THE ULTIMATE GUIDE TO BATTERY OPTIMISATION

🍏 ALEKSANDRA KOMAGORKINA | @AKOMAGORKINA

WHAT THE TALK IS ABOUT?

- 🔋 WHY SHOULD WE CARE ABOUT BATTERIES?
- ⚙️ THE MOST BATTERY CONSUMING ELEMENTS
- 📈 WAYS WE CAN MAKE OUR APPS MORE ENERGY EFFICIENT:
  - WHAT TO LOOK FOR 👀
  - TOOLS WE HAVE AND HOW TO USE THEM 🛠️
  - COMMON AND NOT SO PROBLEMS WITH EXAMPLES 💣

WHY SHOULD WE CARE?

- 📈 EFFICIENCY
- 🚀 PERFOMANCE
- 🌍 GLOBAL IMPACT

WHY SHOULD YOUR MANAGER CARE?

- 🚀 OPTIMIZING PERFOMANCE OFTEN LEADS TO BATTERY USAGE OPTIMIZATION
- 🥇 GET ADVANTAGE TO CONQUER COMPETITOR'S USERS
- ⏳ LESS BATTERY DRAINAGE = LONGER SESSIONS
- 🥚 LOTS OF LONG-TERM BENEFITS

WHAT DOES DRAIN ENERGY ON THE IPHONE?

DON'T WORRY TOO MUCH

IOS ENERGY-SAVING TECHNOLOGIES:

- INTEGRATED HARDWARE & SOFTWARE
- INTELLIGENT APP MANAGEMENT
- NETWORK OPERATION DEFFERAL
- TASK PRIORITIZATION
- DEVELOPER TOOLS

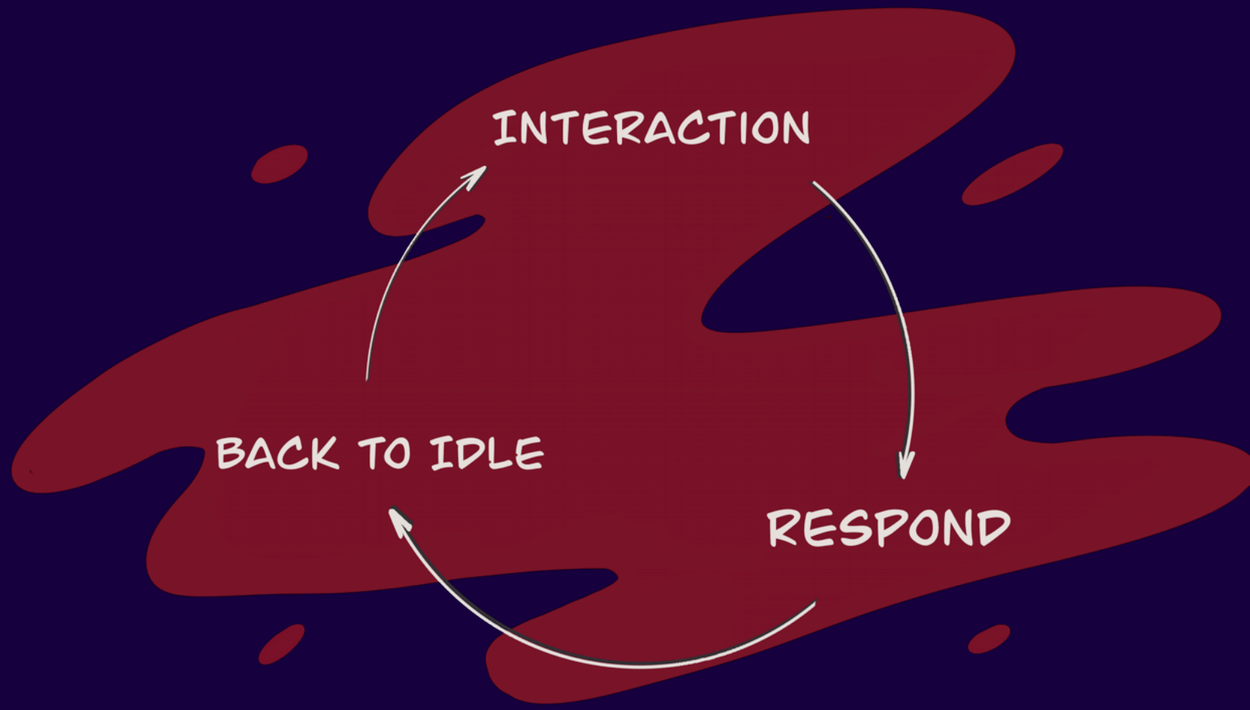
WHAT CAN WE ACHIEVE?

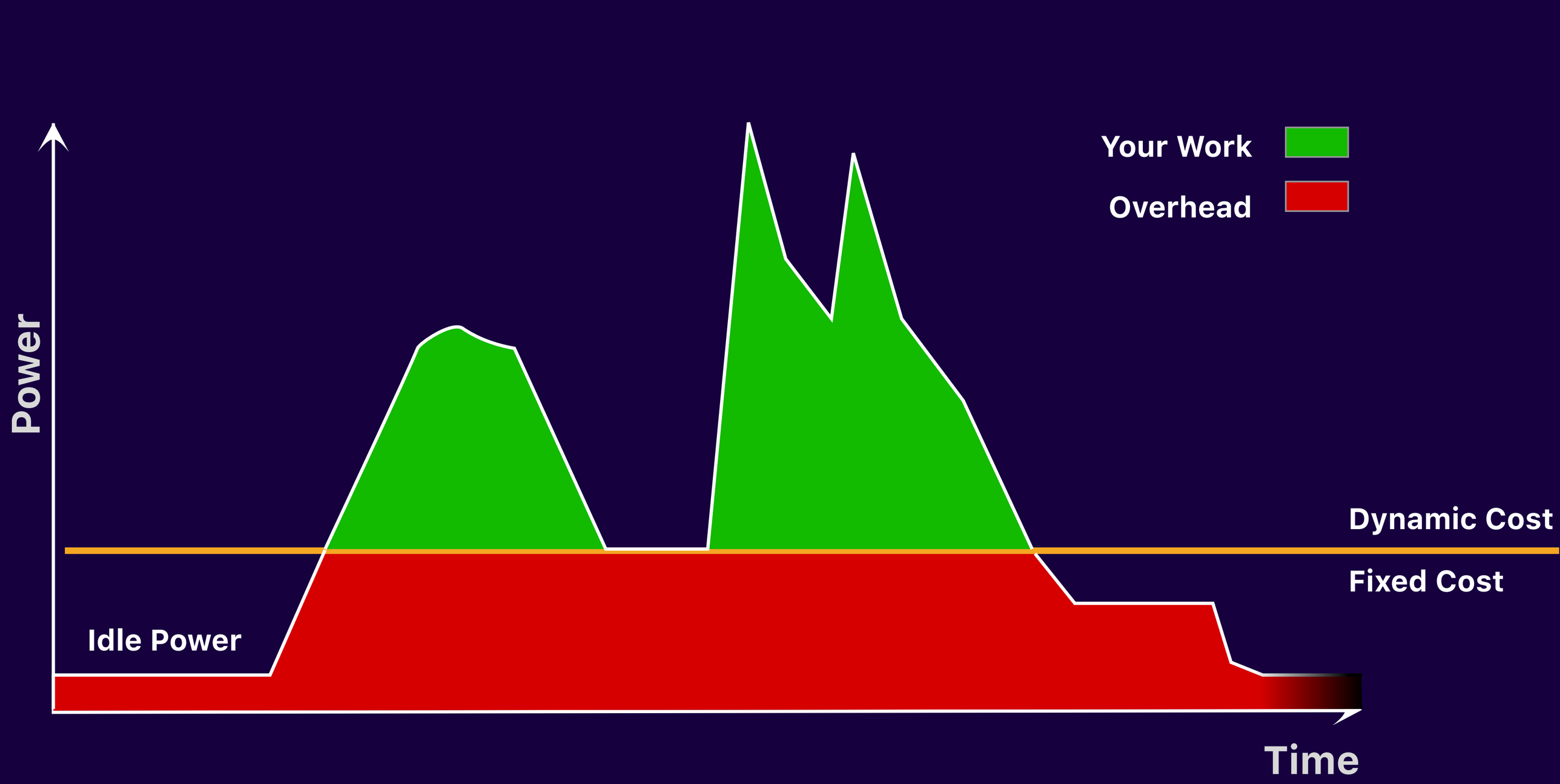
- RESPONSIVENESS 🚀
- GREAT BATTERY LIFE ON THE DEVICE 🔋
- COOL DEVICE 🌡️


STRATEGY

USER-DRIVEN WORK

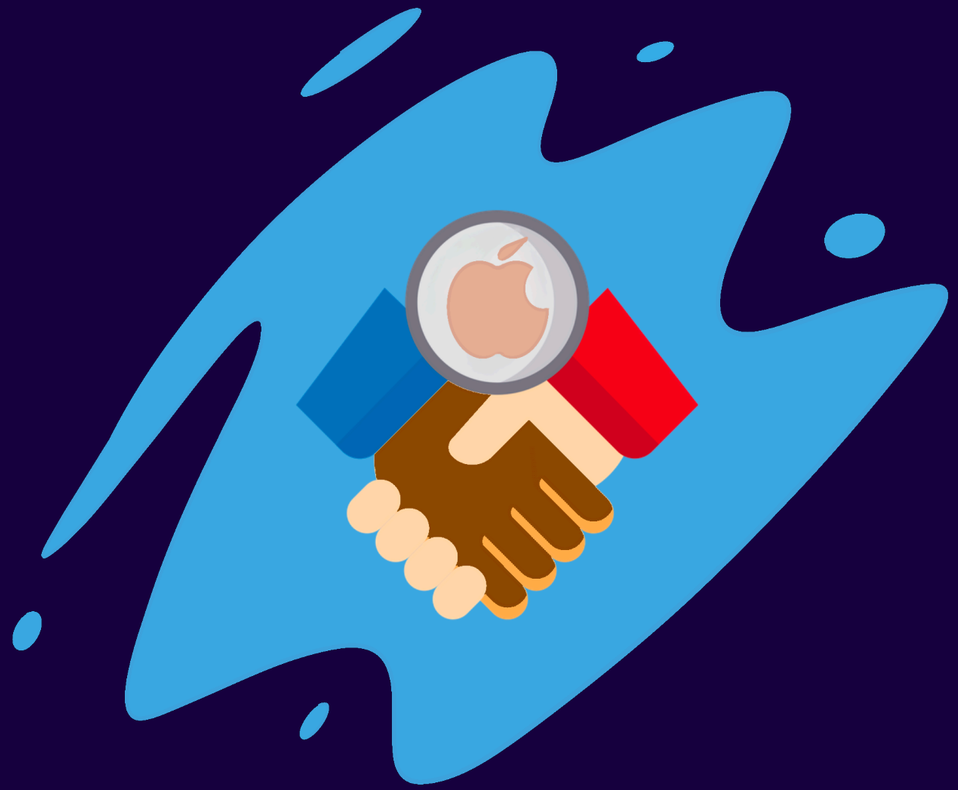
USER-DRIVEN WORK




Your Work
Overhead
Power
Dynamic Cost
Fixed Cost
Idle Power
Time

PRIORITIZING WORK

TRUST THE SYSTEM



TOOLS

-  ENERGY DEBUGGER
-  XCODE INSTRUMENTS
-  XCODE METRICS

GENERAL BATTERY PERFOMANCE CONSIDERATIONS

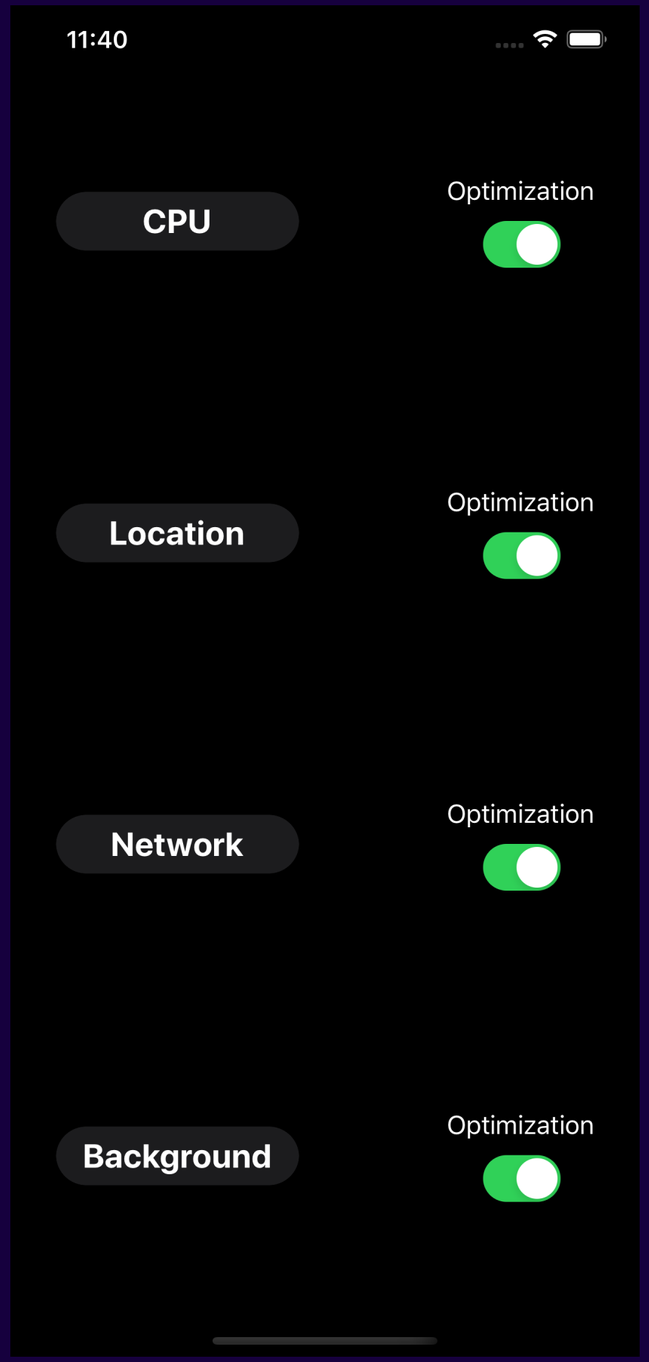
CPU

- MAJOR CONSUMER OF ENERGY
- WORKLOAD DEPENDANT
- WHAT TO DO? DO NOT WAKE WHEN POSSSIBLE

TIME PERFORMANCE

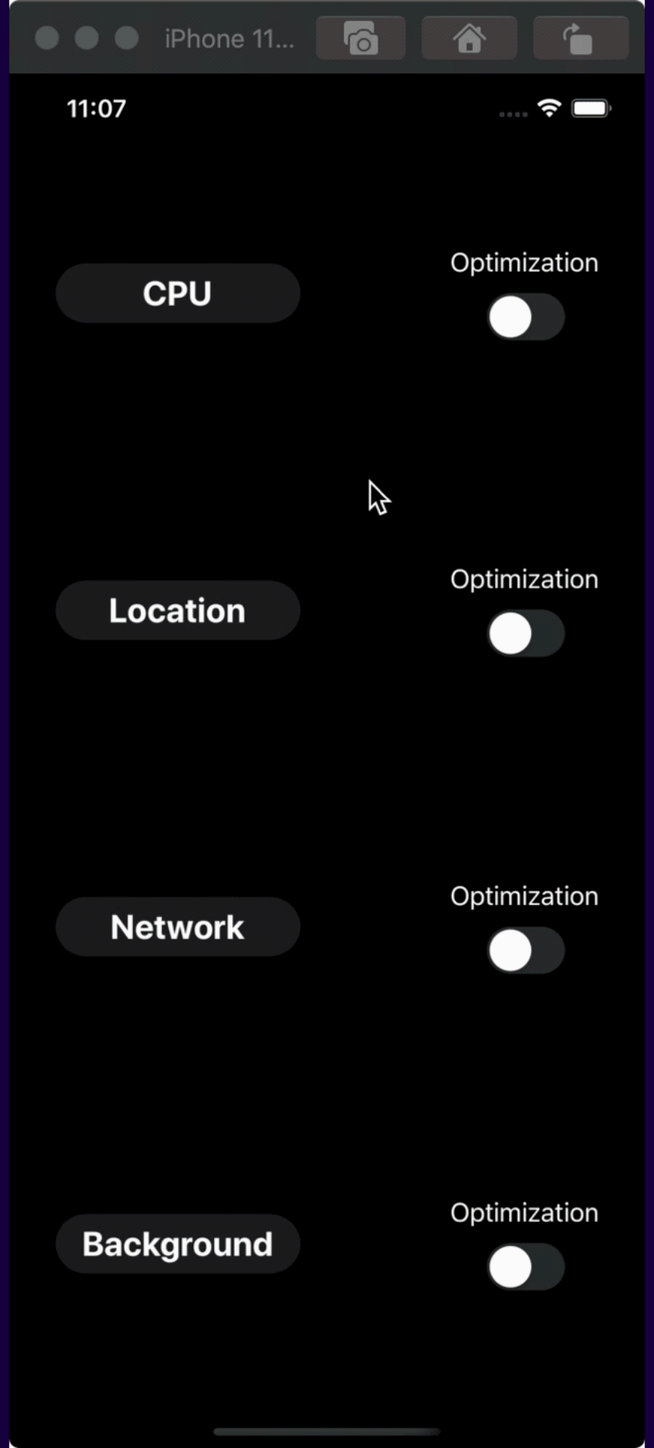
- FASTER = LESS ENERGY IMPACT. BUT DON'T FORGET TO PRIORITIZE.
- WHAT TO DO?
  - UNDERSTAND WHEN, HOW MUCH AND WHAT KIND OF WORK THE APP IS DOING?
  - USE TIME PROFILER TO OPTIMIZE

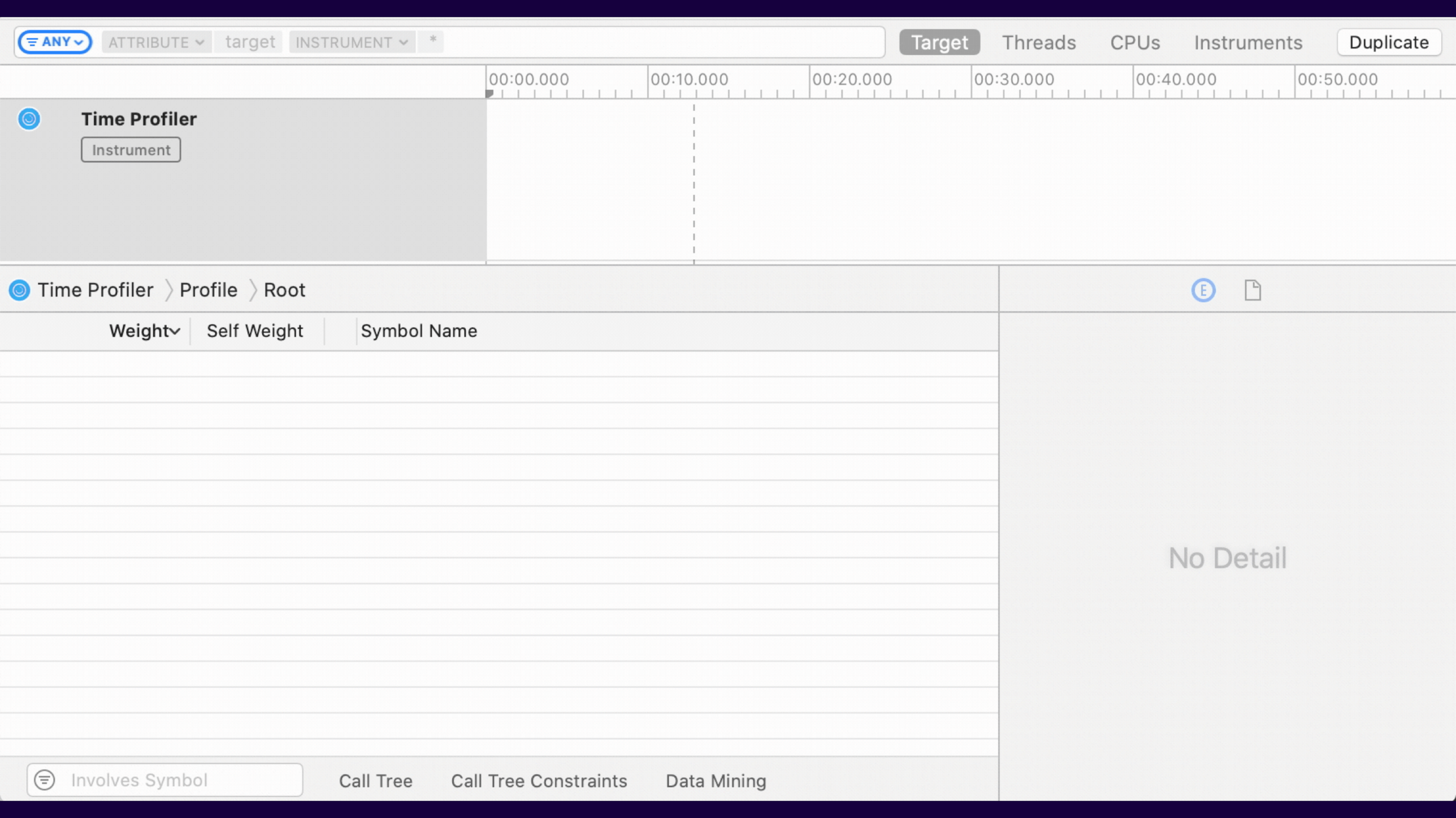
TIME PROFILER


11:40
CPU
Optimization
Location
Optimization
Network
Optimization
Background
Optimization

CPU

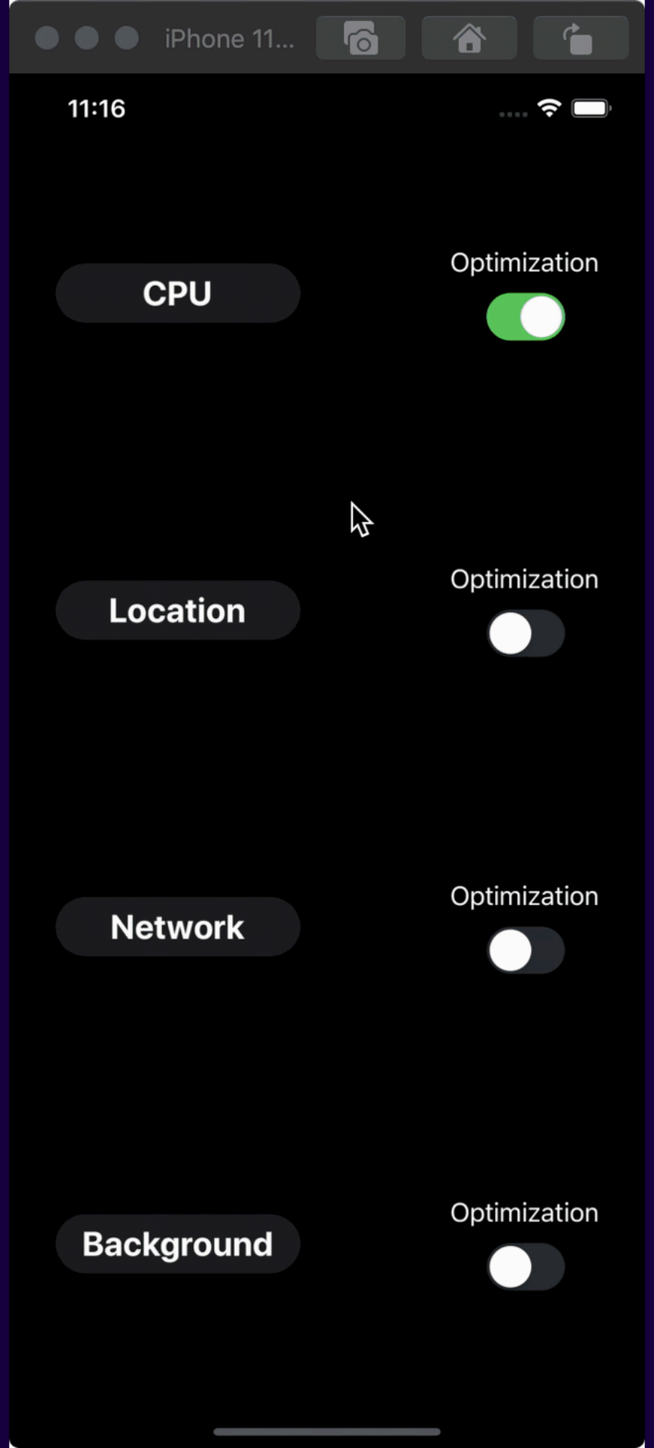
```
func image() -> UIImage? {
  let size = CGSize(width: 36, height: 36)
  UIGraphicsBeginImageContextWithOptions(size, false, 0)
  UIColor.systemBackground.set()
  let rect = CGRect(origin: .zero, size: size)
  UIRectFill(
    CGRect(
      origin: .zero, size: size
    )
  )
  (self as AnyObject).draw(
    in: rect,
    withAttributes: [.font: UIFont.systemFont(ofSize: 30)]
  )
  let image = UIGraphicsGetImageFromCurrentImageContext()
  UIGraphicsEndImageContext()
  cachedEmojis[self] = image
  return image
}

// ...

imageView.image = "🤦".image()
```

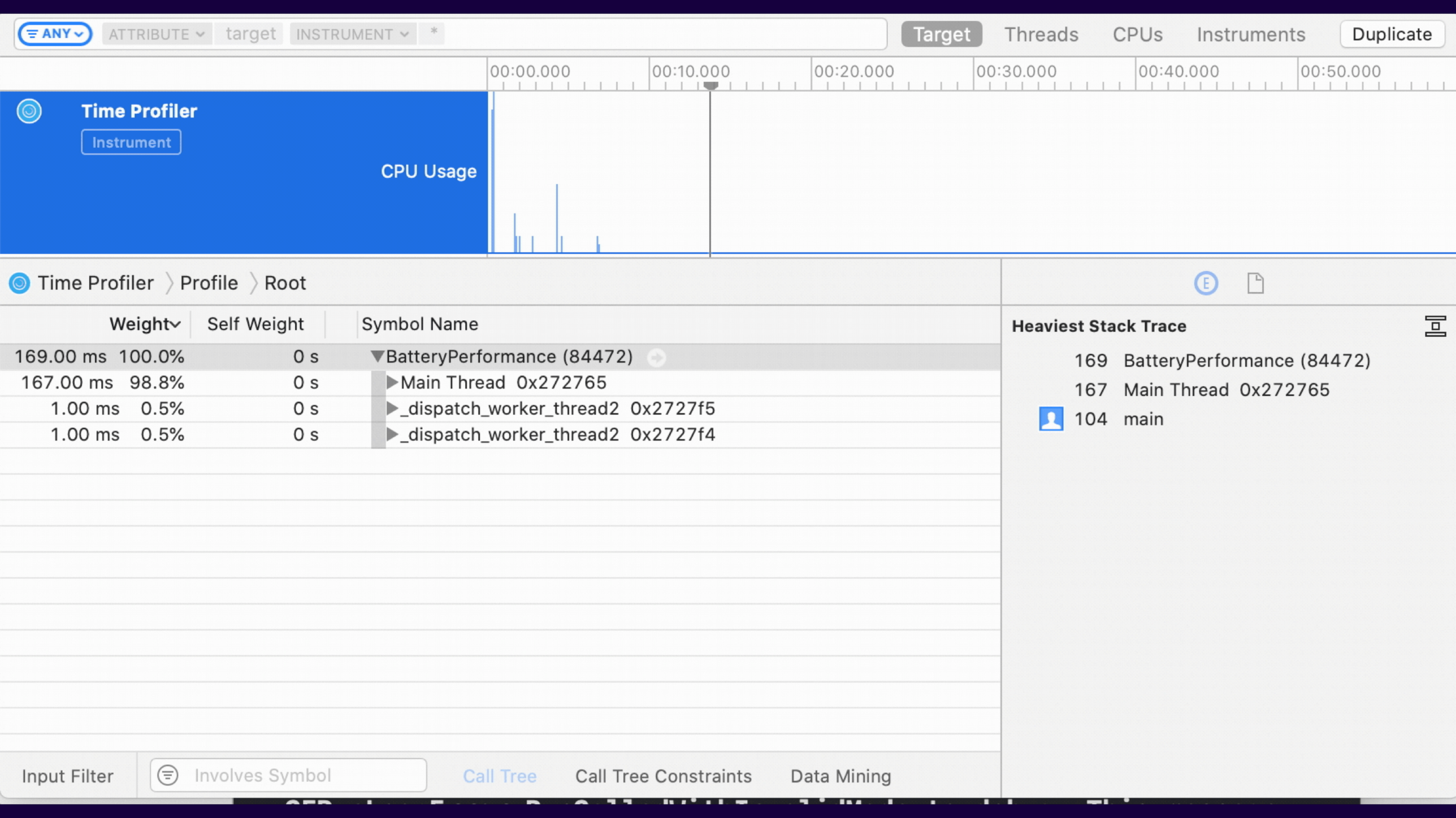


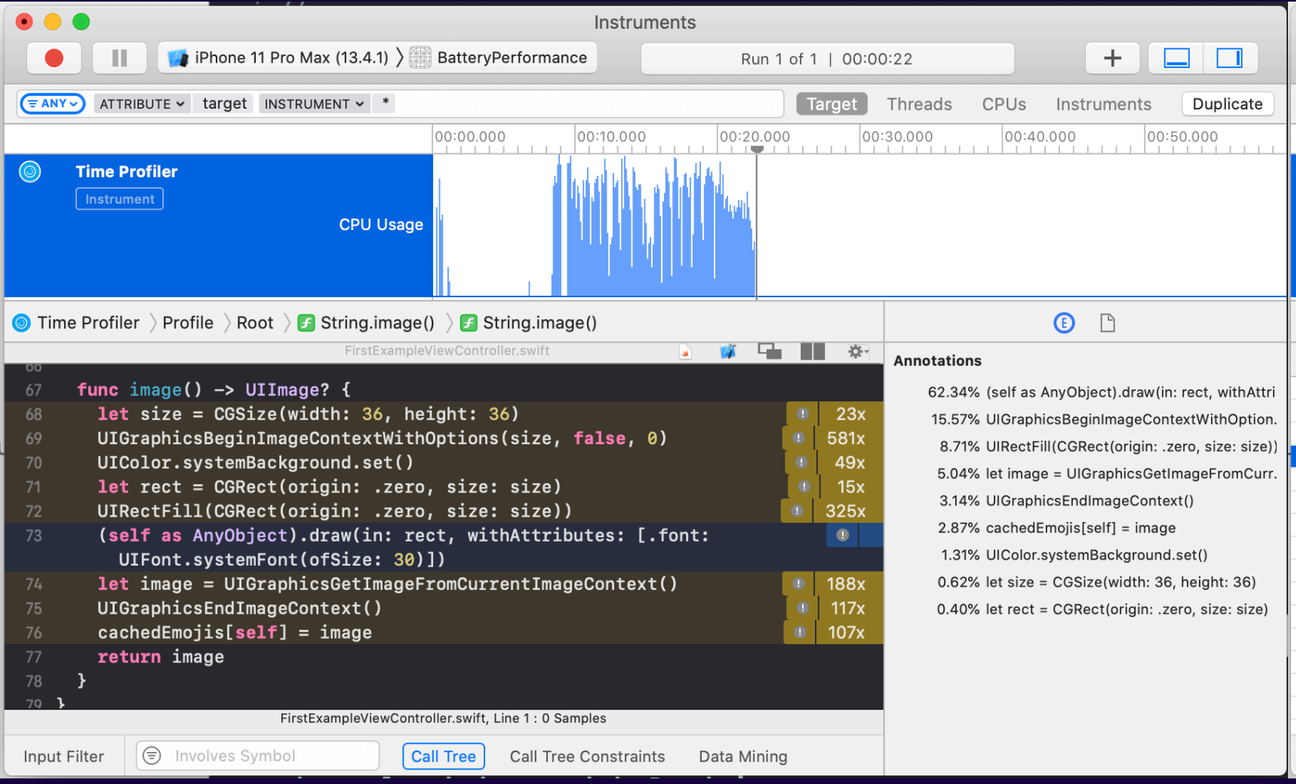

ANY
ATTRIBUTE
target
INSTRUMENT
*
Target
Threads
CPUs
Instruments
Duplicate
00:00.000
00:10.000
00:20.000
00:30.000
00:40.000
00:50.000
Time Profiler
Instrument
Time Profiler
Profile
Root
Weight
Self Weight
Symbol Name
No Detail
Involves Symbol
Call Tree
Call Tree Constraints
Data Mining

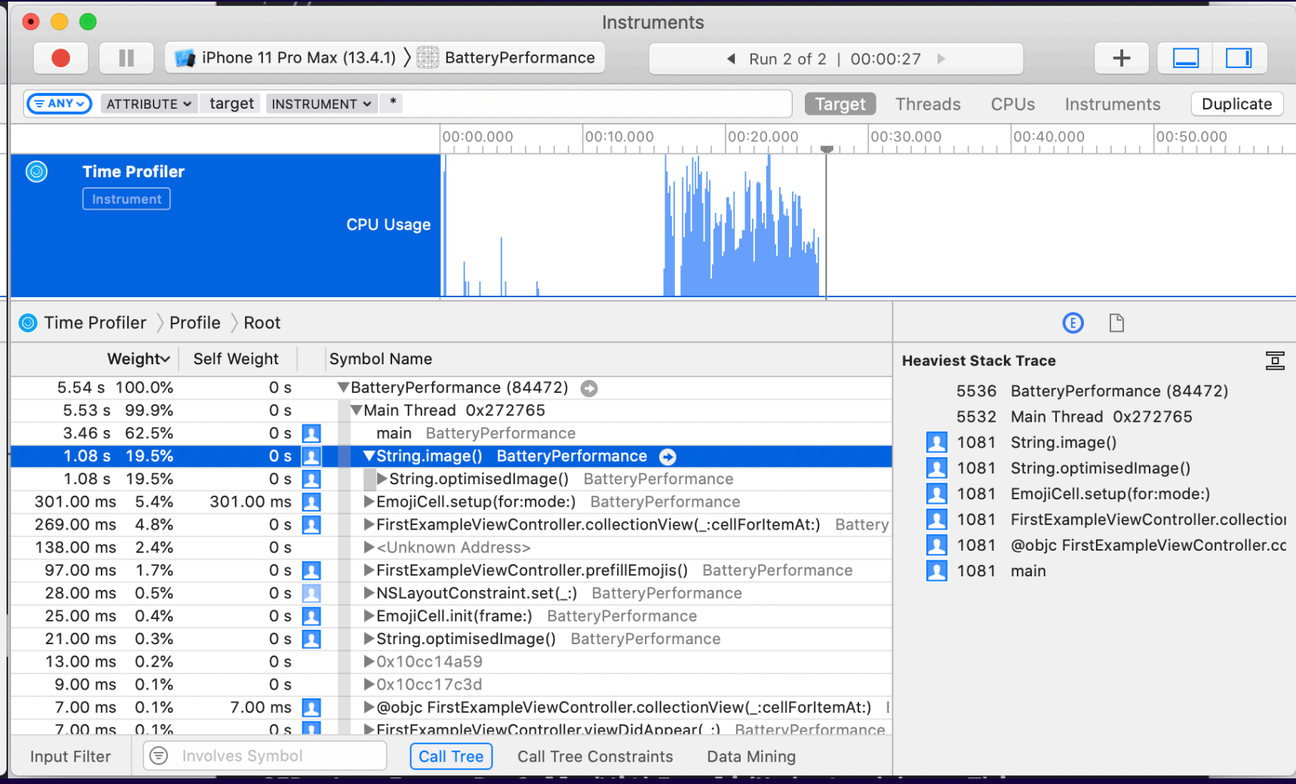
CPU

```
var cachedEmojis = [String: UIImage?]()

func optimisedImage() -> UIImage? {
    if cachedEmojis[self] != nil {
      return cachedEmojis[self] ?? nil
    } else {
      return image()
    }
  }

// ...

imageView.image = "🎉".optimisedImage()
```




ANY
ATTRIBUTE
target
INSTRUMENT
*
Target
Threads
CPUs
Instruments
Duplicate
00:00.000
00:10.000
00:20.000
00:30.000
00:40.000
00:50.000
Time Profiler
Instrument
CPU Usage
Time Profiler
Profile
Root
Weight
Self Weight
Symbol Name
169.00 ms 100.0% 0 s BatteryPerformance (84472)
167.00 ms 98.8% 0 s Main Thread 0x272765
1.00 ms 0.5% 0 s _dispatch_worker_thread2 0x2727f5
1.00 ms 0.5% 0 s _dispatch_worker_thread2 0x2727f4
Heaviest Stack Trace
169 BatteryPerformance (84472)
167 Main Thread 0x272765
104 main
Input Filter
Involves Symbol
Call Tree
Call Tree Constraints
Data Mining


Instruments
iPhone 11 Pro Max (13.4.1) 〉 BatteryPerformance
Run 1 of 1 | 00:00:22
ANY ATTRIBUTE target INSTRUMENT *
Target Threads CPUs Instruments Duplicate
00:00.000 00:10.000 00:20.000 00:30.000 00:40.000 00:50.000
Time Profiler
Instrument
CPU Usage
Time Profiler 〉 Profile 〉 Root 〉 String.image() 〉 String.image()
FirstExampleViewController.swift
func image() -> UIImage? {
let size = CGSize(width: 36, height: 36) 23x
UIGraphicsBeginImageContextWithOptions(size, false, 0) 581x
UIColor.systemBackground.set() 49x
let rect = CGRect(origin: .zero, size: size) 15x
UIRectFill(CGRect(origin: .zero, size: size)) 325x
(self as AnyObject).draw(in: rect, withAttributes: [.font:
UIFont.systemFont(ofSize: 30)])
let image = UIGraphicsGetImageFromCurrentImageContext() 188x
UIGraphicsEndImageContext() 117x
cachedEmojis[self] = image 107x
return image
}
FirstExampleViewController.swift, Line 1 : 0 Samples
Annotations
62.34% (self as AnyObject).draw(in: rect, withAttri
15.57% UIGraphicsBeginImageContextWithOption.
8.71% UIRectFill(CGRect(origin: .zero, size: size))
5.04% let image = UIGraphicsGetImageFromCurr.
3.14% UIGraphicsEndImageContext()
2.87% cachedEmojis[self] = image
1.31% UIColor.systemBackground.set()
0.62% let size = CGSize(width: 36, height: 36)
0.40% let rect = CGRect(origin: .zero, size: size)
Input Filter Involves Symbol Call Tree Call Tree Constraints Data Mining

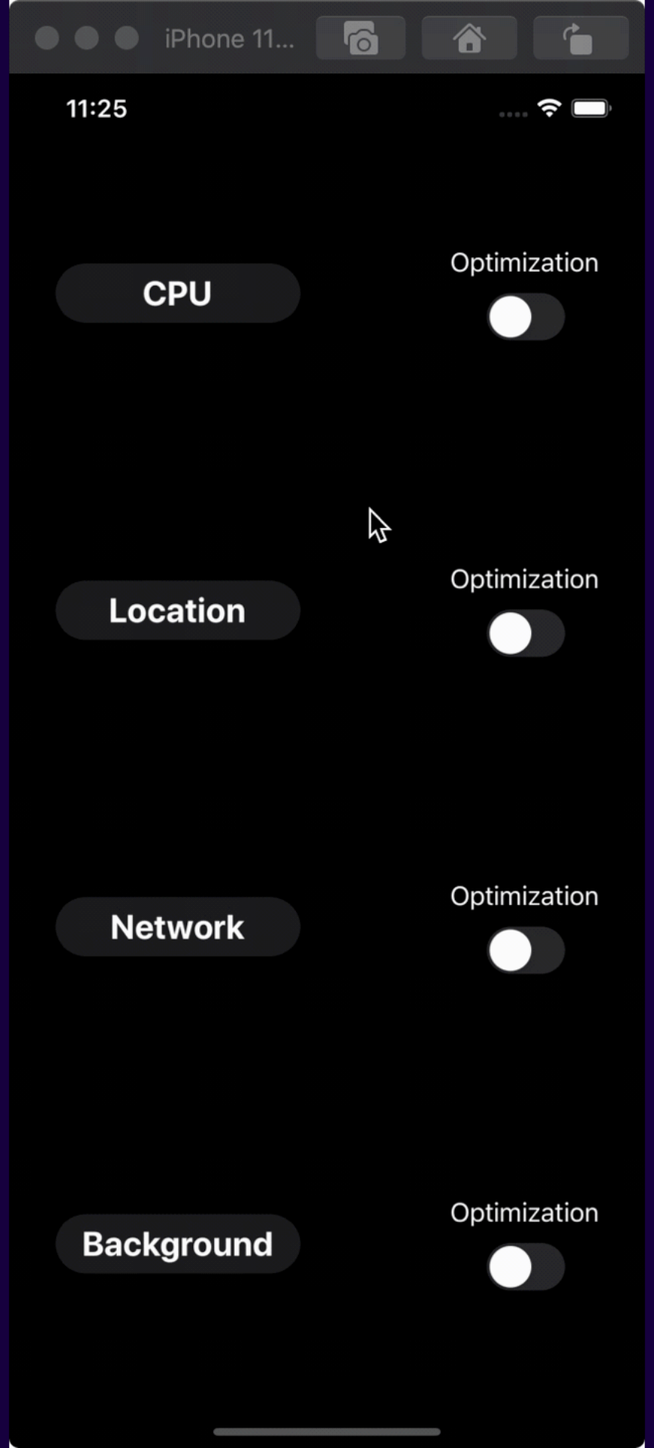

Instruments
iPhone 11 Pro Max (13.4.1) 〉 BatteryPerformance
Run 2 of 2 | 00:00:27
ANY ATTRIBUTE target INSTRUMENT *
Target Threads CPUs Instruments Duplicate
00:00.000 00:10.000 00:20.000 00:30.000 00:40.000 00:50.000
Time Profiler
Instrument
CPU Usage
Time Profiler 〉 Profile 〉 Root
Weight Self Weight Symbol Name
5.54 s 100.0% 0 s BatteryPerformance (84472)
5.53 s 99.9% 0 s Main Thread 0x272765
3.46 s 62.5% 0 s main BatteryPerformance
1.08 s 19.5% 0 s String.image() BatteryPerformance
1.08 s 19.5% 0 s String.optimisedImage() BatteryPerformance
301.00 ms 5.4% 301.00 ms EmojiCell.setup(for:mode:) BatteryPerformance
269.00 ms 4.8% 0 s FirstExampleViewController.collectionView(_:cellForItemAt:) Battery
138.00 ms 2.4% 0 s <Unknown Address>
97.00 ms 1.7% 0 s FirstExampleViewController.prefillEmojis() BatteryPerformance
28.00 ms 0.5% 0 s NSLayoutConstraint.set(_:) BatteryPerformance
25.00 ms 0.4% 0 s EmojiCell.init(frame:) BatteryPerformance
21.00 ms 0.3% 0 s String.optimisedImage() BatteryPerformance
13.00 ms 0.2% 0 s 0x10cc14a59
9.00 ms 0.1% 0 s 0x10cc17c3d
7.00 ms 0.1% 7.00 ms @objc FirstExampleViewController.collectionView(_:cellForItemAt:)
Heaviest Stack Trace
5536 BatteryPerformance (84472)
5532 Main Thread 0x272765
1081 String.image()
1081 String.optimisedImage()
1081 EmojiCell.setup(for:mode:)
1081 FirstExampleViewController.collectio
1081 @objc FirstExampleViewController.cc
1081 main
Input Filter Involves Symbol Call Tree Call Tree Constraints Data Mining

LOCATION

- USUALLY NOT HIGH IN POWER, BUT TIME-CONSUMING
- ACCURACY AND FREQUANCY DEPENDANT
- EFFICIENCY = ACCURACY
- WHAT TO DO? WORK WHEN NEEDED AND STOP WHEN DONE

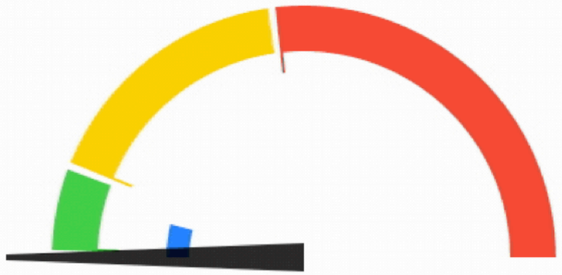
LOCATION

```
manager.desiredAccuracy = kCLLocationAccuracyBest
manager.startUpdatingLocation()
```



Energy

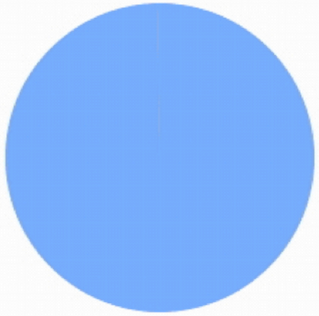
Average Energy Impact



Zero

Energy Impact

Average Component Utilization



- **Overhead** 0%
- **Location** 0%
- **CPU** 0%
- **GPU** 0%
- **Network** 0%
- **Display** 99.8%

Energy Impact

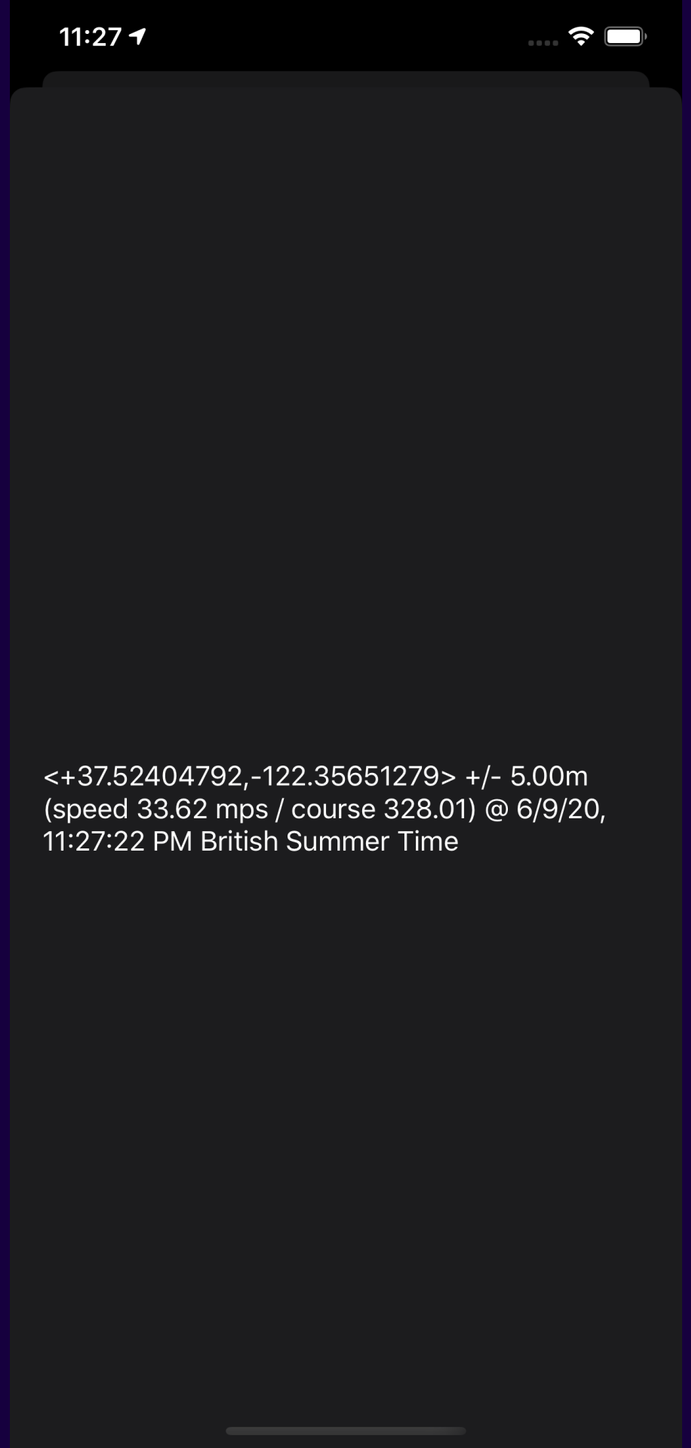
Component Utilization

Overhead **CPU** **Network** **Location** **GPU** **Display** **Background**

Application State

LOCATION

```
manager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
manager.requestLocation()
```



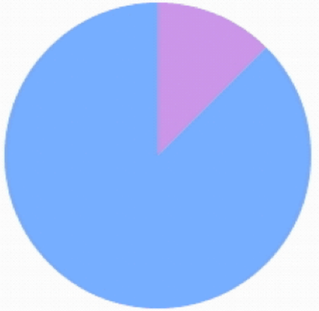
Energy

Average Energy Impact

Zero

Energy Impact

Average Component Utilization

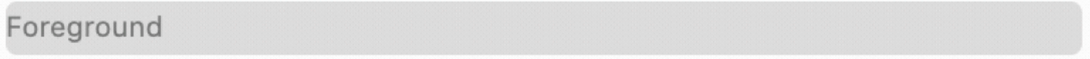


Overhead 0%	**Location** 0%
CPU 12.4%	**GPU** 0%
Network 0%	**Display** 87.6%

Energy Impact

Component Utilization

Application State

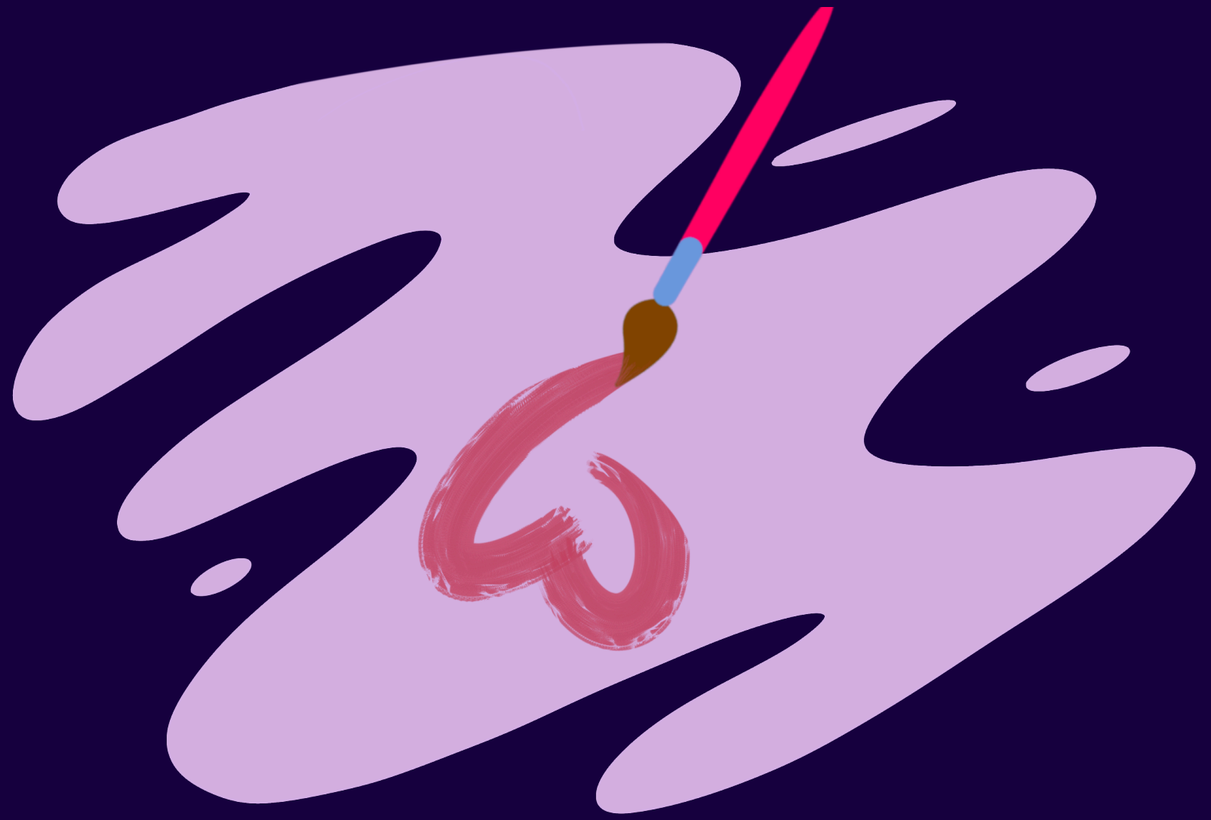


NETWORKING

- TRAFFIC DEPENDANT
- UNDERSTAND WHAT YOU NEED TO DO
- COALESE TRANSFERS => REDUCE FIXED COAST

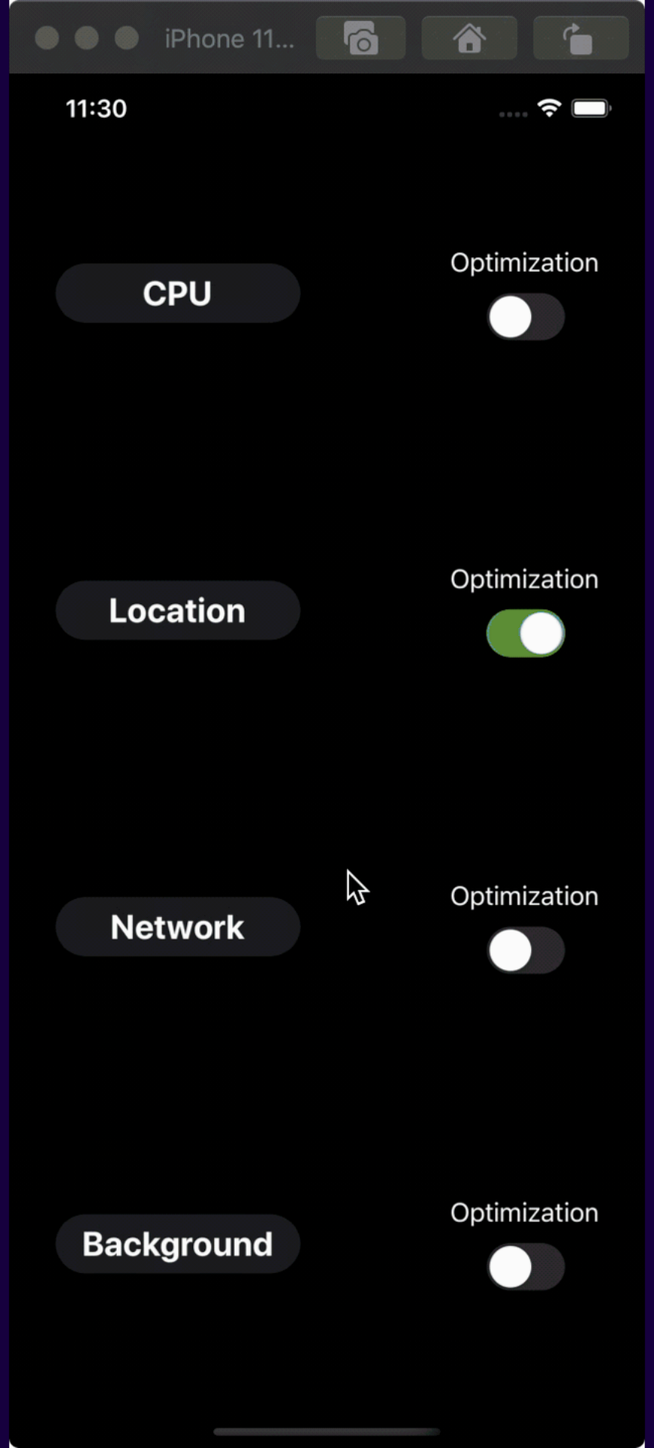
GRAPHICS

- ANIMATIONS AND UI-DEPENDANT
- COMPLEXITY DEPENDANT: MORE RENDERING => MORE ENERGY
- REDUCE COMPLEXITY WHEN POSSIBLE



NETWORK

```
imageRequest = ImageRequest(
        path: "/2000/3000/", // Dimensions
        host: "api.aleksandra.tech",
        queryItems: nil
      )
// ...

DataLoader.shared.load(imageRequest) { [weak self] result in
      switch result {
      case .success(let data):
        let image = UIImage(data: data)
        DispatchQueue.main.async {
          self?.imageView.image = image
        }
      case .failure:
        break
      }
    }
```



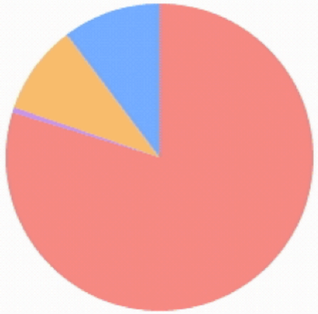
Energy

Average Energy Impact

Low

Energy Impact

Average Component Utilization



Overhead 79.8%	**Location** 0%
CPU 0.4%	**GPU** 0%
Network 9.4%	**Display** 10.4%

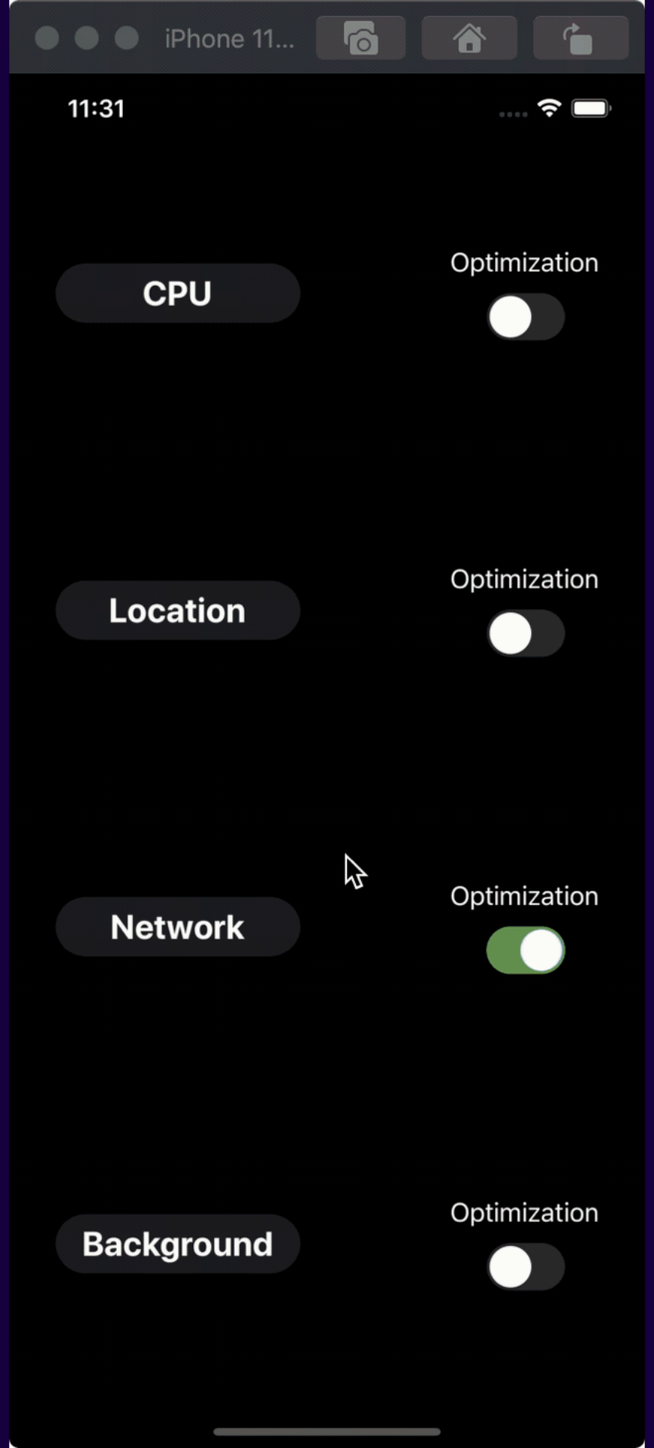
Energy Impact

Component Utilization

Application State

NETWORK

```
imageRequest = ImageRequest(
        path: "/160/150/", // Dimensions
        host: "api.aleksandra.tech",
        queryItems: nil
      )
// ...

DataLoader.shared.load(imageRequest) { [weak self] result in
      switch result {
      case .success(let data):
        let image = UIImage(data: data)
        DispatchQueue.main.async {
          self?.imageView.image = image
        }
      case .failure:
        break
      }
    }
```



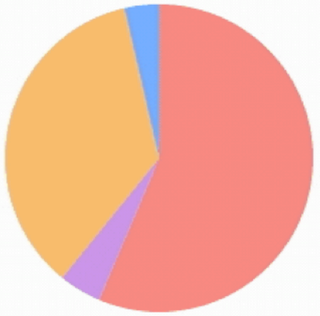
Energy

Average Energy Impact

Low

Energy Impact

Average Component Utilization



- **Overhead** 56.4%
- **Location** 0%
- **CPU** 4.3%
- **GPU** 0%
- **Network** 35.5%
- **Display** 3.7%

Energy Impact

Component Utilization

Overhead **CPU** **Network** **Location** **GPU** **Display** **Background**

Application State

BACKGROUND

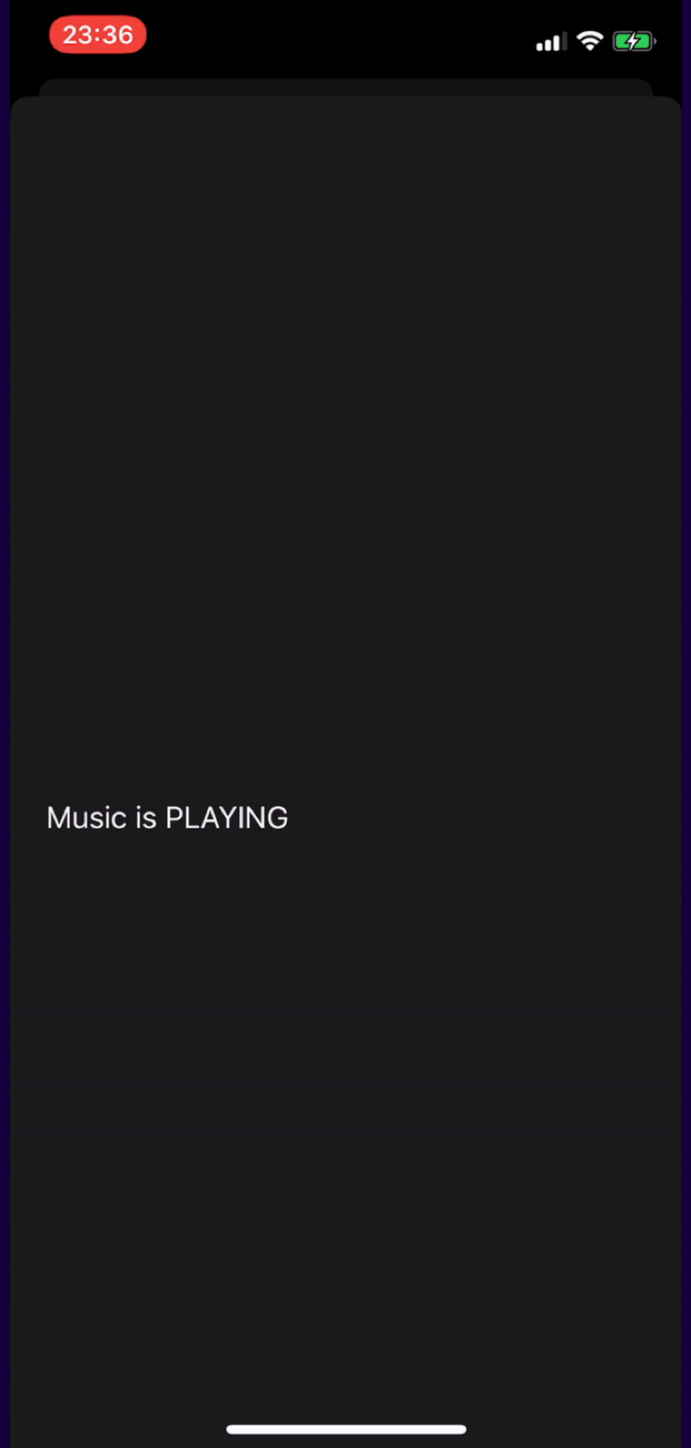

MUSIC
CONNECTING TO HOTSPOTS
SIRI INTEGRATION
NEW SERVER DATA
NAVIGATION
BLUETOOTH
NOT RUNNING
FOREGROUND
BACKGROUND
SUSPENDED
DOWNLOADING AND UPLOADING FILES
FINISH FOREGROUND WORK
WATCH APP
MAINTENANCE
CALLS
PERIODING CONTENT UPDATES

BACKGROUND

- FAST AND RARE WORK => LESS ENERGY IMPACT
- DON'T DO ANYTHING IF YOU DON'T **NEED** TO
- LEARN HOW TO COMMUNICATE WITH SYSTEM TO GET YOUR STUFF DONE

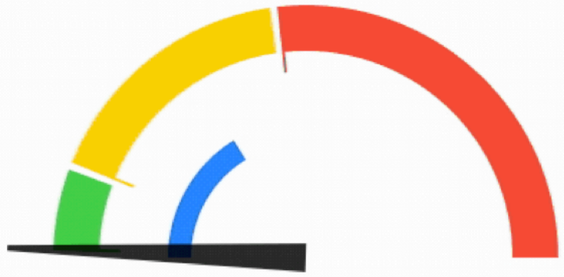
BACKGROUND

```
private func playSound() {
    let url = Bundle.main.url(forResource: "example", withExtension: "mp3")!

    do {
      player = try AVAudioPlayer(contentsOf: url)
      guard let player = player else { return }
      player.numberOfLoops = -1
      player.prepareToPlay()
      player.play()
      label.text = "Music is PLAYING"
    } catch let error as NSError {
      label.text = "Music is NOT PLAYING: \(error.localizedDescription)"
    }
  }
```



Energy

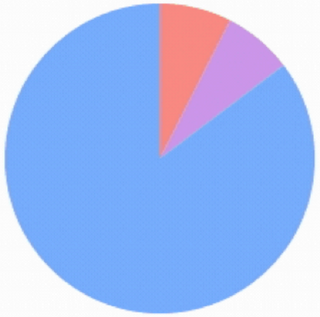
Average Energy Impact



Low

Energy Impact

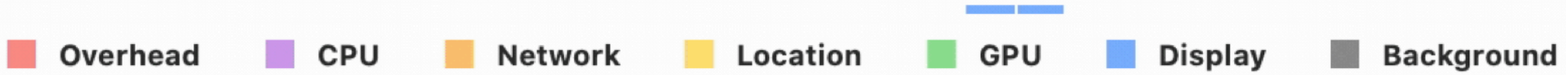
Average Component Utilization



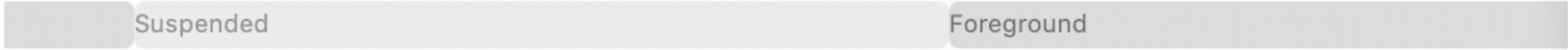
Overhead 7.4%	**Location** 0%
CPU 7.3%	**GPU** 0%
Network 0%	**Display** 100%

Energy Impact

Component Utilization

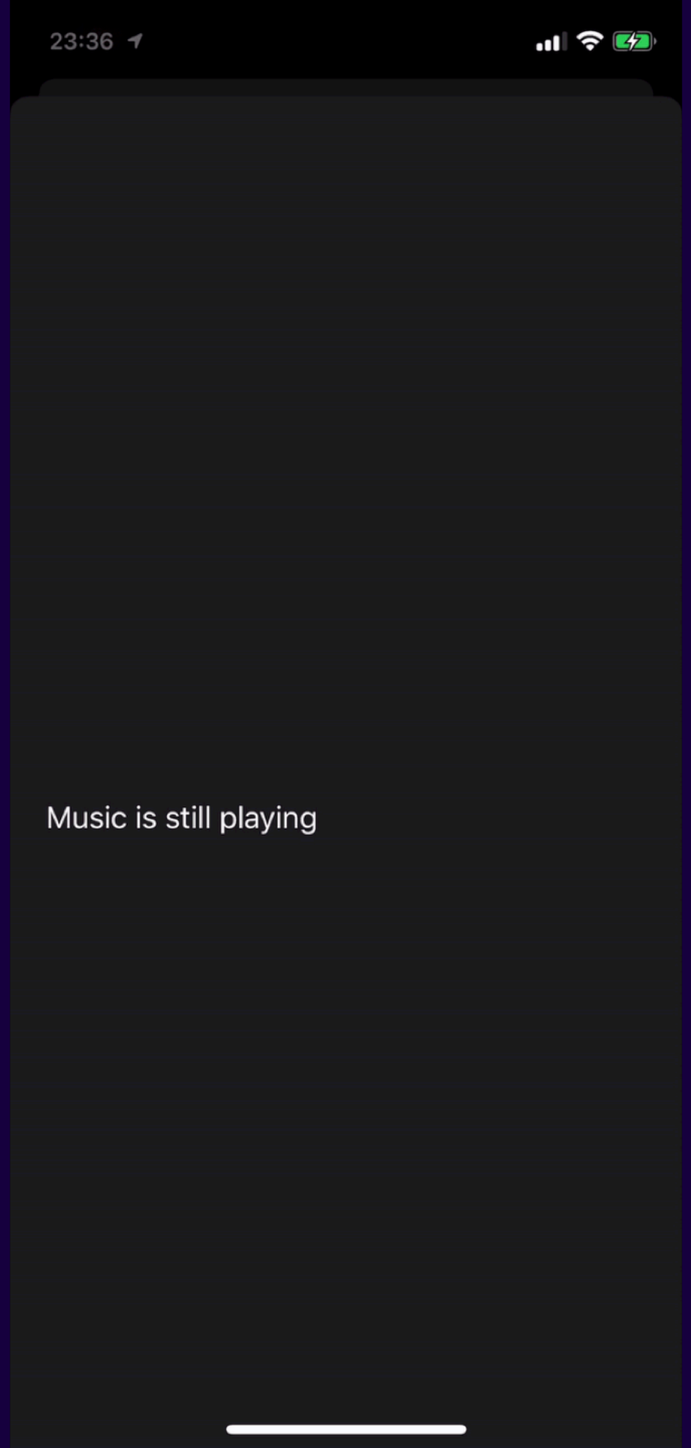


Application State



BACKGROUND

```
var outputValueObserver: NSKeyValueObservation?

outputValueObserver = AVAudioSession
  .sharedInstance()
  .observe( \.outputVolume ) {
  [weak self] (av, change) in
        guard let self = self else { return }
        if av.outputVolume <= 0.01 {
          self.player?.stop() // Player stopped
        }
      }

// ...

playSound()
```



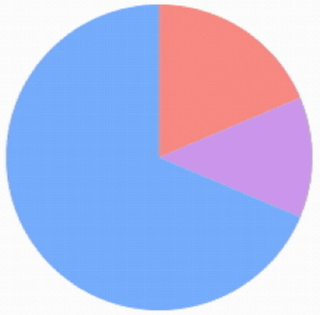
Energy

Average Energy Impact

Low

Energy Impact

Average Component Utilization



- **Overhead** 18.7%
- **Location** 0%
- **CPU** 12.8%
- **GPU** 0%
- **Network** 0%
- **Display** 105.6%

Energy Impact

Component Utilization

Overhead **CPU** **Network** **Location** **GPU** **Display** **Background**

Application State

COMPLETE BACKGROUND TASKS

BACKGROUND TASK COMPLETION

- USER EXPECTS IMMEDIATE COMPLETION
- PROTECT COMPLETION

=> GIVE THE APP ADDITIONAL TIME TO RUN IN THE BACKGROND BEFORE BEING SUSPENDED

BACKGROUND TASK COMPLETION

```
UIApplication.beginBackgroundTask(expirationHandler:)

ProcessInfo.performExpiringActivity(withReason:using:)
```

BACKGROUND TASK COMPLETION

```
// Guarding Important Tasks While App is Still in the Foreground
func send(_ message: Message) {
  let sendOperation = SendOperation(message: message)
  var identifier: UIBackgroundTaskIdentifier!
  identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
     sendOperation.cancel()
     postUserNotification("Message not sent, please resend")
     // Background task will be ended in the operation's completion block below
  })
  sendOperation.completionBlock = {
     UIApplication.shared.endBackgroundTask(identifier)
  }
  operationQueue.addOperation(sendOperation)
}
```

DEFER THE DOWNLOAD UNTIL THE BETTER TIME

DISCRETIONARY BACKGROUND URL SESSION

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

DISCRETIONARY BACKGROUND URL SESSION

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

METRICS

- **XCTEST METRICS**
  PERFORMANCE OF MEASURE BLOCKS
- **XCODE METRICS ORGANIZER**
  LAUNCH TIME. HANG RATE. WRITES TO STORAGE. MEMORY USE. AND ENERGY USE.
- **METRICKIT**
  THESE METRICS ARE IN THE FORM OF HISTOGRAMS THAT RECORD THE FREQUENCY OF OBSERVED VALUES OVER A DAY.

COLLECTING METRICS USING XCTEST

```
// This test measures Launch Time
func testAppLaunchTime() {
  measure(metrics: [XCTOSSignpostMetric.applicationLaunch]) {
  XCUIApplication().launch()
 }
}
```

ADOPTING METRICKIT TO RECEIVE METRICS

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

WHAT ELSE?

TIPS FOR SUCCESS:

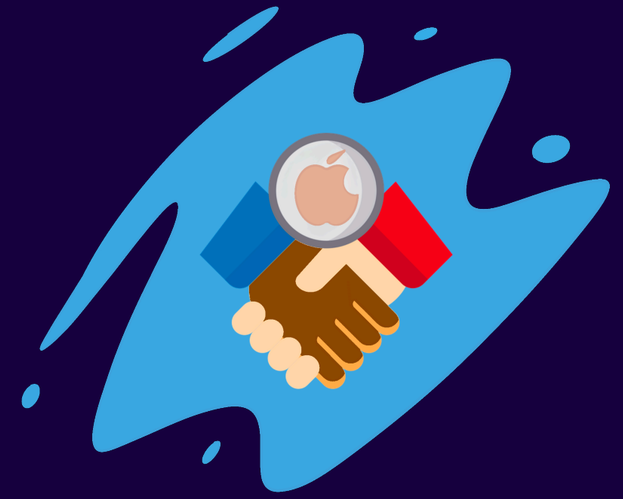
- MEASURE WHENEVER YOU CAN. MAKE IT YOUR HABIT.
- THE SOONER YOU START THE EASIER IT WILL BE.

TO SUM UP:

PRIORITIZE WORK

MAKE IT USER-DRIVEN

TRUST THE SYSTEM



TO SUM UP:

- OPTIMAZED BATTERY USAGE OF YOUR APP IS IMPORTANT FOR BETTER EXPERIENCES OF YOUR USERS AND THE SUCCESS OF YOUR APP
- THERE ARE VARIOUS WAYS TO MINIMAZE BATTERY USAGE: GO THROUGH EACH RESOURSE WHICH YOU USE ON THE DEVICE AND OPTIMISE EACH ELEMENT
- TEST AS OFTEN AS POSSIBLE TO MAKE YOUR BATTERY OPTIMIZATION PAINLESS AND EFFECTIVE.

"THERE IS NO POWER FOR CHANGE GREATER THAN A COMMUNITY DISCOVERING WHAT IT CARES ABOUT."

– MARGARET WHEATLEY

FURTHER LINKS & READING

THANK YOU!

@AKOMAGORKINA | ALEKSANDRA.TECH