolay, Gözlemlmek Zordur

Bilgisayar mimarisi eğitiminde temel bir zorluk vardır: CPU'nun "az sayıda güçlü çekirdeği" ile GPU'nun "binlerce küçük çekirdeği" arasındaki fark, öğrenciler için genellikle soyut bir kavram olarak kalır. Mikrosaniyeler içinde gerçekleşen donanım süreçlerini gözle görmek imkansızdır.

Vurgulanan Sorunlar

- ✖ **Gözlemlenemeyen Hız:** Gerçek donanım işlemleri, insan algısının çok ötesinde bir hızda tamamlanır.
- ✖ **Soyut Bilgi:** 'Low Latency' ve 'High Throughput' gibi kavamların pratikteki karşılığını canlandırmak zordur.
- ✖ **Pratik Eksikliği:** Öğrenciler, bu iki farklı mimarinin bir veri setini nasıl işlediğini canlı olarak deneyimleyemez.

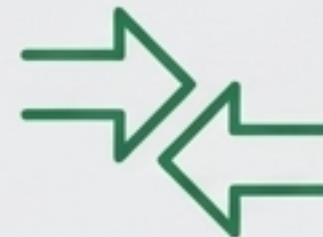


Projenin Üç Temel Amacı



Görselleştirme

Milisaniyeler içinde biten görüntü işlemeye süreçlerini, insan gözünün algılayabileceği bir hızda yavaşlatarak simüle etmek. Soyut olanı somut hale getirmek.



Karşılaştırma

Seri (Sequential) ve Paralel (Parallel) algoritmaların, bir veri kümesi üzerindeki gezinme stratejilerini (Scanning vs. Random Access) net bir şekilde yan yana göstermek.



Eğitim Materyali

Yüksek lisans ve lisans seviyesindeki bilgisayar mimarisi derslerinde, karmaşık donanım kavramlarını öğretmek için kullanılabilecek etkileşimli ve kalıcı bir araç geliştirmek.

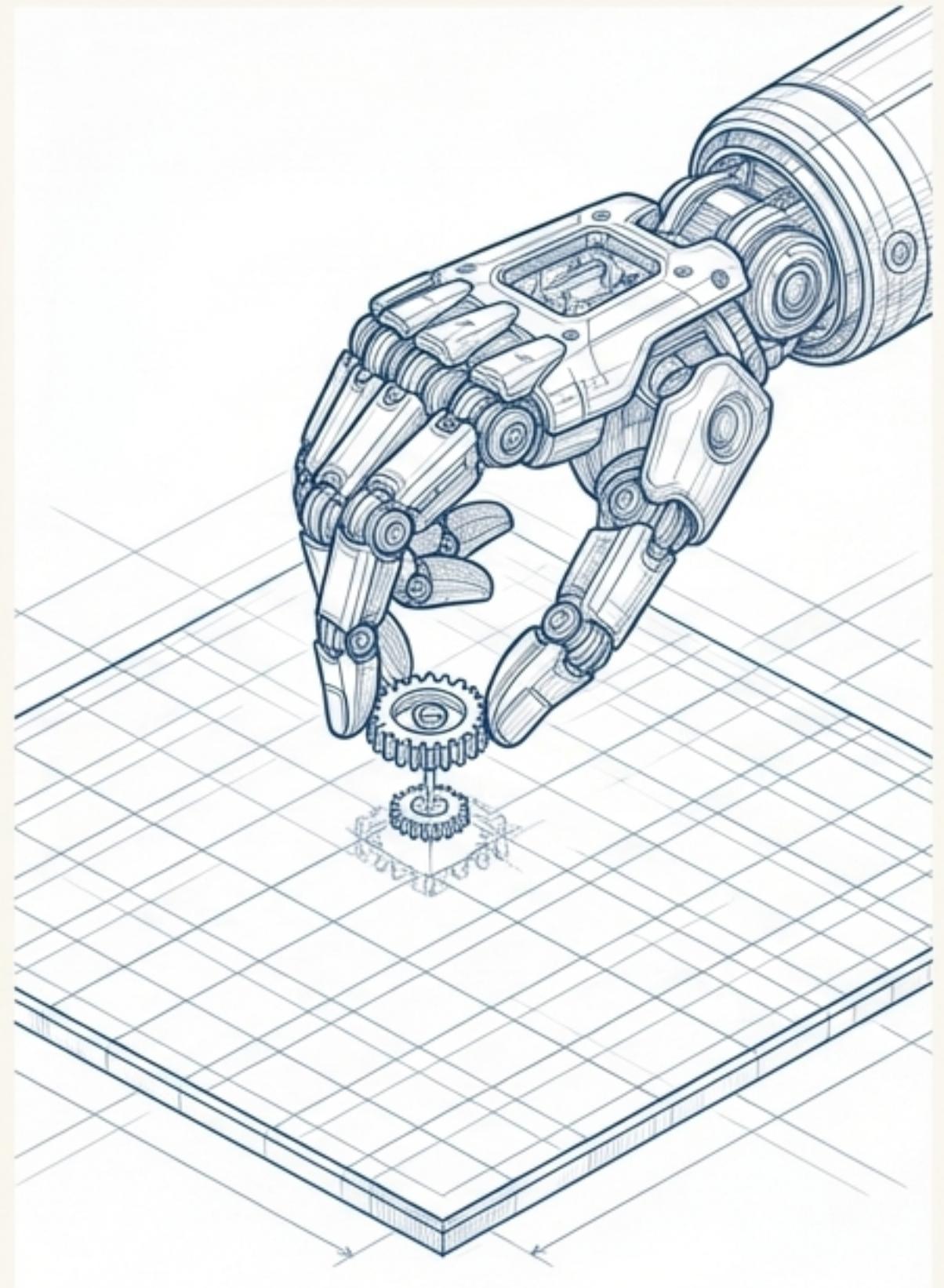
Birinci Oyuncu: CPU (Merkezi İşlem Birimi)

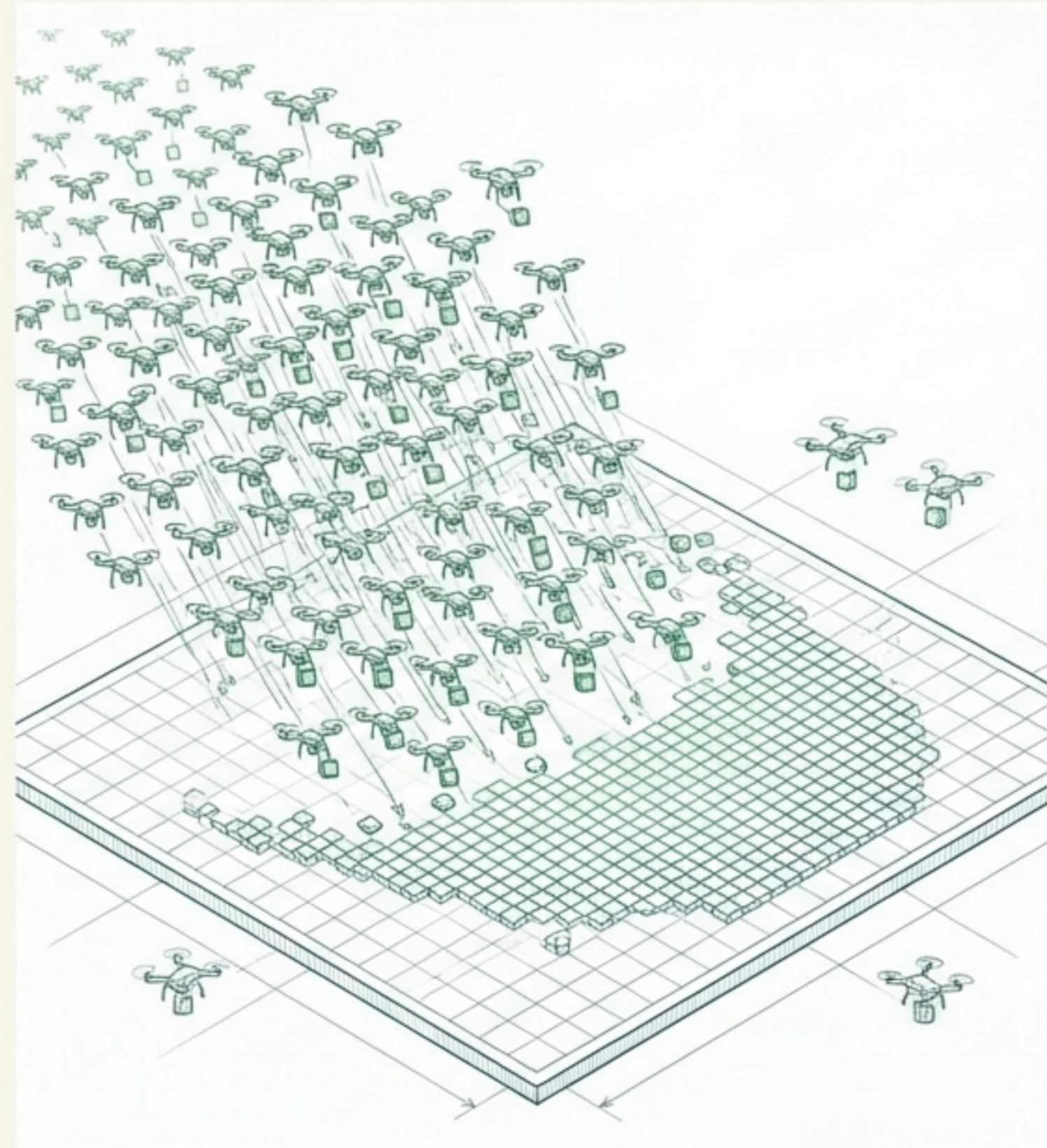
Usta Zanaatkâr: Hassas ve Sıralı

CPU, düşük gecikme süresi (low latency) ve karmaşık mantıksal görevler için optimize edilmiştir. Az sayıda (örneğin 4-16) ancak çok güçlü ve yüksek saat hızına sahip çekirdeklerden oluşur.

Çalışma Prensibi: Verileri bir sıraya koyar ve her görevi başından sonuna kadar tek tek, yüksek hassasiyetle işler.

Simülasyondaki Temsili: Projemizde CPU, görüntüyü sol üst köşeden başlayarak satır satır, düzenli bir şekilde işleyen tek tek bir 'işçi' (thread) olarak modellenmiştir.





İkinci Oyuncu: GPU (Grafik İşlem Birimi)

Paralel Ordu: Kitlesel ve Eşzamanlı

GPU, yüksek işlem hacmi (high throughput) için tasarlanmıştır. Binlerce daha basit ve daha küçük çekirdekten oluşur. Tek bir çekirdeği CPU'dan yavaş olsa da, hepsi aynı anda çalışlığında muazzam bir işlem gücü ortaya çıkarır.

Çalışma Prensibi: Büyük bir görevi binlerce küçük parçaya böler ve tüm parçaları aynı anda işler.

Simülasyondaki Temsili: Projemizde GPU, görüntünün farklı bölgelerine aynı anda müdahale eden yüzlerce 'işçi' olarak modellenmiştir.

Çözümümüz: Etkileşimli Bir Simülasyon Ortamı

Bu iki farklı mimariyi görselleştirmek için Python programlama dili ve Pygame kütüphanesi kullanılarak bir simülasyon aracı geliştirildi. Amacımız, gerçek donanım hızlarını yapay gecikmelerle yavaşlatarak, altta yatan algoritmik yaklaşımı gözlemlenebilir kılmaktı.

Teknoloji Yığını



Python 3.x: Hızlı prototipleme ve yüksek okunabilirlik için.



Pygame: 2D grafik çizimi ve olay döngüsü yönetimi için.



Numpy: Matris işlemleri ve piksel verisinin verimli sayısal yönetimi için.

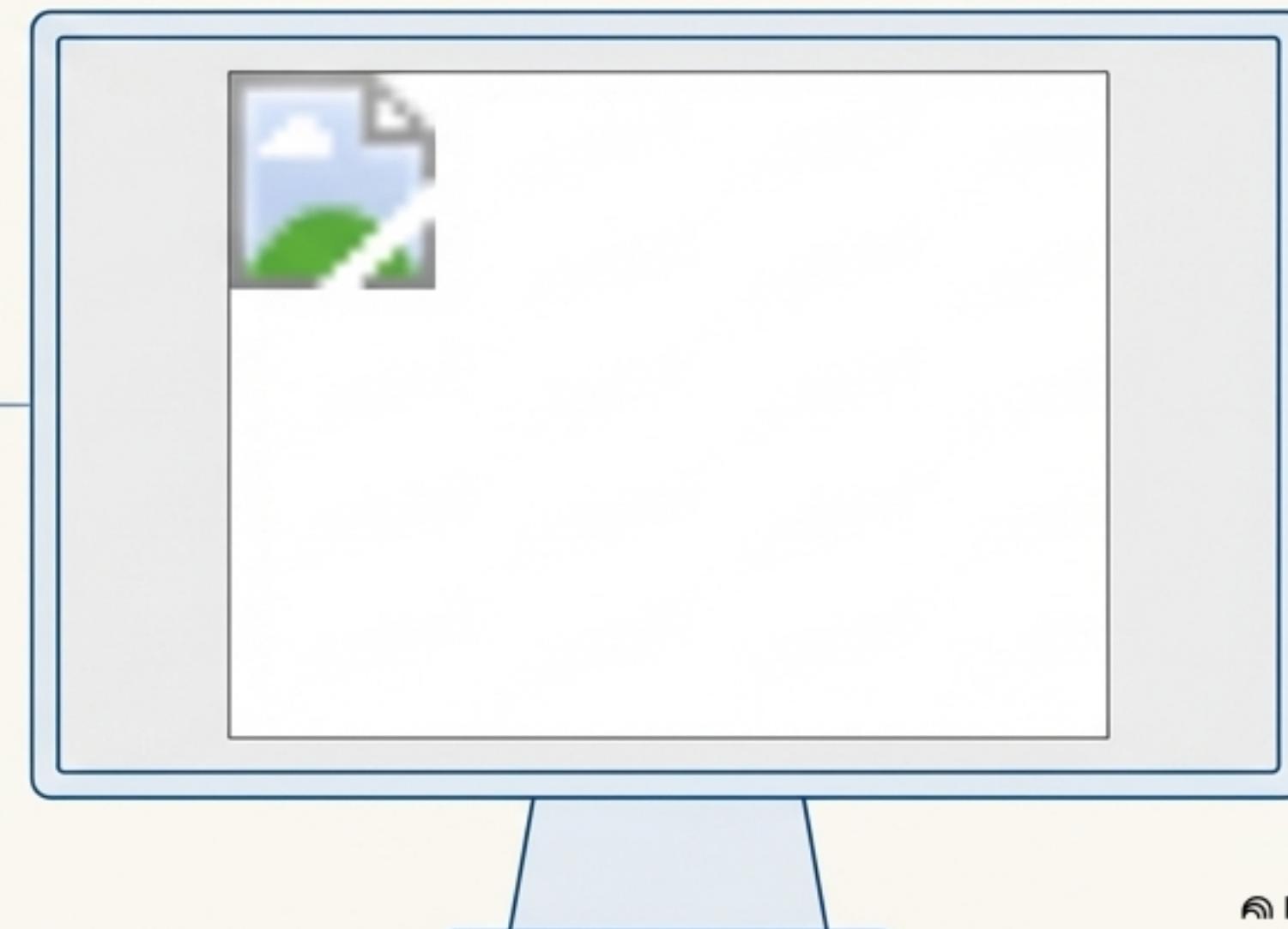
Kodun Arkasındaki Mantık: CPU'nun Sıralı Erişimi

Algoritma

```
# CPU Algoritması (Nested Loops)
for y in range(height):
    for x in range(width):
        pixel = get_color_from_source(x, y)
        draw_pixel(x, y, pixel)
        update_screen() # Her pikselden sonra güncelleme
```

Açıklama ve Görselleştirme

Bu iç içe geçmiş döngü yapısı, CPU'nun bellekteki verilere sıralı (doğrusal) erişimini mükemmel bir şekilde temsil eder. Her piksel bir öncekinin işi bittikten sonra çizilir.



Kodun Arkasındaki Mantık: GPU'nun Paralel Hücumu

Algoritma

```
# GPU Algoritması (Batch Processing)
all_pixels = get_all_pixel_coordinates()
random.shuffle(all_pixels) # Rastgele erişim simülasyonu

while pixels_remain:
    # 500 çekirdekli bir GPU'yu simüle et
    batch = get_next_batch(all_pixels, 500)
    for pixel_coords in batch:
        draw_pixel(pixel_coords)

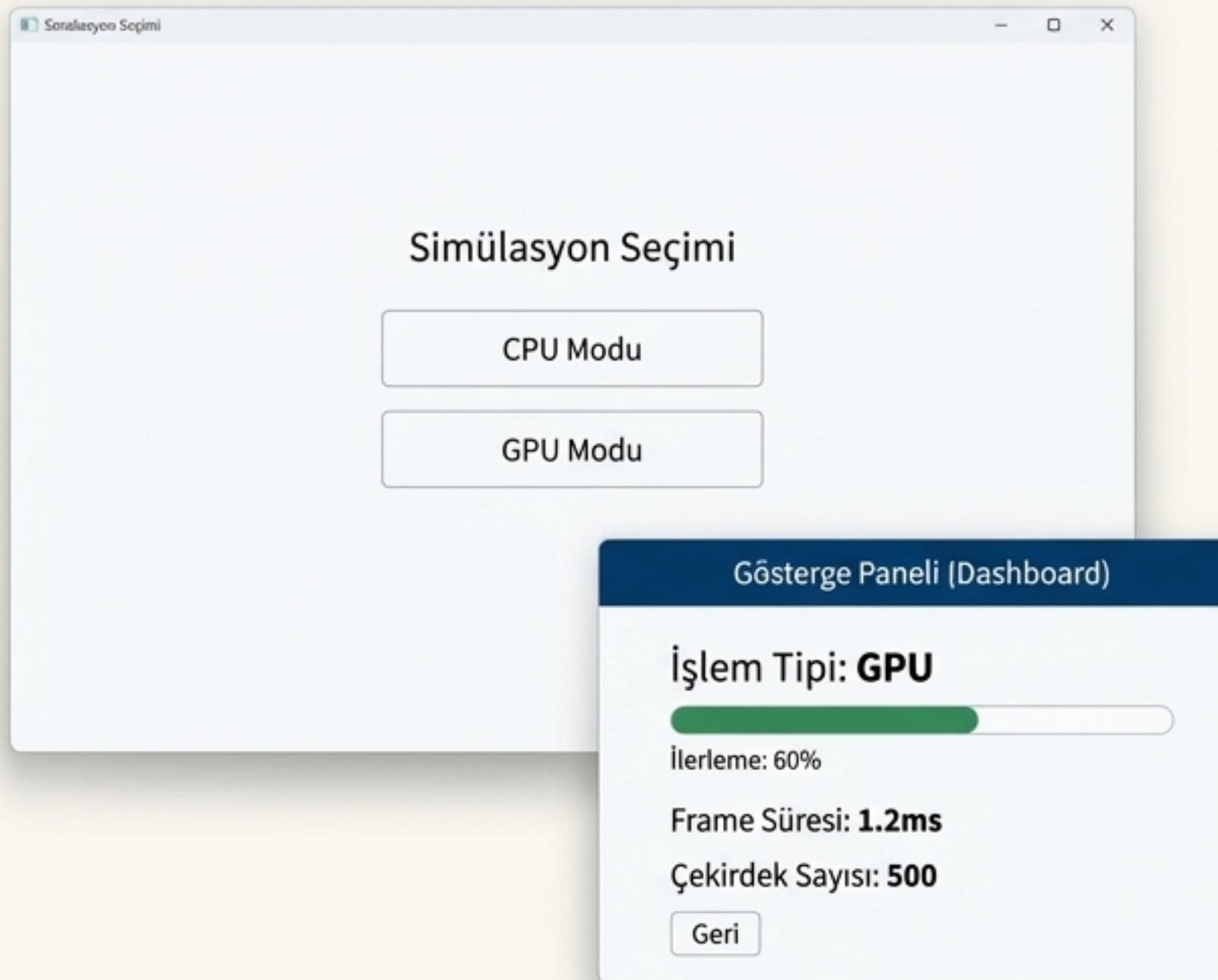
    update_screen() # Tüm grup işlendikten sonra güncelleme
```

Açıklama ve Görselleştirme

Bu algoritma, GPU'nun SIMD (Single Instruction, Multiple Data) yapısını taklit eder. Piksel verileri gruplar halinde (batch) işlenir ve tek bir güncelleme döngüsünde ekrana yansıtılır.



Kullanıcı Dostu Arayüz ve Etkileşim



Özellikler



- **Ana Menü:** Kullanıcının CPU veya GPU simülasyon modunu seçebileceği net butonlar.



- **Gösterge Paneli (Dashboard):** Simülasyon sırasında işlem tipi, ilerleme yüzdesi gibi anlık bilgileri gösteren bir panel.



- **Tam Etkileşim:** Kullanıcı istediği zaman simülasyonu durdurup 'Geri' butonu ile menüye dönebilir veya 'ESC' tuşu ile programdan çıkabilir.



- **Platform Bağımsızlığı:** Windows, macOS ve Linux sistemlerde ek sürücü gerektirmeden çalışacak şekilde tasarlanmıştır.

Sonuçların GörSEL Kanıtı: Erişim Desenleri

CPU - 'Satır Satır' Oluşum



GPU - 'Mozaik' Belirme



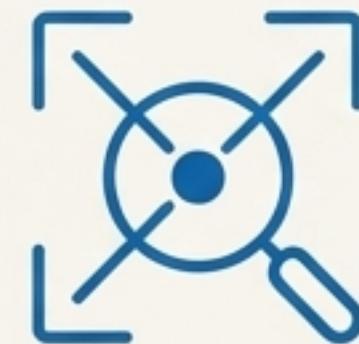
Sıralı Erişim (Sequential Access): Görüntü, tahmin edilebilir ve düzenli bir şekilde, yukarıdan aşağıya doğru tamamlanır.

Rastgele Erişim (Random Access): Görüntü, bütünsel olarak ve her bölgeden eşzamanlı piksellerle belirginleşir.

Temel Gözlemler ve Sonuçlar

Simülasyonlar, iki mimari arasındaki temel davranışsal farkları net bir şekilde ortaya koymuştur:

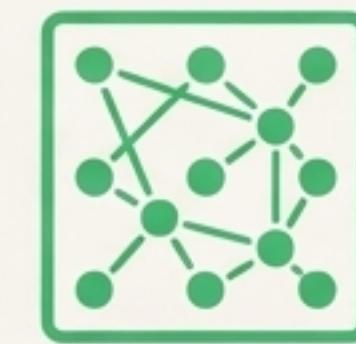
CPU Davranışı



Kavram: Tek Odak (Single Focus)

CPU, tek bir iş parçacığı ile tek bir noktaya odaklanır. Veri setini baştan sona metodik bir şekilde tarar. Bu, **düşük gecikme (low latency)** gerektiren görevler için idealdir.

GPU Davranışı



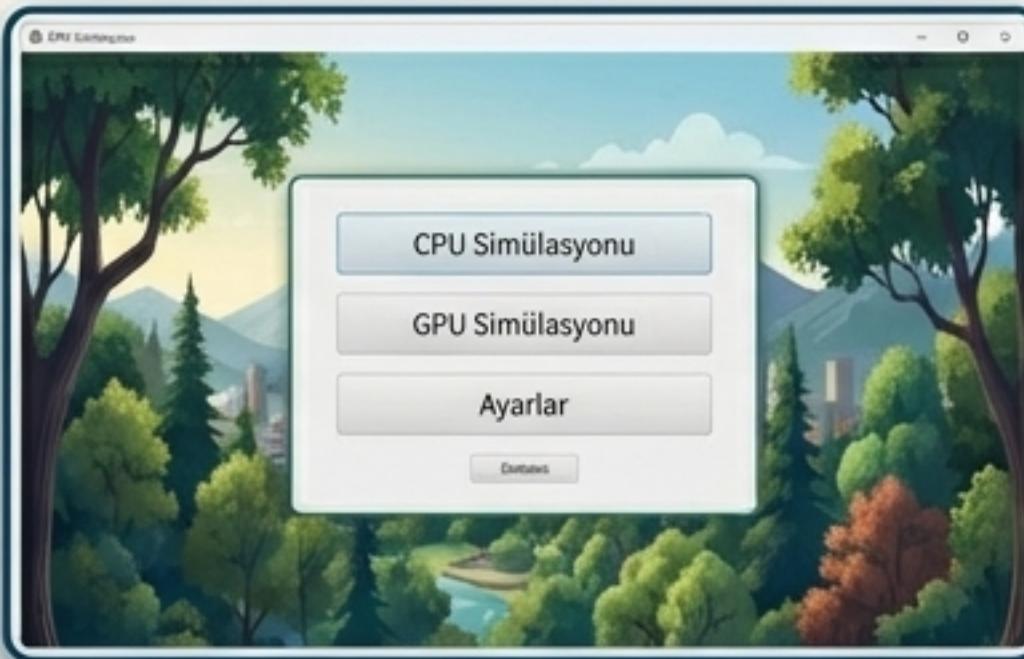
Kavram: Dağıtık Odak (Distributed Focus)

GPU, binlerce çekirdeği ile problemin tamamına dağıtık bir şekilde saldırır. Bu, **yüksek işlem hacmi (high throughput)** gerektiren görevler için idealdir.

Soyuttan Somuta: Donanım Kavramlarını Anlaşıılır Kılmak

Bu proje, CPU (Seri) ve GPU (Paralel) işlem nımarileri arasındaki temel farkları, teorik bilgiden çıkarıp gözlemlenebilir ve etkileşimli bir deneyime dönüştürmüştür.

Başarı Vurgusu: Geliştirilen simülasyon aracı, 'Latency vs. Throughput' ve 'Serial vs. Parallel' gibi karmaşık bilgisayar mimarisi kavramlarının eğitim ortamında daha kalıcı ve sezgisel bir şekilde anlaşılmasını sağlayan etkili bir eğitim materyali olduğunu kanıtlamıştır.



Projeyi Keşfedin

Projenin tüm kaynak kodlarına, detaylı dökümantasyonuna ve çalıştırılabilir dosyalarına aşağıdaki GitHub deposu üzerinden erişebilirsiniz.



github.com/agkocaman/kostu-gpu-cpu

Katkıda Bulunanlar

Ahmet Gökhan Kocaman
Bahadır Bekeç

Kaynakça

1. Hennessy, J. L., & Patterson, D. A. (2017). *Computer Architecture: A Quantitative Approach*.
2. NVIDIA. *CUDA C++ Programming Guide*.
3. Python Software Foundation. (2024). *Python Language Reference*.
4. Pygame Community. (2024). *Pygame Documentation*.

Bu sunum Kocaeli Sağlık ve Teknoloji Üniversitesi, Bilişim Sistemleri Mühendisliği Yüksek Lisans Programı, Bilgisayar Mimarisi ve Organizasyonu dersi kapsamında hazırlanmıştır.