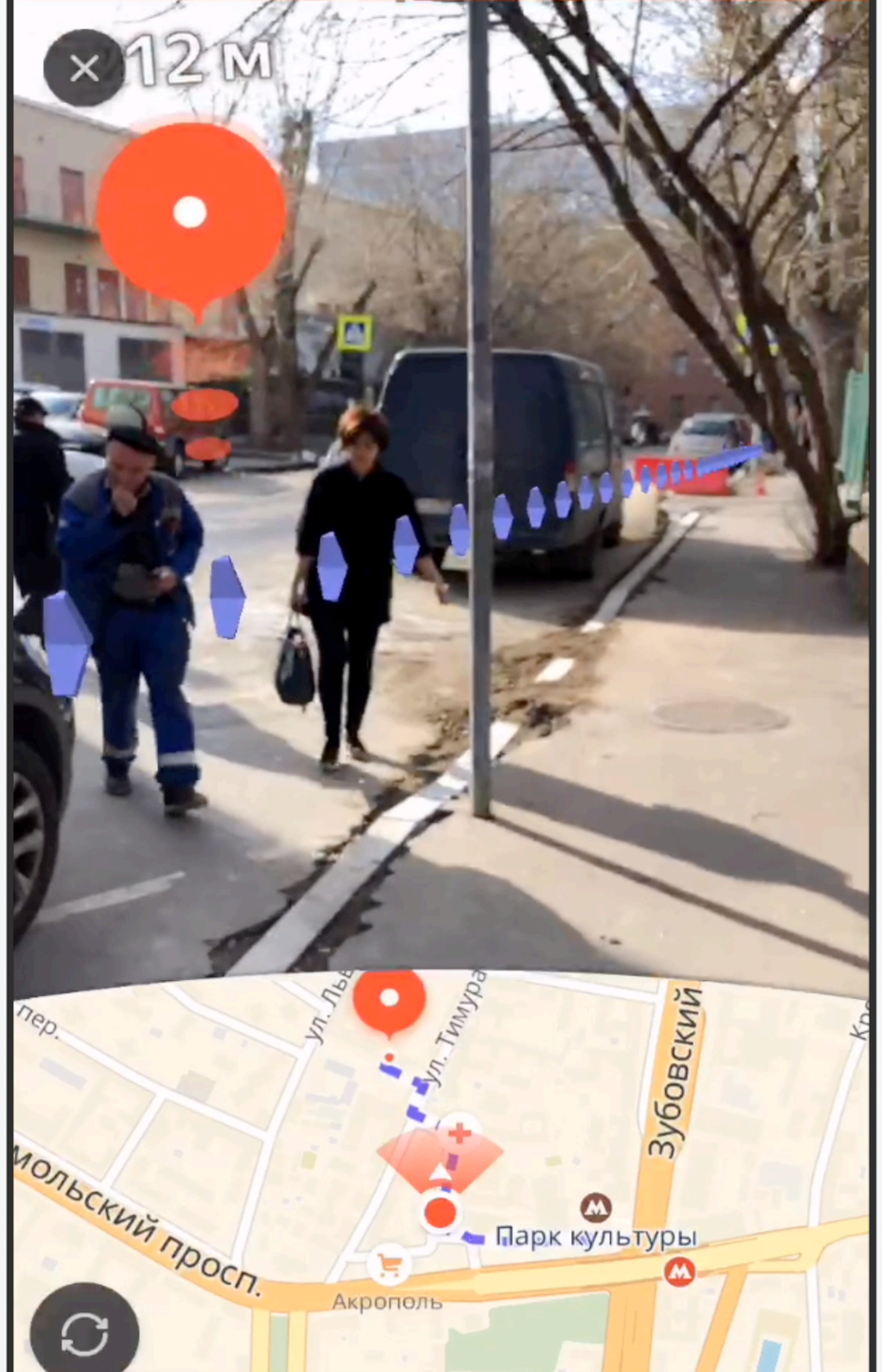
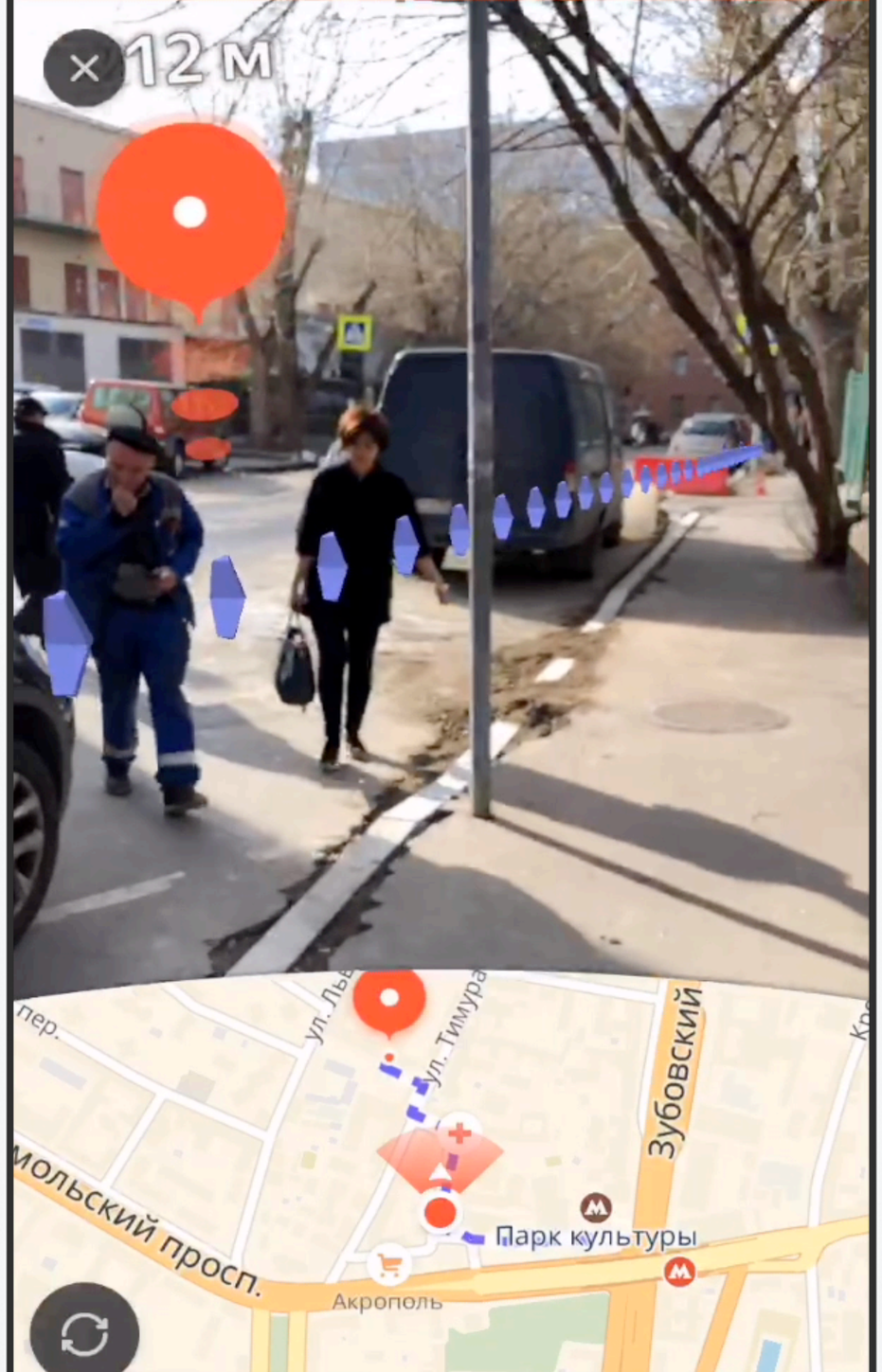
Яндекс

Яндекс Карты

Дополненная реальность в Яндекс.Картах

Дмитрий Тримонов, iOS-разработчик


12 м
пер.
ул. Тимура
Зубовский
мольский просп.
Парк культуры
Акрополь


12 м
пер.
ул. Тимура
Зубовский
мольский просп.
Парк культуры
Акрополь

План

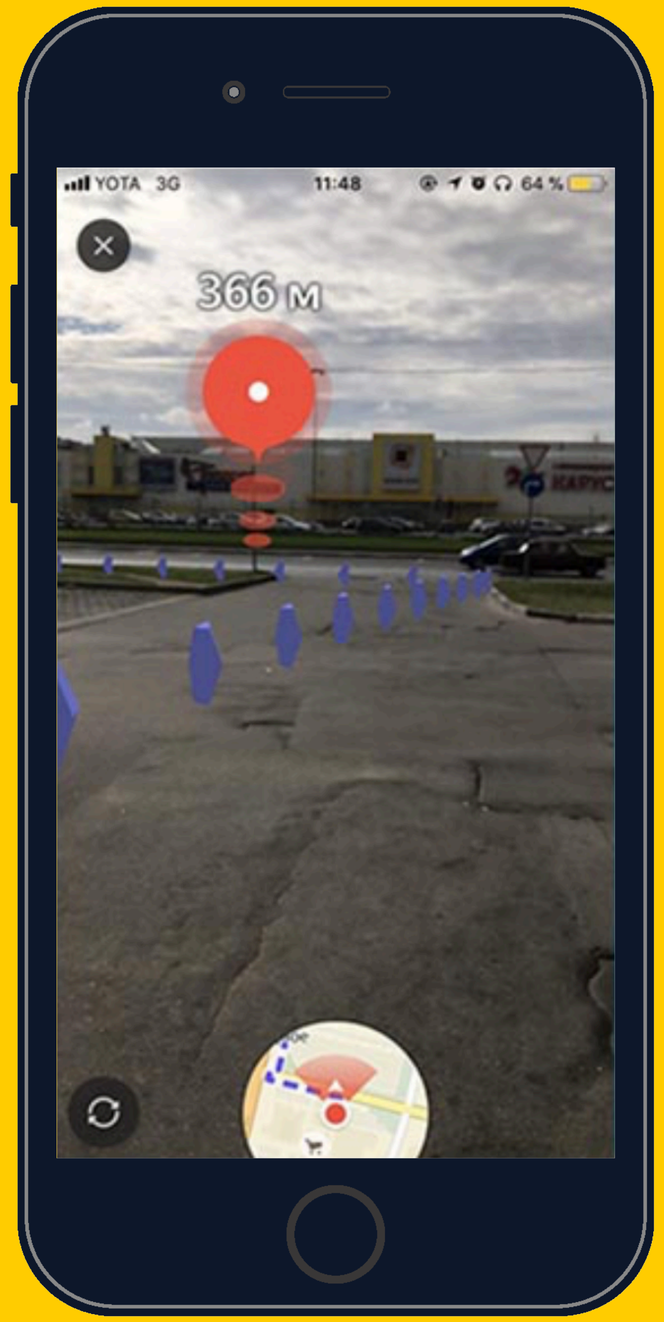
Дополненная реальность

- AR, VR, AV. Факты и не только

Пешеходная маршрутизация

- SceneKit, ARKit
- Позиционирование маршрута на 3D сцене
- Рендеринг объектов маршрута

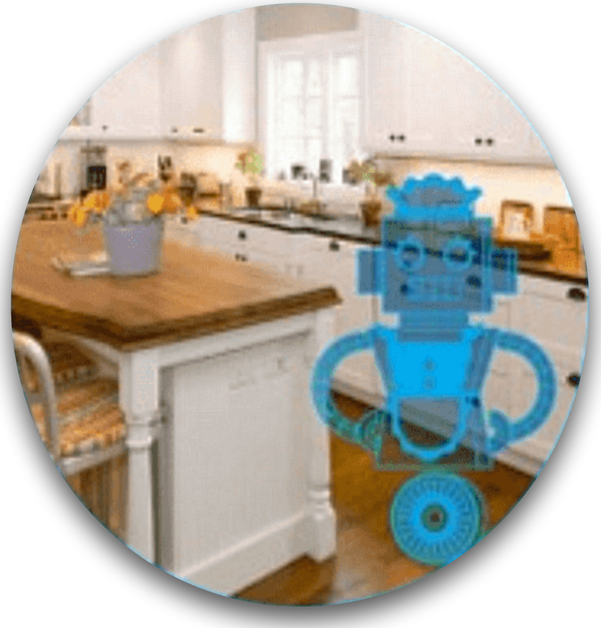
Статистика использования и что это нам дало



VIRTUALITY CONTINUUM

MIXED REALITY

Reality

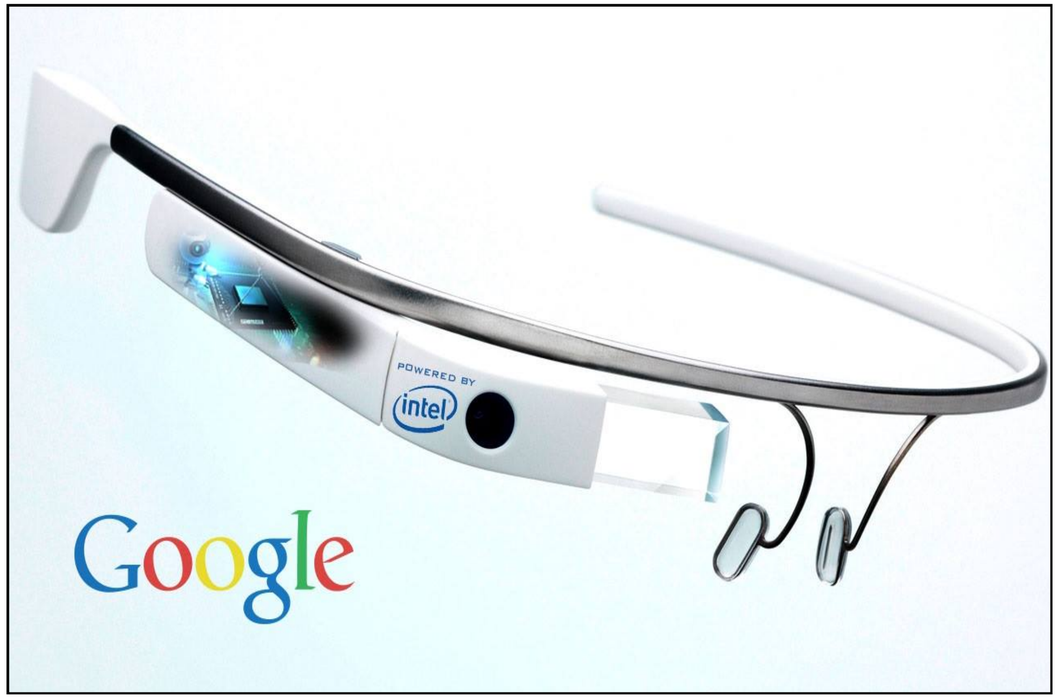


Augmented Reality (AR)

Augmented Virtuality (AV)

Virtual Reality

Augmented Reality
ISSUE
WWW.ESQUIRE.COM/AR
Esquire
MAN AT HIS BEST
DECEMBER 2009
ROBERT
DOWNEY JR.
PRESENTS
THE BEST
& BRIGHTEST
Radicals
WTF?!
A LIVING,
BREATHING,
MOVING,
TALKING
MAGAZINE?


POWERED BY
intel
Google

2009

2014

2016

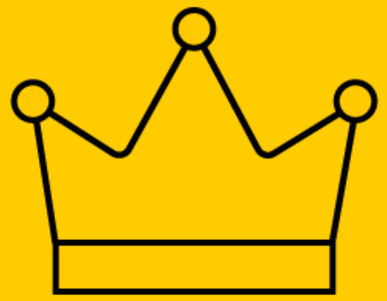
2017

Конкуренты ARKit

- Google Tango Project*
  - ограниченный ряд устройств
- Microsoft Hololens
  - цена ~ 3000$
  - необходим шлем
- Qualcomm Vuforia, ARToolkit и другие
  - необходим специальный маркер для «привязки» к реальному миру

* лежал в основе ARCore

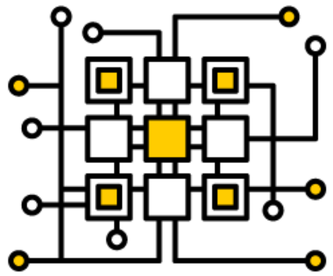
Секрет успеха ARKit



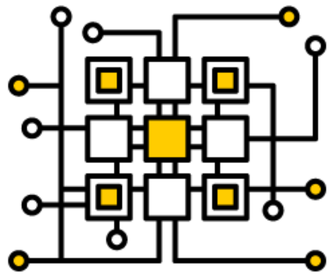
- Не нужны специальные метки
- Интеграция со `SceneKit`, `SpriteKit`…
- Решен вопрос синхронизации времени в подсистемах мобильного устройства
- Ошибки инерциальной системы сведены к минимуму*
- Никакой калибровки

* Практически идеальное измерение ускорения

Под капотом ARKit

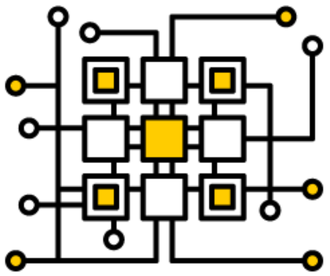


Под капотом ARKit



- Visual Inertial Odometry

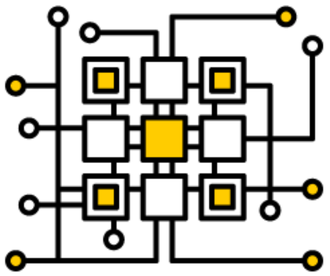
Под капотом ARKit



Visual Inertial Odometry

- объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Под капотом ARKit

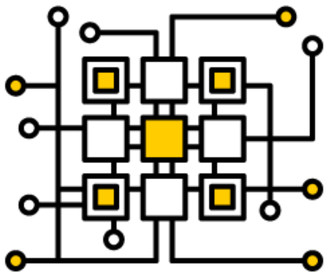


Visual Inertial Odometry

› объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Scene Undestanding

Под капотом ARKit



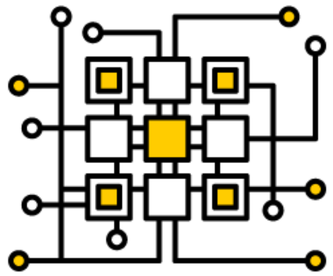
Visual Inertial Odometry

- объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Scene Undestanding

- детектирование горизонтальных и вертикальных плоскостей

Под капотом ARKit



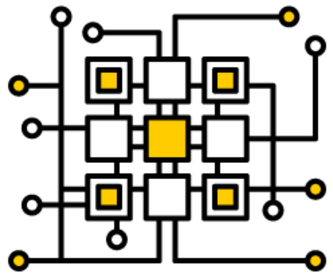
Visual Inertial Odometry

- объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Scene Undestanding

- детектирование горизонтальных и вертикальных плоскостей
- возможность добавления виртуальных объектов на сцену и отслеживания положения и ориентации реальных объектов

Под капотом ARKit



Visual Inertial Odometry

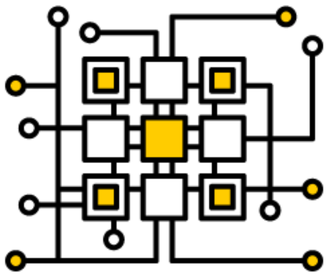
- объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Scene Undestanding

- детектирование горизонтальных и вертикальных плоскостей
- возможность добавления виртуальных объектов на сцену и отслеживания положения и ориентации реальных объектов

Lighting Estimation

Под капотом ARKit



Visual Inertial Odometry

- объединяет данные акселерометра, гироскопа и камеры для определение положения и смещения на сцене

Scene Undestanding

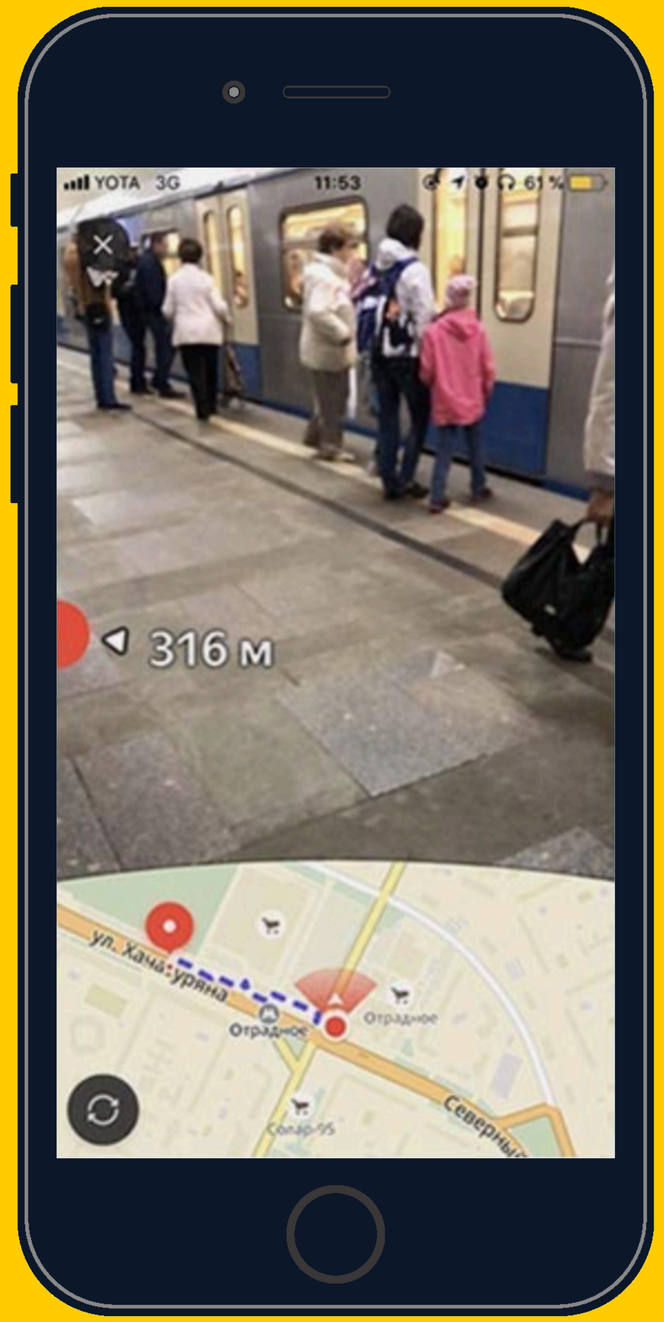
- детектирование горизонтальных и вертикальных плоскостей
- возможность добавления виртуальных объектов на сцену и отслеживания положения и ориентации реальных объектов

Lighting Estimation

- определяет условия освещения сцены

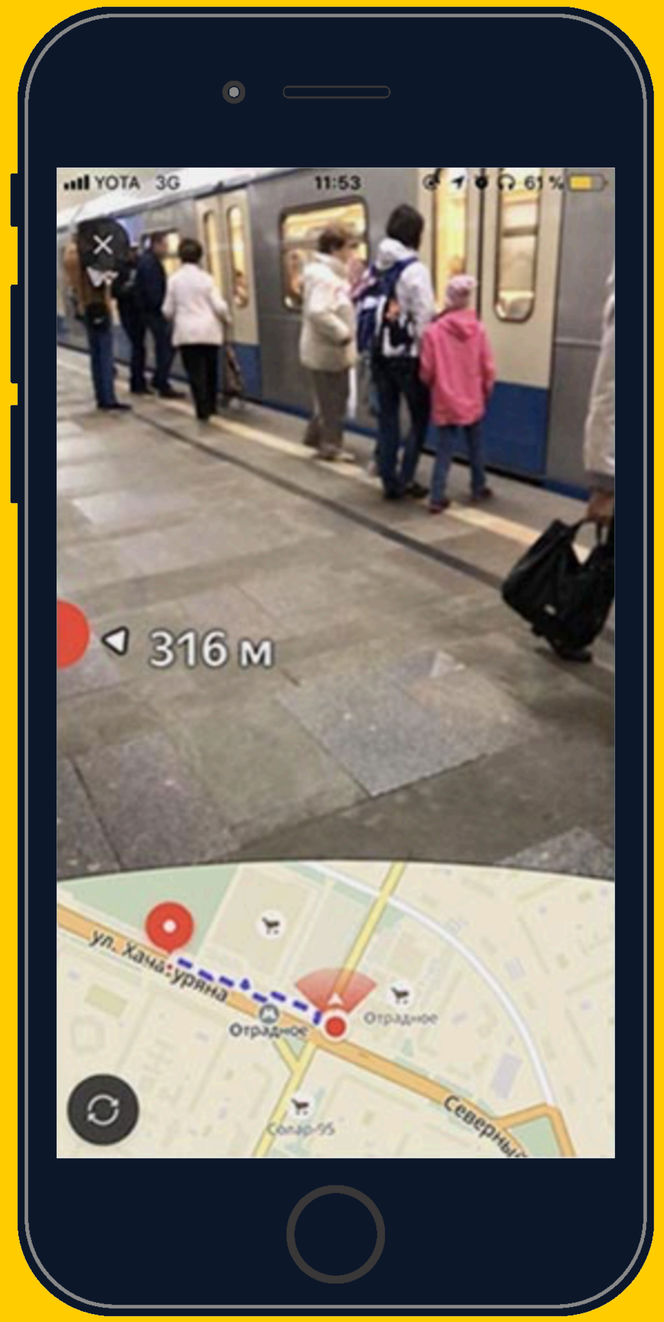
Маршрутизация с AR

Маршрутизация с AR



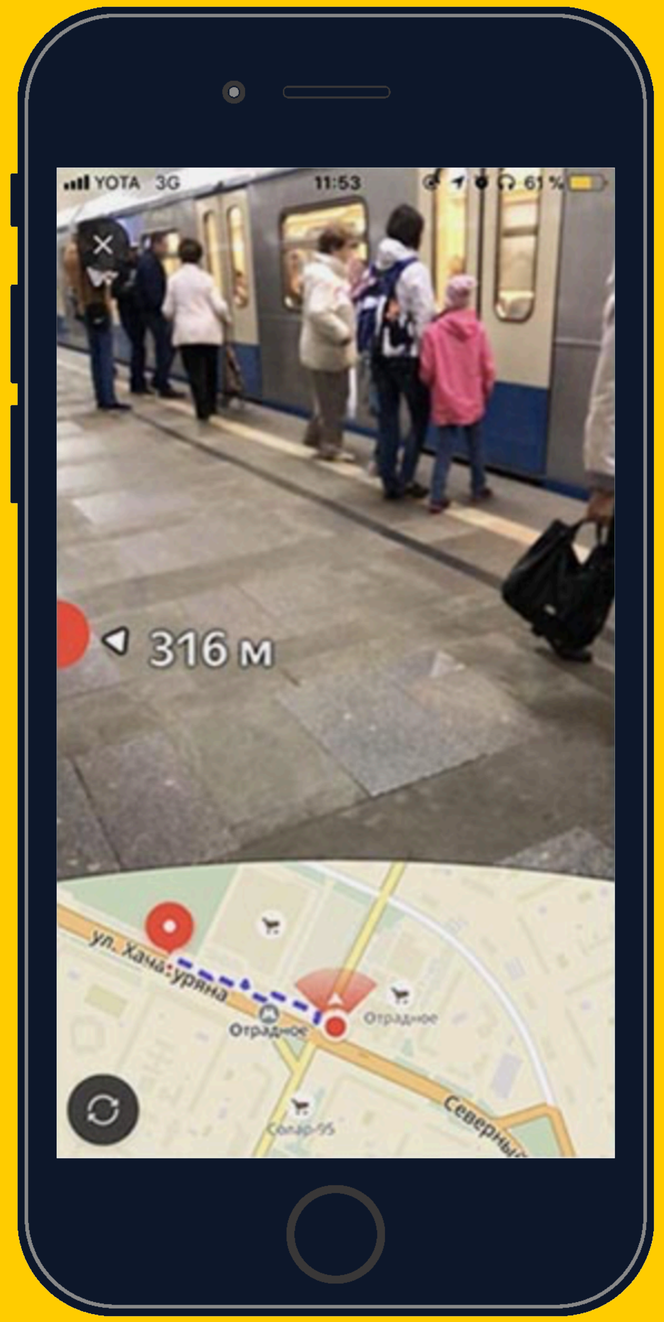
Маршрутизация с AR

- Получение маршрута



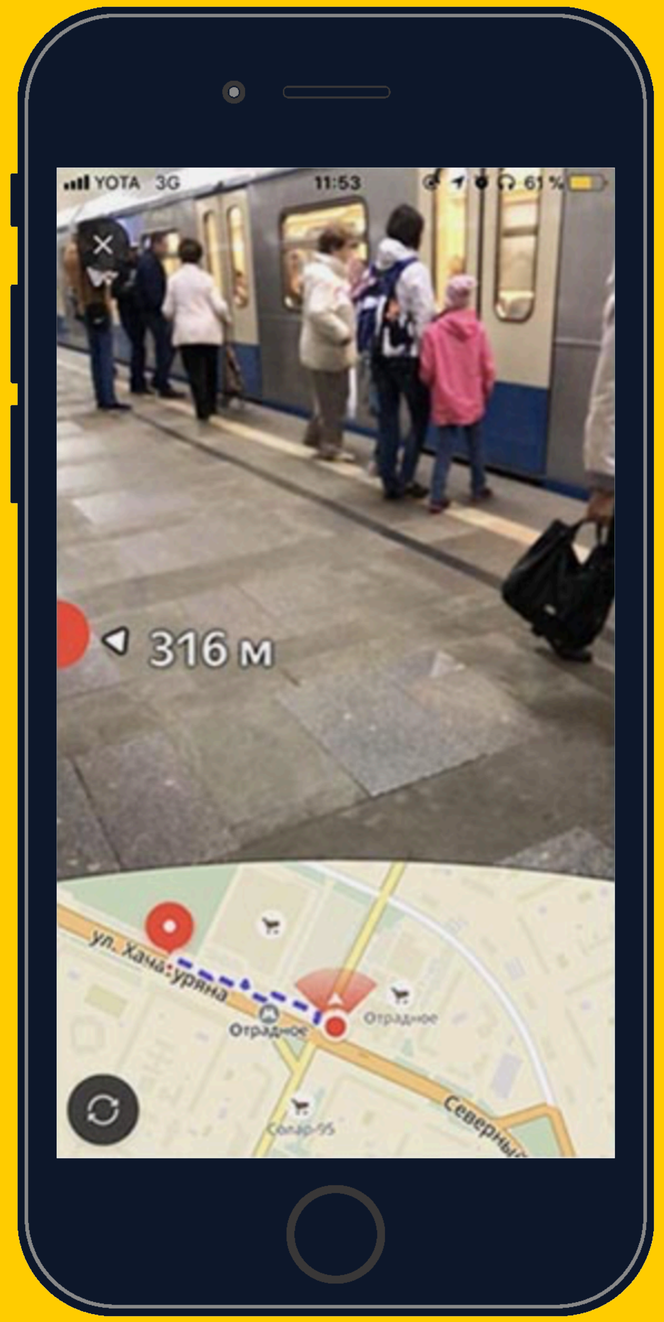
Маршрутизация с AR

- Получение маршрута
- Позиционирование объектов маршрута на 3D сцене



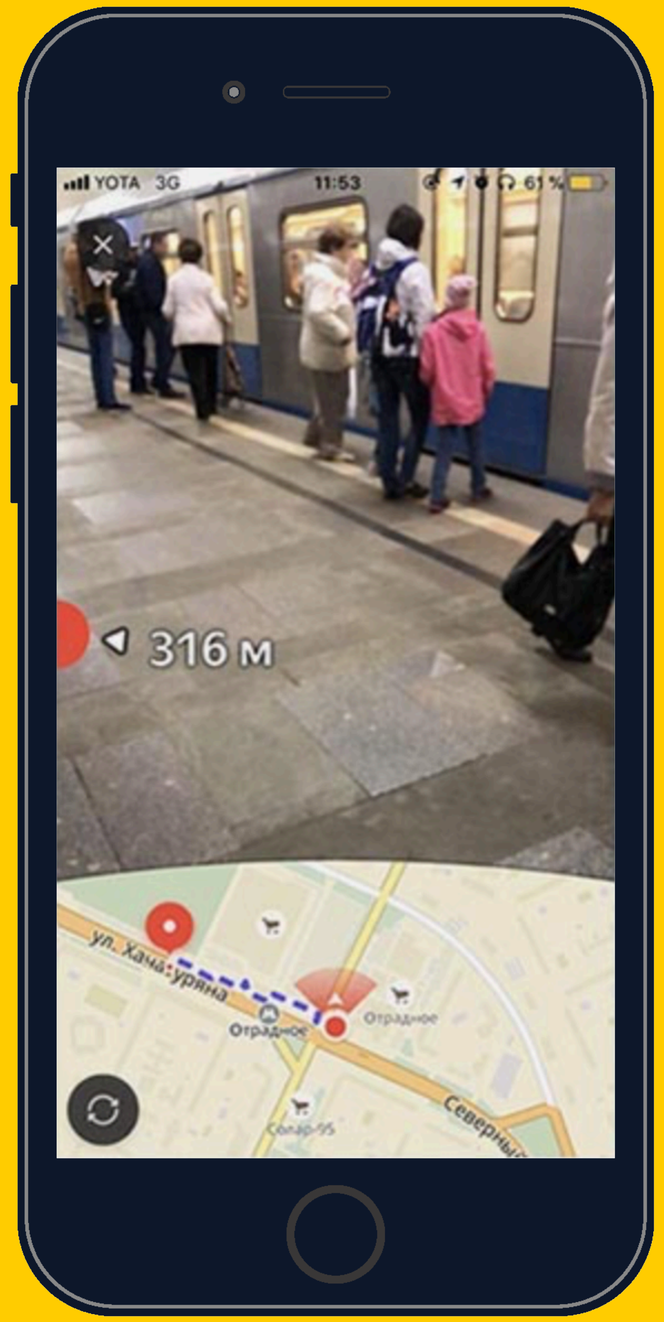
Маршрутизация с AR

- Получение маршрута
- Позиционирование объектов маршрута на 3D сцене
- Отрисовка промежуточных точек маршрута, метки финиша и вспомогательной метки



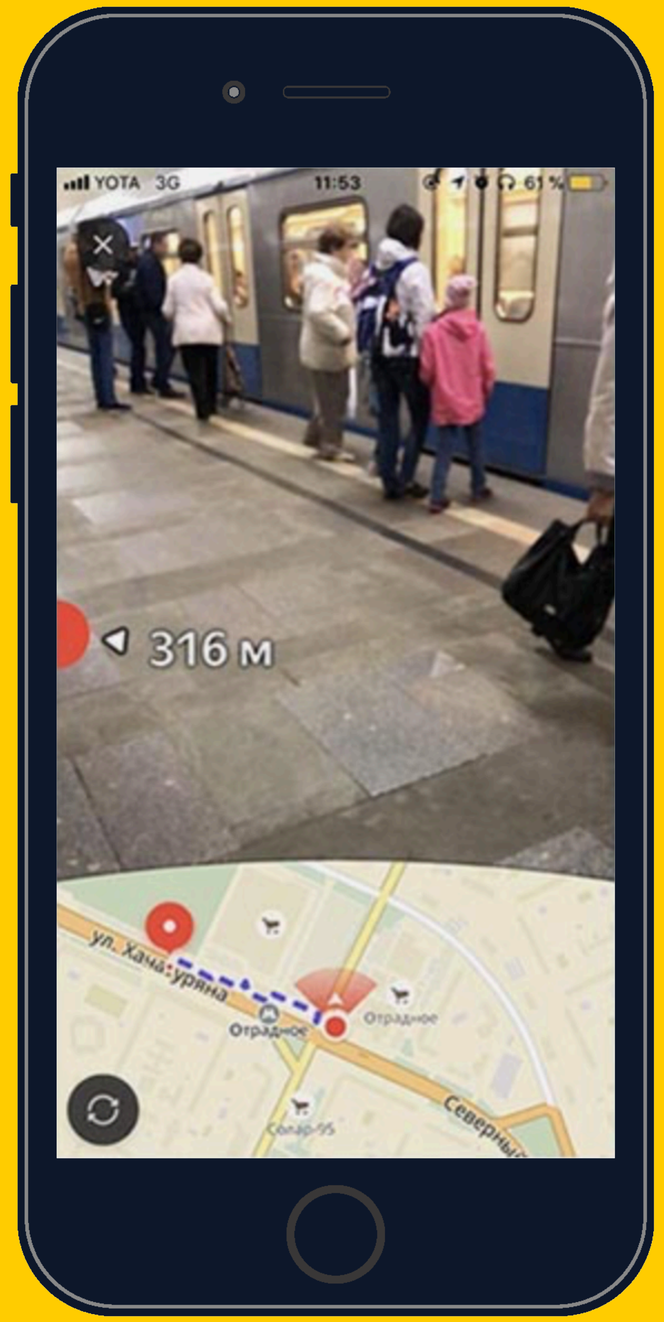
Маршрутизация с AR

- Получение маршрута
- Позиционирование объектов маршрута на 3D сцене
- Отрисовка промежуточных точек маршрута, метки финиша и вспомогательной метки
- Интеграция в карты



Маршрутизация с AR

- Получение маршрута
- Позиционирование объектов маршрута на 3D сцене
- Отрисовка промежуточных точек маршрута, метки финиша и вспомогательной метки
- Интеграция в карты



Маршрутизация с AR

Маршрутизация с AR

- Позиционирование объектов маршрута на 3D сцене

Маршрутизация с AR

Позиционирование объектов маршрута на 3D сцене

› перевод `CLLocation` в позицию на сцене `(SCNVector3)`

Маршрутизация с AR

Позиционирование объектов маршрута на 3D сцене

- перевод `CLLocation` в позицию на сцене `(SCNVector3)`
- коррекция системы координат сцены

Маршрутизация с AR

- Позиционирование объектов маршрута на 3D сцене
  - перевод `CLLocation` в позицию на сцене `(SCNVector3)`
  - коррекция системы координат сцены
- Отрисовка элементов маршрута

Маршрутизация с AR

- Позиционирование объектов маршрута на 3D сцене
  - перевод `CLLocation` в позицию на сцене `(SCNVector3)`
  - коррекция системы координат сцены
- Отрисовка элементов маршрута
  - метка финиша

Маршрутизация с AR

Позиционирование объектов маршрута на 3D сцене

- перевод `CLLocation` в позицию на сцене `(SCNVector3)`
- коррекция системы координат сцены

Отрисовка элементов маршрута

- метка финиша
- вспомогательная метка

Маршрутизация с AR

Позиционирование объектов маршрута на 3D сцене

- перевод `CLLocation` в позицию на сцене `(SCNVector3)`
- коррекция системы координат сцены

Отрисовка элементов маршрута

- метка финиша
- вспомогательная метка
- линия маршрута

Инструменты

ARSession

Объект сессии дополненной реальности

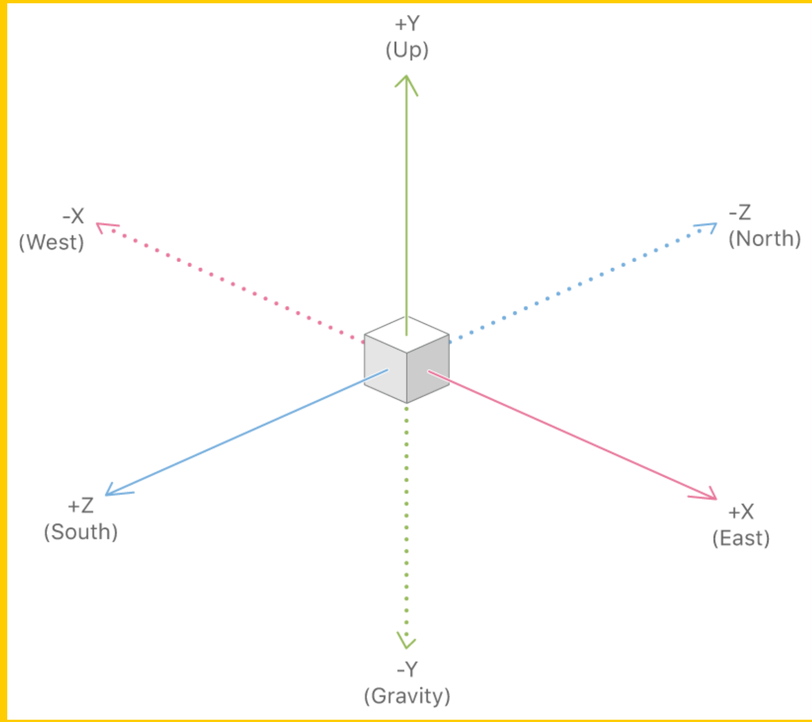
- осуществляет процессинг данных
- конфигурируется объектом `ARConfiguration`
- `run(ARConfiguration, …), pause()`
- `delegate`
- …

WorldAlignment

Определяет направление координатных осей сцены в момент инициализации сессии

- `gravity`
- `gravityAndHeading`
- `camera`

Положение системы координат совпадает с положением телефона в момент старта сессии



ARSCNView

Наследник SCNView для работы с ARKit

- синхронизирует положение телефона в пространстве с положением камеры на сцене
- система координат сцены совпадает с системой координат ARKit
- добавляет к сцене изображение с камеры телефона
- предоставляет объект сессии

SCNNode

Элемент графа сцены

- представляет позицию в 3D пространстве относительно своего родителя (`parent`)
- не имеет видимого контента сам по себе
- может являться `pointOfView` сцены
- аналог UIView в мире SceneKit

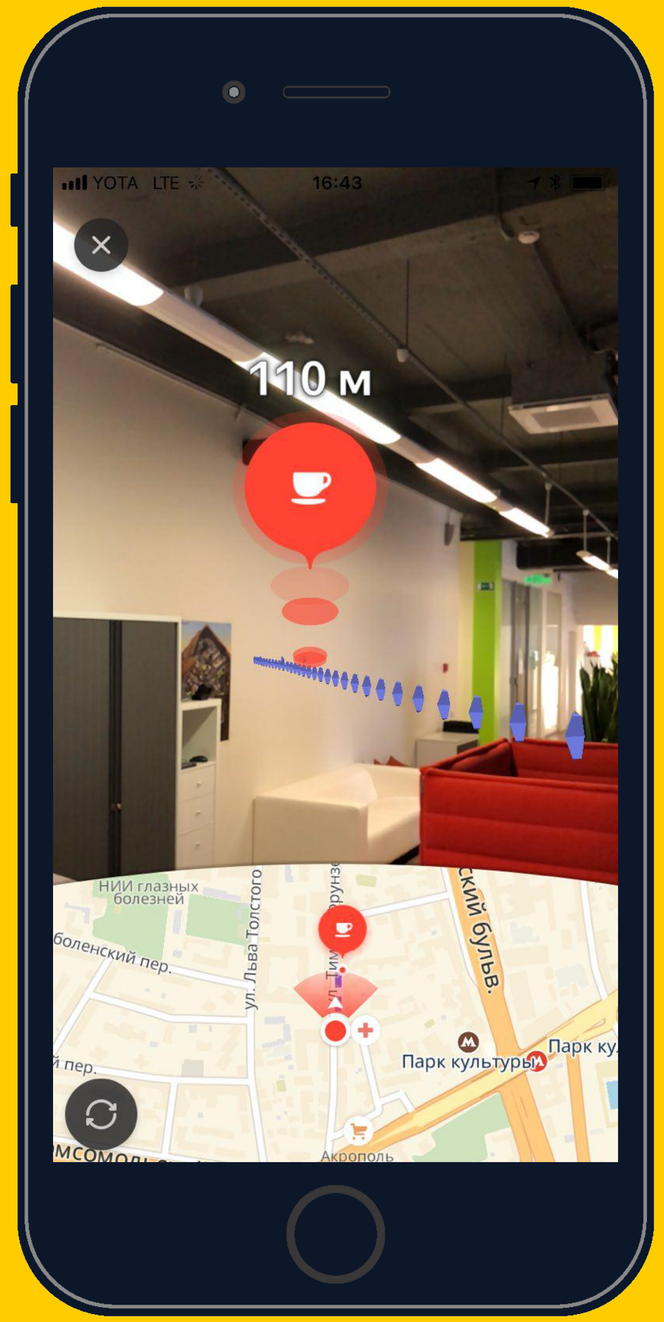
Отрисовка маршрута – добавление объектов SCNNode на сцену и расчет их позиции на сцене по географическим координатам

Отрисовка маршрута – добавление объектов SCNNode на сцену и расчет их позиции на сцене по географическим координатам

Метка финиша

Метка финиша

- Позиция
- Размер
- Анимация
- Билборд с расстоянием



UIKit
SceneKit
FIGHT

После прототипа появились они...

После прототипа появились они...

Требования к размеру

> 2 км

фиксированный
минимальный размер

Требования к размеру

> 2 км

фиксированный
минимальный размер

50 м - 2 км

размер увеличивается
пропорционально
расстоянию

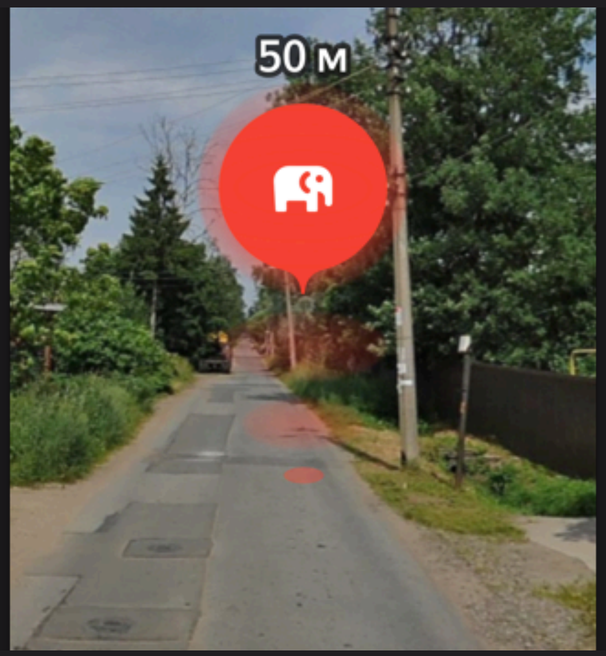
Требования к размеру

> 2 км

фиксированный
минимальный размер

50 м - 2 км

размер увеличивается
пропорционально
расстоянию

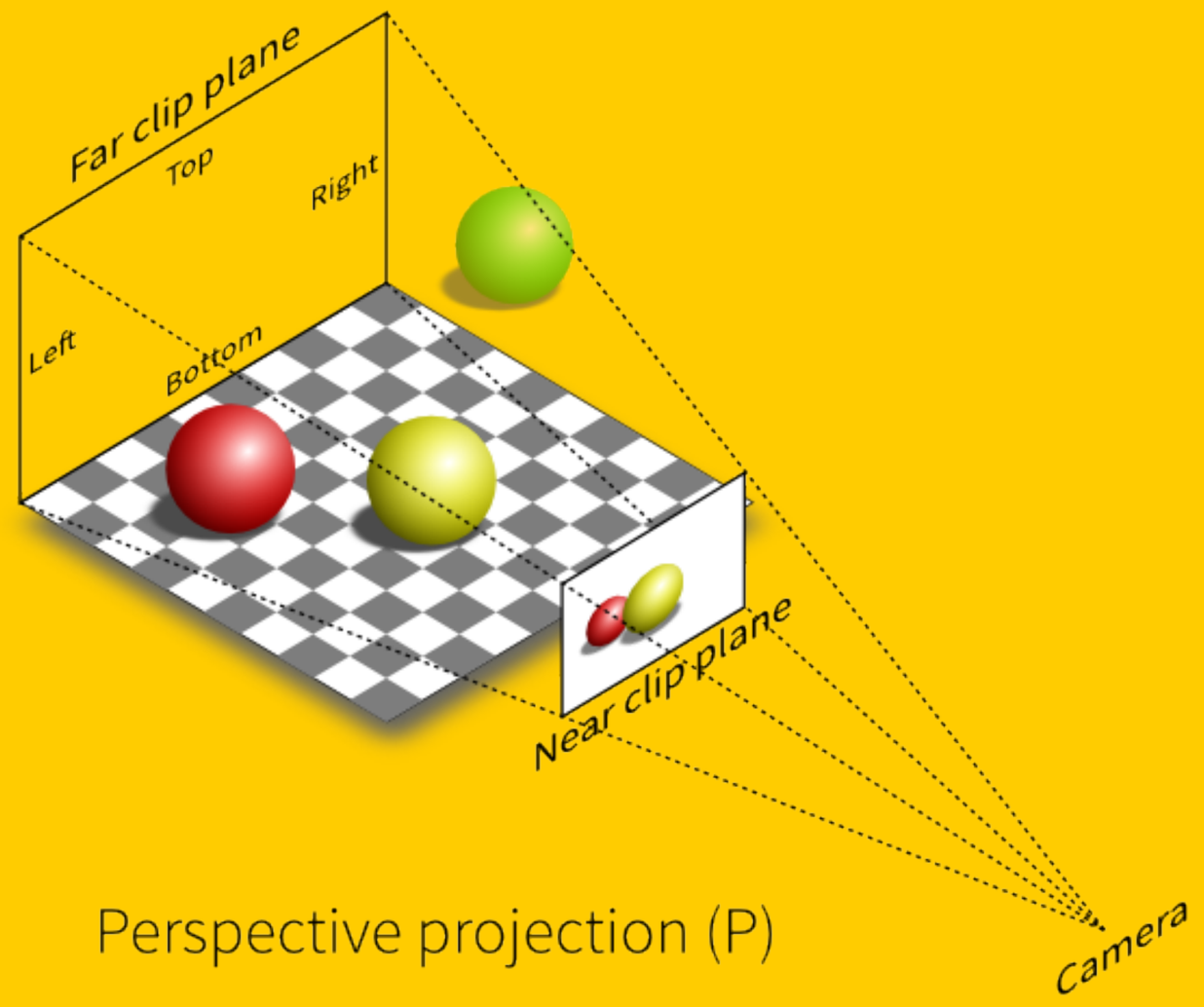


< 50 м

фиксированный
максимальный размер

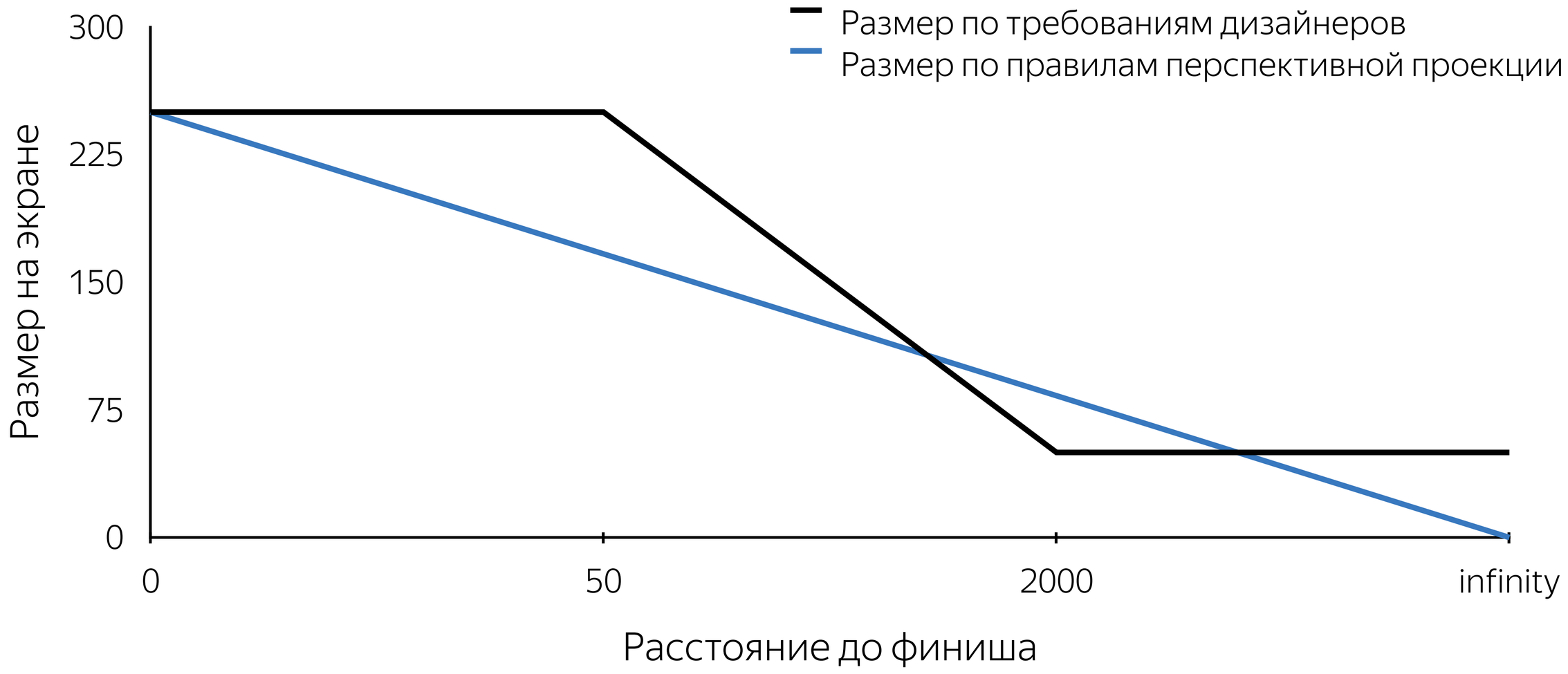
Перспективная проекция

- При отдалении камеры объект становится меньше
- Логика проектирования объектов 3-х мерной сцены - ядро SceneKit*



* и не только SceneKit. Любого движка или фреймворка для работы с 3D графикой

Требования к размеру



ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

(x,y)
ВИДЕО
ФОТО
КВАДРАТ

220 м
(x,y)
ВИДЕО
ФОТО
КВАДРАТ

220 м
ВИДЕО
ФОТО
КВАДРАТ

220 м
ВИДЕО
ФОТО
КВАДРАТ

Метка финиша на UIKit

```
func renderer(_ renderer: SCNSceneRenderer, didRenderScene scene: SCNScene, atTime
time: TimeInterval) {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    DispatchQueue.main.async {
        self.routeFinishView.frame = CGRect(x: projection.x - sizeOnScreen / 2,
                                            y: projection.y - sizeOnScreen / 2,
                                            width: sizeOnScreen,
                                            height: sizeOnScreen)
    }
}
```

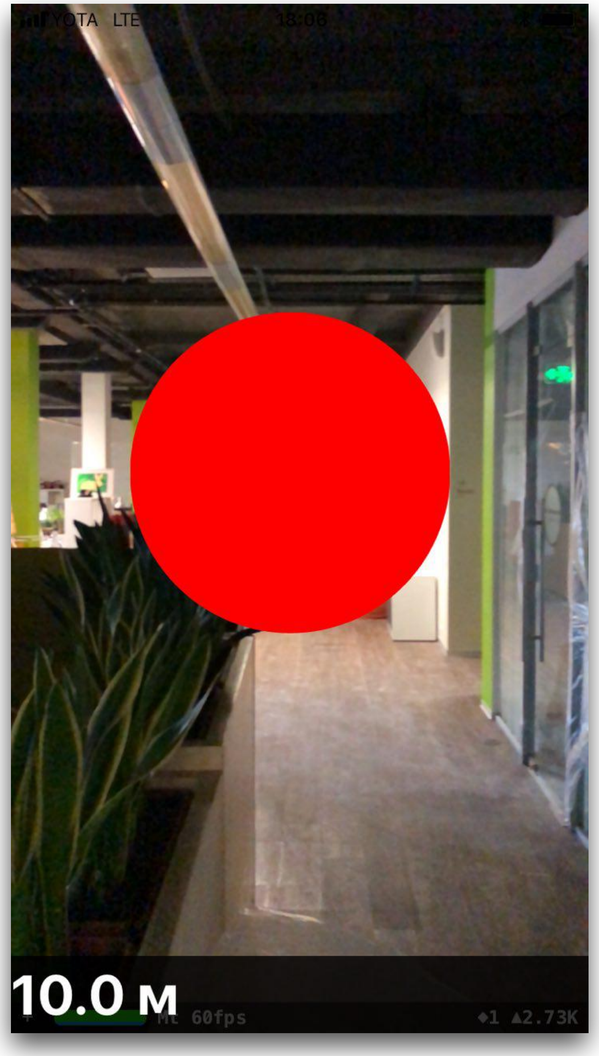
Метка финиша на UIKit

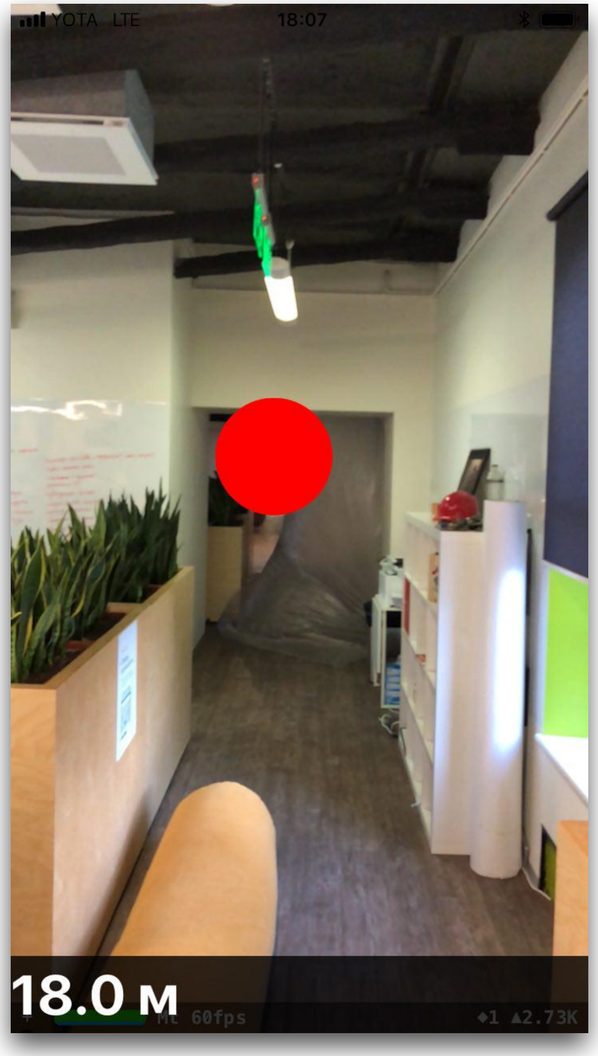
```
func renderer(_ renderer: SCNSceneRenderer, didRenderScene scene: SCNScene, atTime
time: TimeInterval) {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    DispatchQueue.main.async {
        self.routeFinishView.frame = CGRect(x: projection.x - sizeOnScreen / 2,
                                            y: projection.y - sizeOnScreen / 2,
                                            width: sizeOnScreen,
                                            height: sizeOnScreen)
    }
}
```

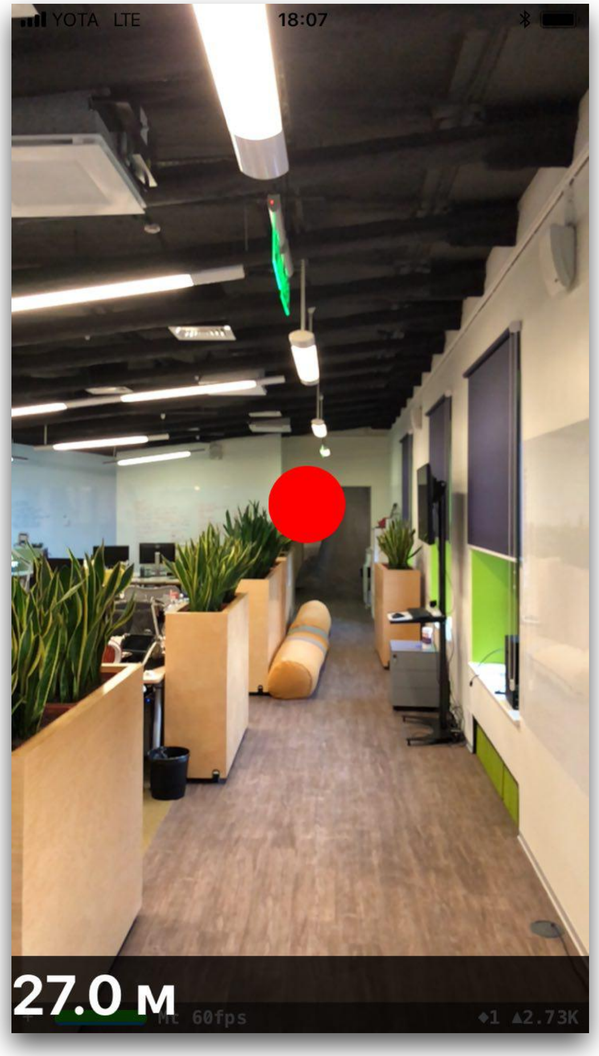
Метка финиша на UIKit

```
func renderer(_ renderer: SCNSceneRenderer, didRenderScene scene: SCNScene, atTime
time: TimeInterval) {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    DispatchQueue.main.async {
        self.routeFinishView.frame = CGRect(x: projection.x - sizeOnScreen / 2,
                                            y: projection.y - sizeOnScreen / 2,
                                            width: sizeOnScreen,
                                            height: sizeOnScreen)
    }
}
```

Результат в UIKit







ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

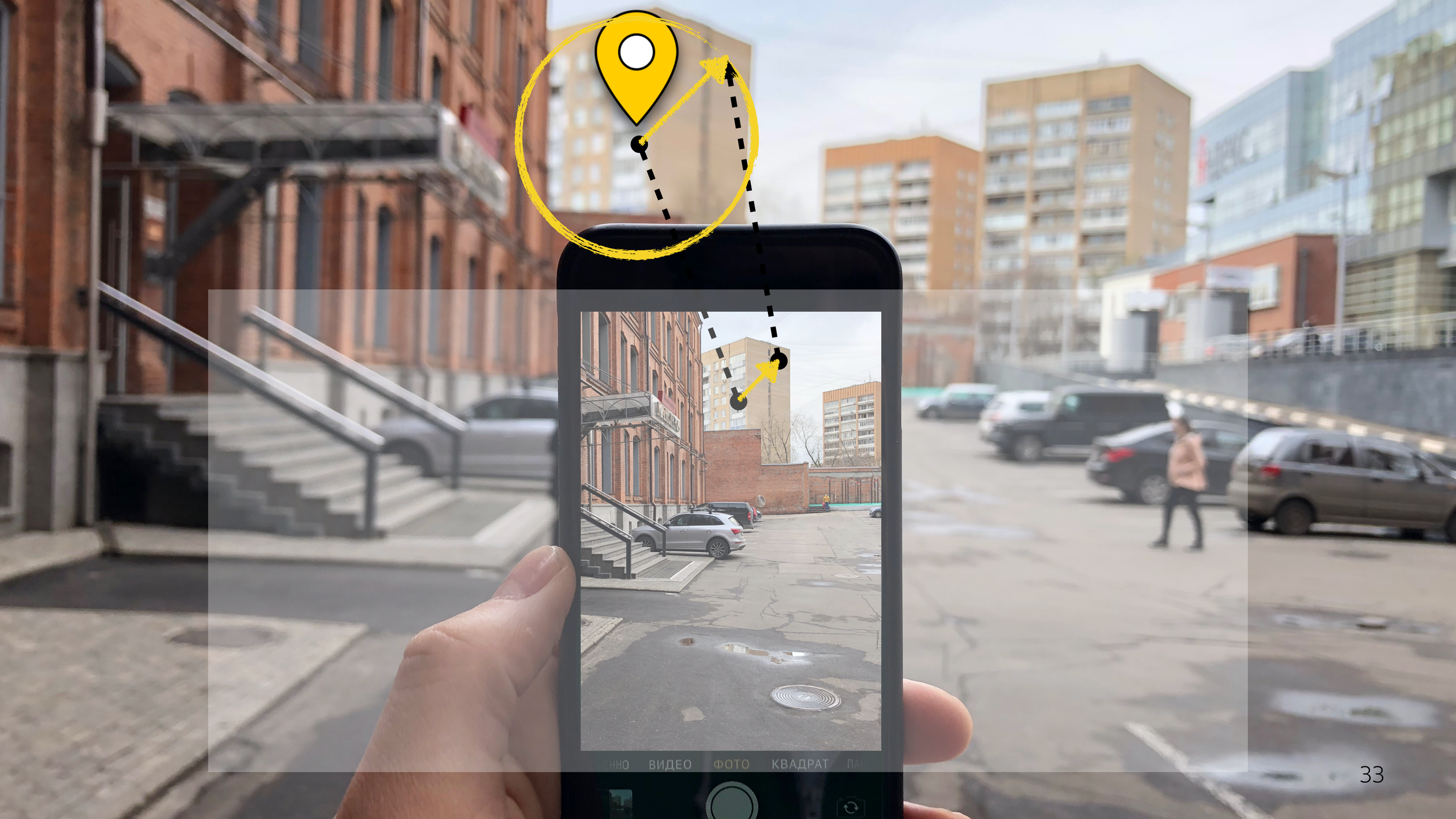
ВИДЕО
ФОТО
КВАДРАТ

(x,y)
ВИДЕО
ФОТО
КВАДРАТ

220 м
(x,y)
ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ


ВИДЕО
ФОТО
КВАДРАТ

Определение радиуса геометрии

```
func finishNodeSize(forNode node: SCNNode, inSceneOfView scnView: SCNView) -> Float {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    let pointToUnproject = projection.shifted(byX: sizeOnScreen/2, y: 0.0)
    let unprojectedPointInWorld = scnView.unprojectPoint(pointToUnproject)
    return (node.worldPosition - unprojectedPointInWorld).length
}
```

Определение радиуса геометрии

```
func finishNodeSize(forNode node: SCNNode, inSceneOfView scnView: SCNView) -> Float {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    let pointToUnproject = projection.shifted(byX: sizeOnScreen/2, y: 0.0)
    let unprojectedPointInWorld = scnView.unprojectPoint(pointToUnproject)
    return (node.worldPosition - unprojectedPointInWorld).length
}
```

Определение радиуса геометрии

```
func finishNodeSize(forNode node: SCNNode, inSceneOfView scnView: SCNView) -> Float {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    let pointToUnproject = projection.shifted(byX: sizeOnScreen/2, y: 0.0)
    let unprojectedPointInWorld = scnView.unprojectPoint(pointToUnproject)
    return (node.worldPosition - unprojectedPointInWorld).length
}
```

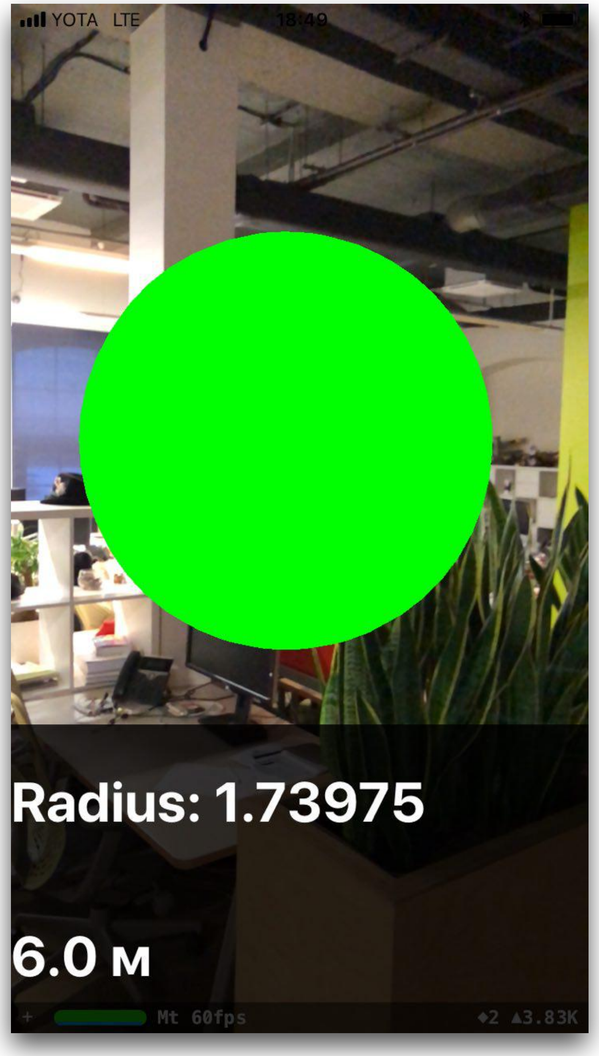
Определение радиуса геометрии

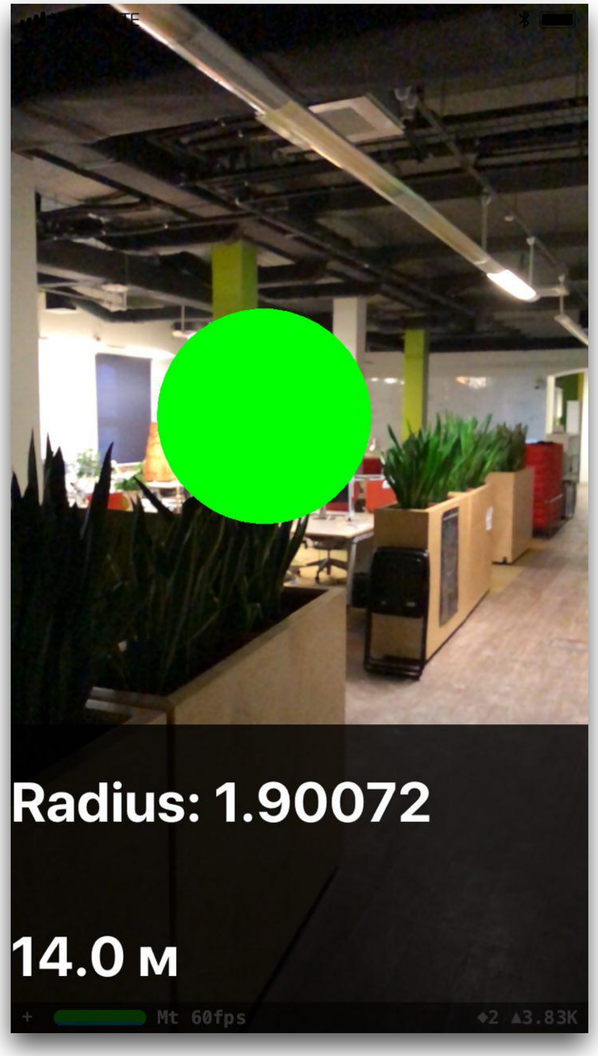
```
func finishNodeSize(forNode node: SCNNode, inSceneOfView scnView: SCNView) -> Float {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    let pointToUnproject = projection.shifted(byX: sizeOnScreen/2, y: 0.0)
    let unprojectedPointInWorld = scnView.unprojectPoint(pointToUnproject)
    return (node.worldPosition - unprojectedPointInWorld).length
}
```

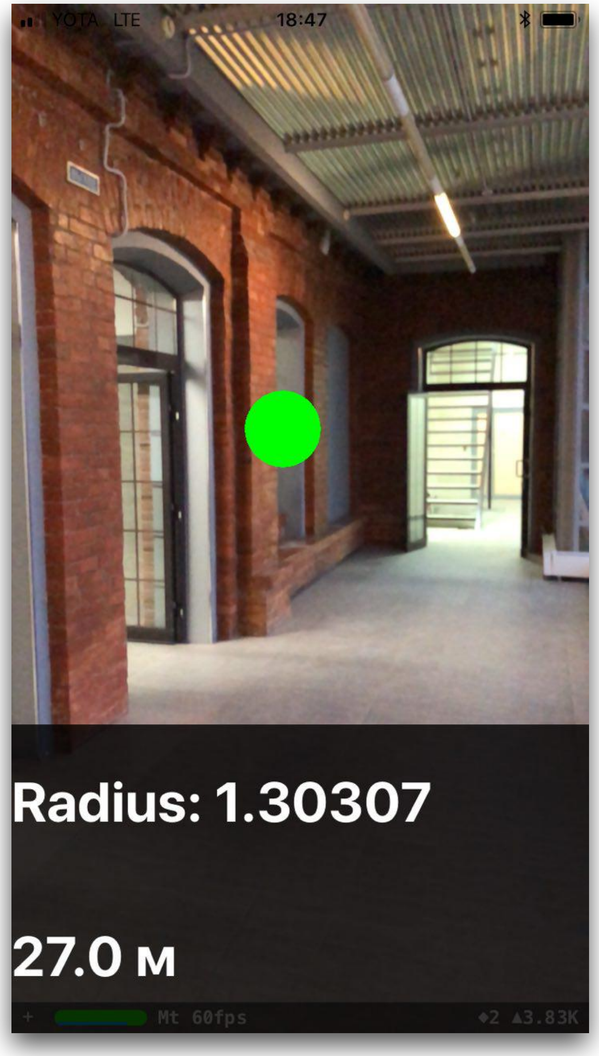
Определение радиуса геометрии

```
func finishNodeSize(forNode node: SCNNode, inSceneOfView scnView: SCNView) -> Float {
    let projection = findProjection(...)
    let distance = calculateDistance(...)
    let sizeOnScreen = finishPlacemarkSize(...)
    let pointToUnproject = projection.shifted(byX: sizeOnScreen/2, y: 0.0)
    let unprojectedPointInWorld = scnView.unprojectPoint(pointToUnproject)
    return (node.worldPosition - unprojectedPointInWorld).length
}
```

Результат. SceneKit





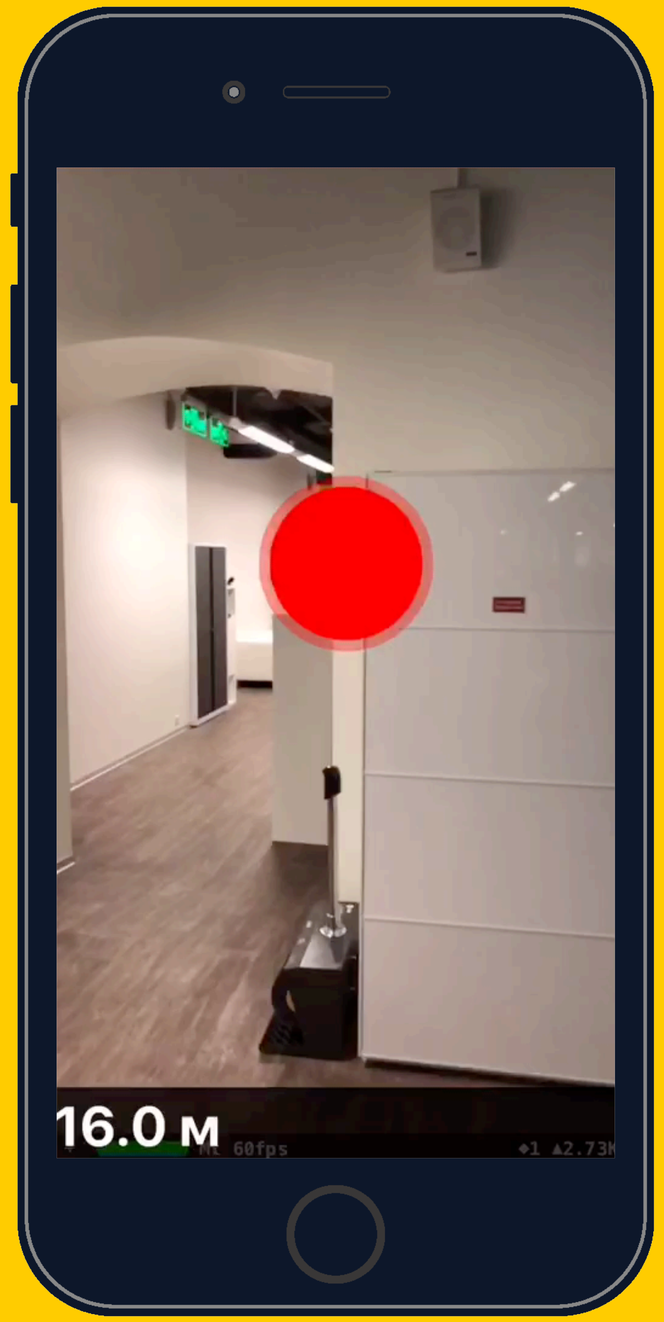


Анимация метки финиша

В каждом кадре мы явно задаем размер метки

- `view.frame` в UIKit
- `node.geometry.radius` в SceneKit

Добавить анимацию размера можно, используя другие свойства

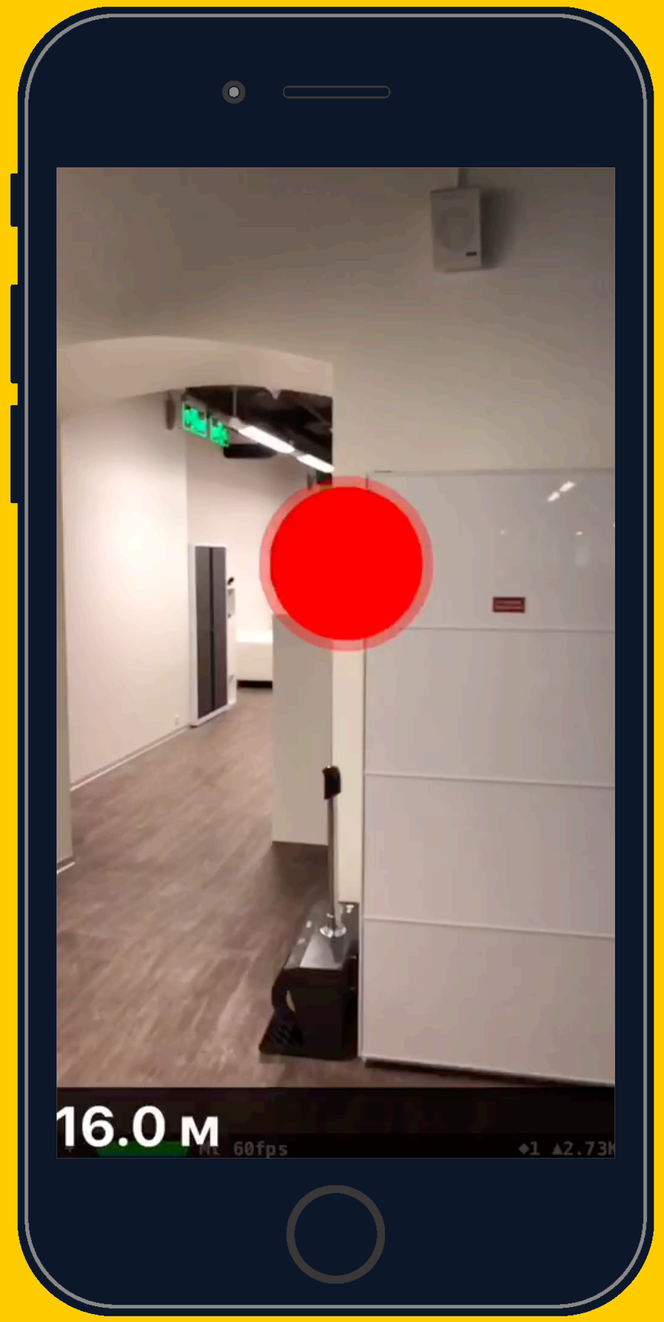


Анимация метки финиша

В каждом кадре мы явно задаем размер метки

- `view.frame` в UIKit
- `node.geometry.radius` в SceneKit

Добавить анимацию размера можно, используя другие свойства



Одна из возможных реализаций

UIKit	SceneKit
1. Используем `UIView`, идентичную основной 2. Добавим анимацию для свойства `transform.scale.xy`	1. Используем `SCNNode`, идентичный основному 2. Добавим анимацию для свойства `scale`

Создание дополнительного SCNNode

```
finishNodePulse = createSphereNode(withRadius: 0.0,
                                   color: UIColor.green)
finishNodePulse.position = SCNVector3Make(0.0, 0.0, 0.0)
finishNode.addChildNode(finishNodePulse)
```

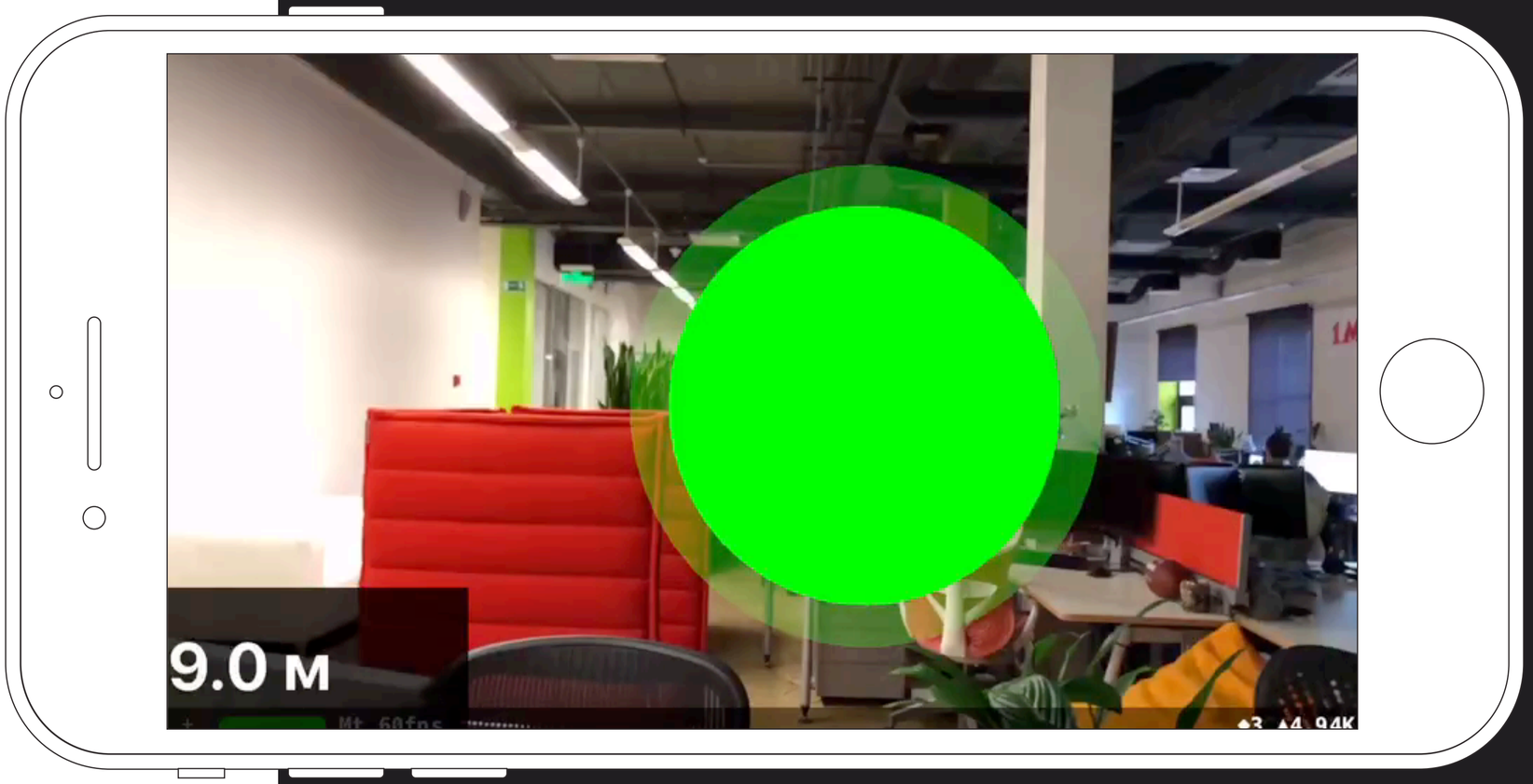
Создание дополнительного SCNNode

```
finishNodePulse = createSphereNode(withRadius: 0.0,
                                   color: UIColor.green)
finishNodePulse.position = SCNVector3Make(0.0, 0.0, 0.0)
finishNode.addChildNode(finishNodePulse)
```

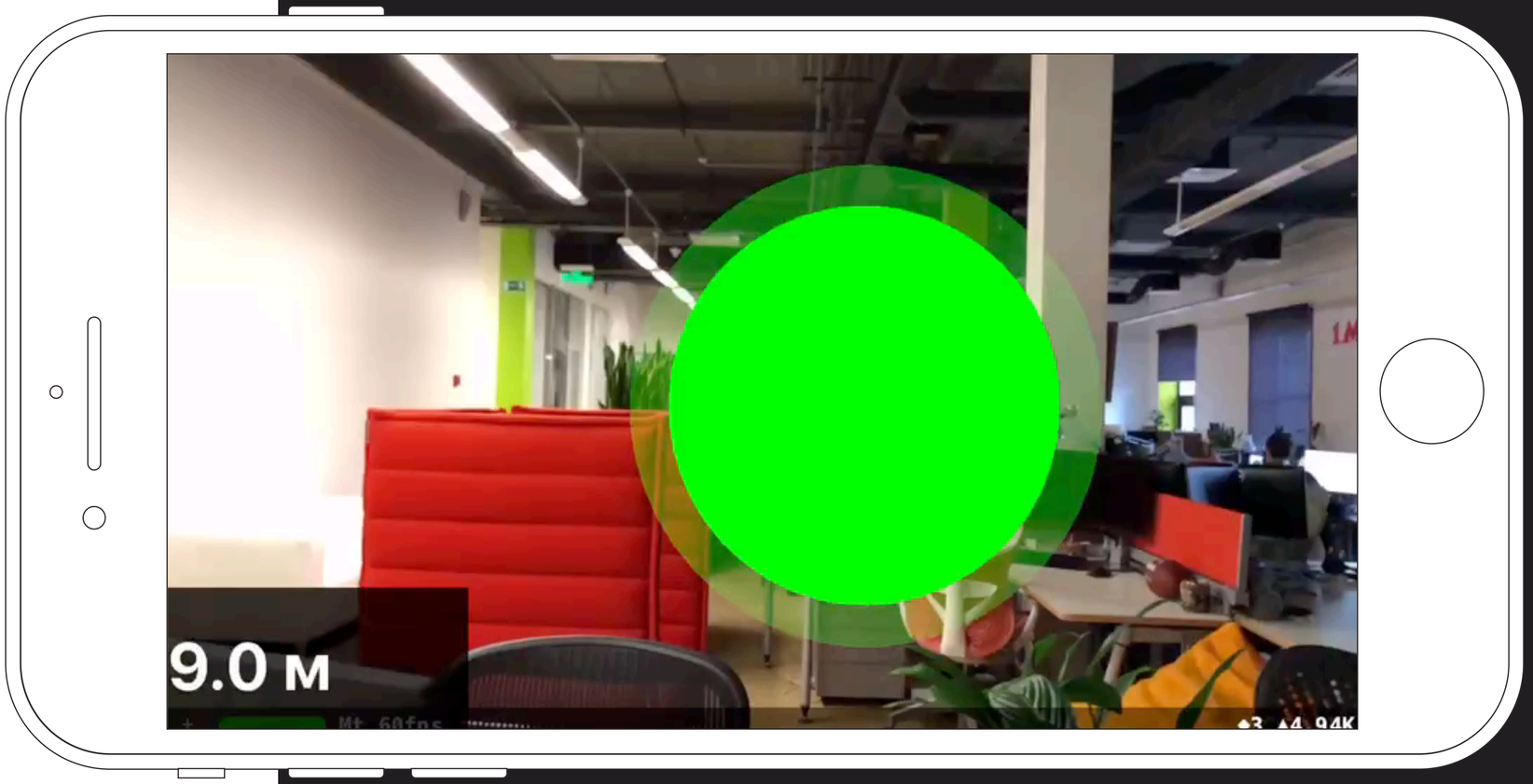
Анимация. CoreAnimation

```
let animationGroup = CAAnimationGroup.init()
animationGroup.duration = 1.0
animationGroup.repeatCount = .infinity

let opacityAnimation = CABasicAnimation(keyPath: "opacity")
opacityAnimation.fromValue = NSNumber(value: 1.0)
opacityAnimation.toValue = NSNumber(value: 0.1)

let scaleAnimation = CABasicAnimation(keyPath: "scale")
scaleAnimation.fromValue = NSValue(scnVector3: SCNVector3(1.0, 1.0, 1.0))
scaleAnimation.toValue = NSValue(scnVector3: SCNVector3(1.2, 1.2, 1.2))

animationGroup.animations = [opacityAnimation, scaleAnimation]
finishNodePulse.addAnimation(animationGroup, forKey: "animations")
```

Результат



Результат



Билборд с расстоянием до финиша

Билборд с расстоянием до финиша

> Как заставить лейбл ~~повернуться к лесу задом~~ всегда смотреть “лицом” в камеру и иметь правильную ориентацию?

Билборд с расстоянием до финиша

> Как заставить лейбл ~~повернуться к лесу задом~~ всегда смотреть “лицом” в камеру и иметь правильную ориентацию?

› в UIKit объект вью **не зависит** от положения телефона

Билборд с расстоянием до финиша

> Как заставить лейбл ~~повернуться к лесу задом~~ всегда смотреть “лицом” в камеру и иметь правильную ориентацию?

- в UIKit объект вью **не зависит** от положения телефона
- в SceneKit проекция объекта на экран **зависит** от положения телефона

Билборд с расстоянием до финиша

Как заставить лейбл ~~повернуться к лесу задом~~ всегда смотреть “лицом” в камеру и иметь правильную ориентацию?

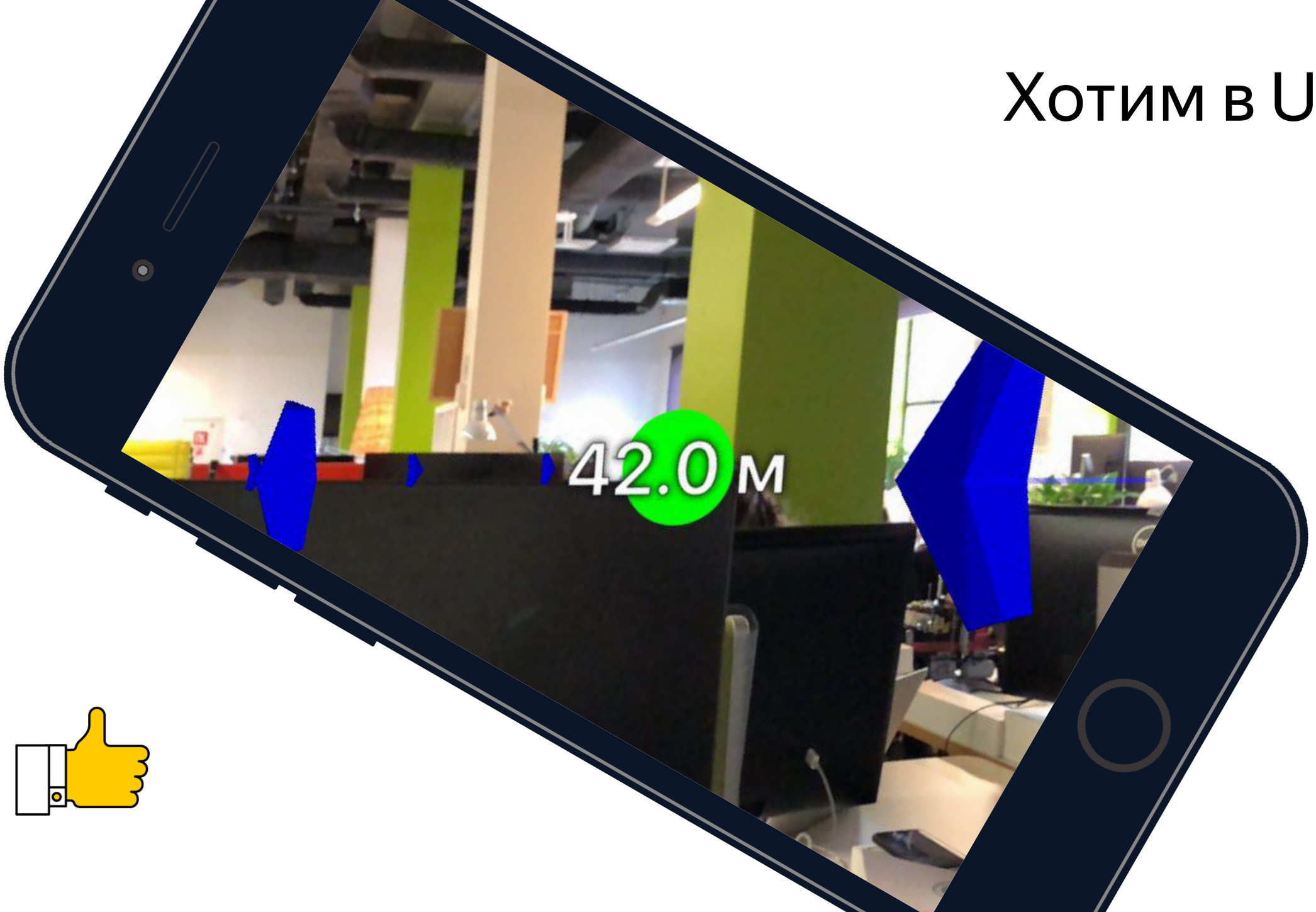
- в UIKit объект вью **не зависит** от положения телефона
- в SceneKit проекция объекта на экран **зависит** от положения телефона

Положение телефона - `camera.transform`

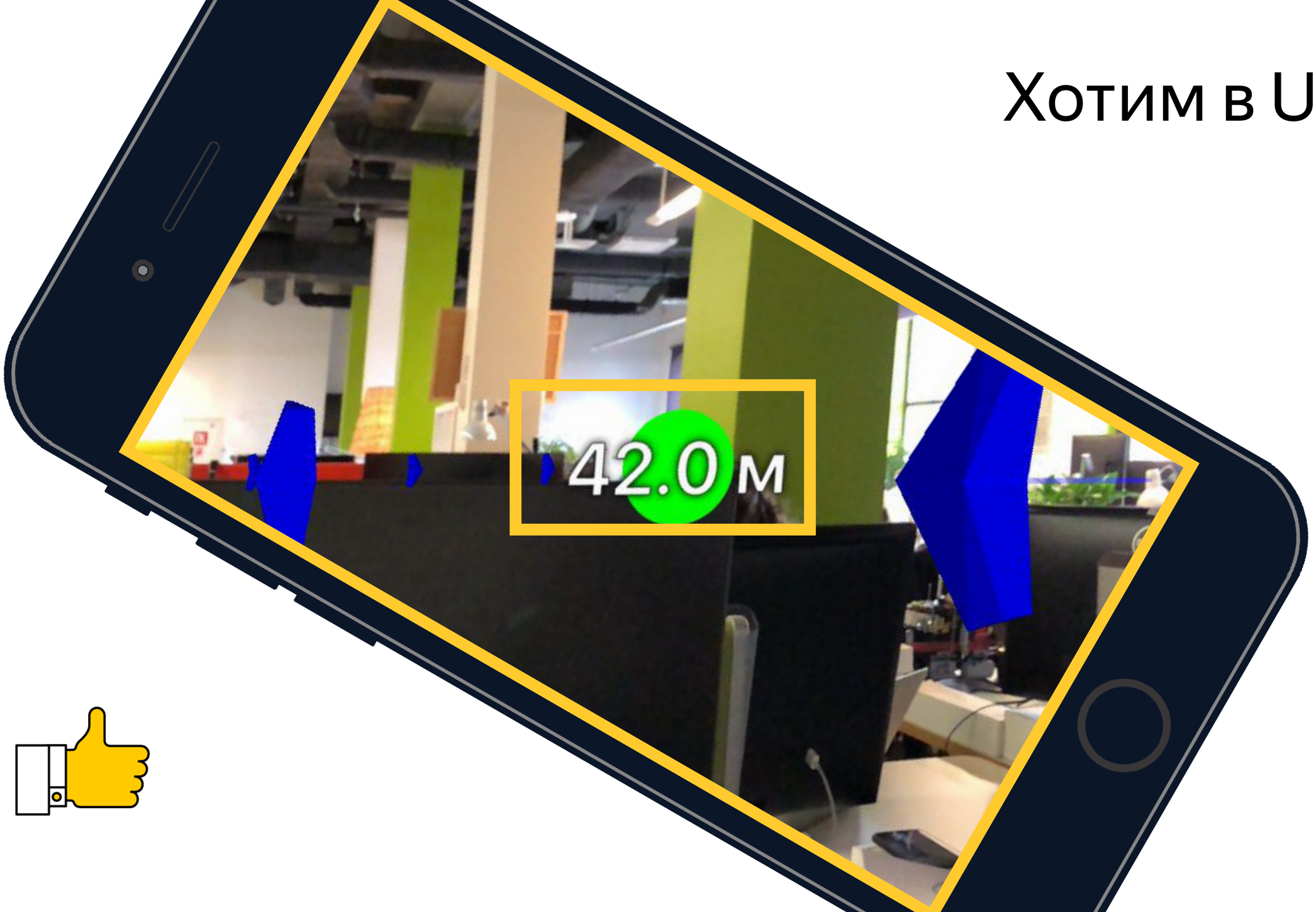
Проблема в UIKit

Проблема в UIKit

Хотим в UIKit

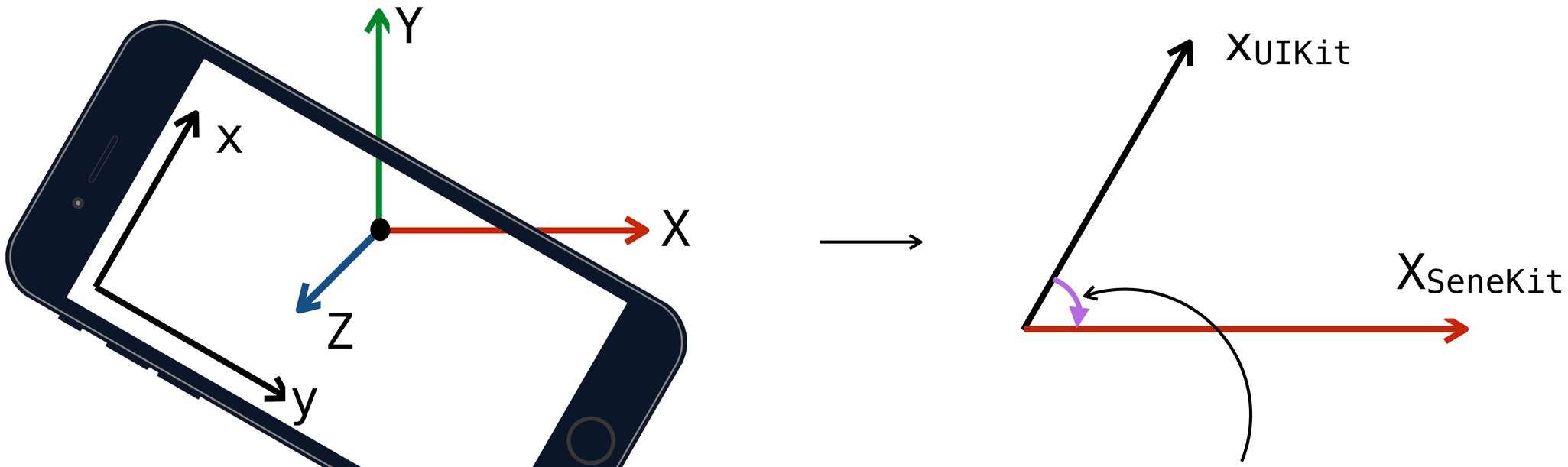


Хотим в UIKit



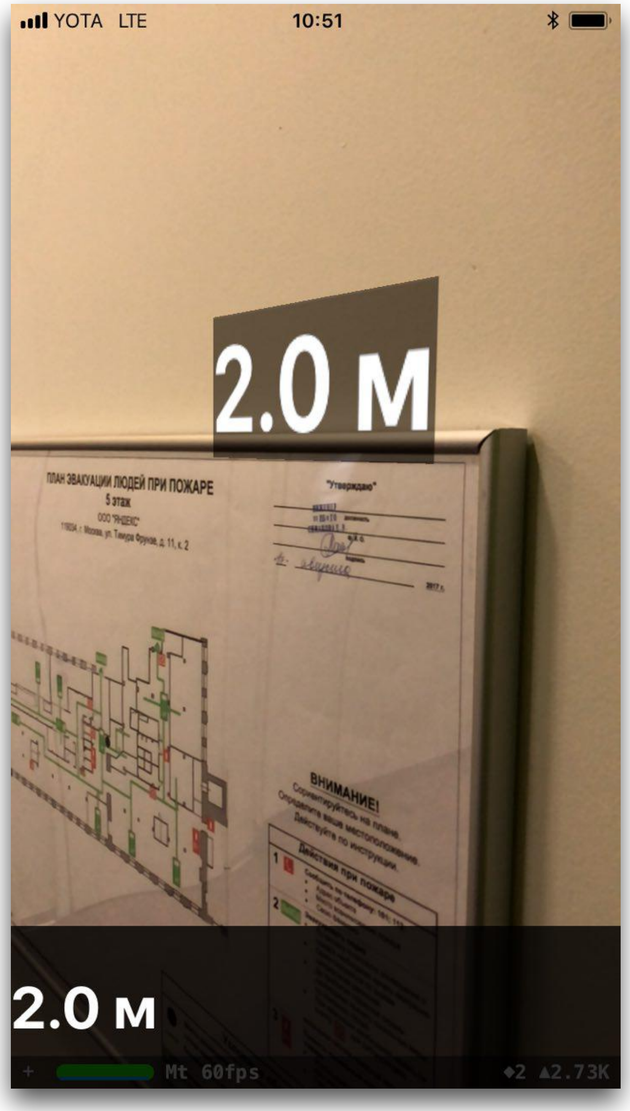
Решение в UIKit

Повернуть UILabel на угол между осью x системы координат UIKit и проекцией оси X системы координат SceneKit на плоскость экрана



`CGAffineTransform.rotated`

Проблема в SceneKit



Решение в SceneKit

> Использовать **constraints** для **SCNNode**, который должен всегда быть обращен к камере

```
let billboardConstraint = SCNBillboardConstraint()
billboardConstraint.freeAxes = SCNBillboardAxis.Y
finishNode.constraints = [billboardConstraint]
```

Решение в SceneKit

Использовать **constraints** для **SCNNode**, который должен всегда быть обращен к камере

```
let billboardConstraint = SCNBillboardConstraint()
billboardConstraint.freeAxes = SCNBillboardAxis.Y
finishNode.constraints = [billboardConstraint]
```

Решение в SceneKit

Использовать **constraints** для **SCNNode**, который должен всегда быть обращен к камере

```
let billboardConstraint = SCNBillboardConstraint()
billboardConstraint.freeAxes = SCNBillboardAxis.Y
finishNode.constraints = [billboardConstraint]
```

Результат

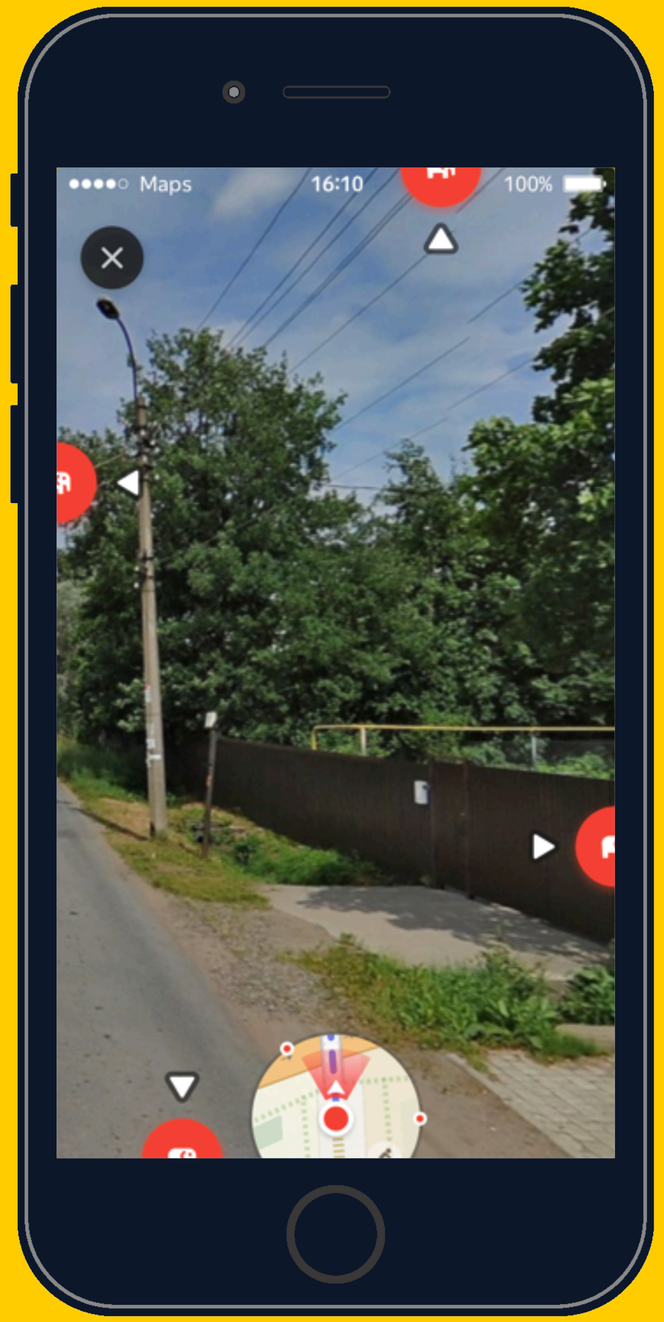
Результат

UIKit vs SceneKit

	UIKit	SceneKit
Позиция	`projectPoint`	–
Размер	–	`unprojectPoint`
Анимация	`transform.scale.xy`	`scale`
Билборд	`CGAffineTransform`	`SCNBillboardConstraint`

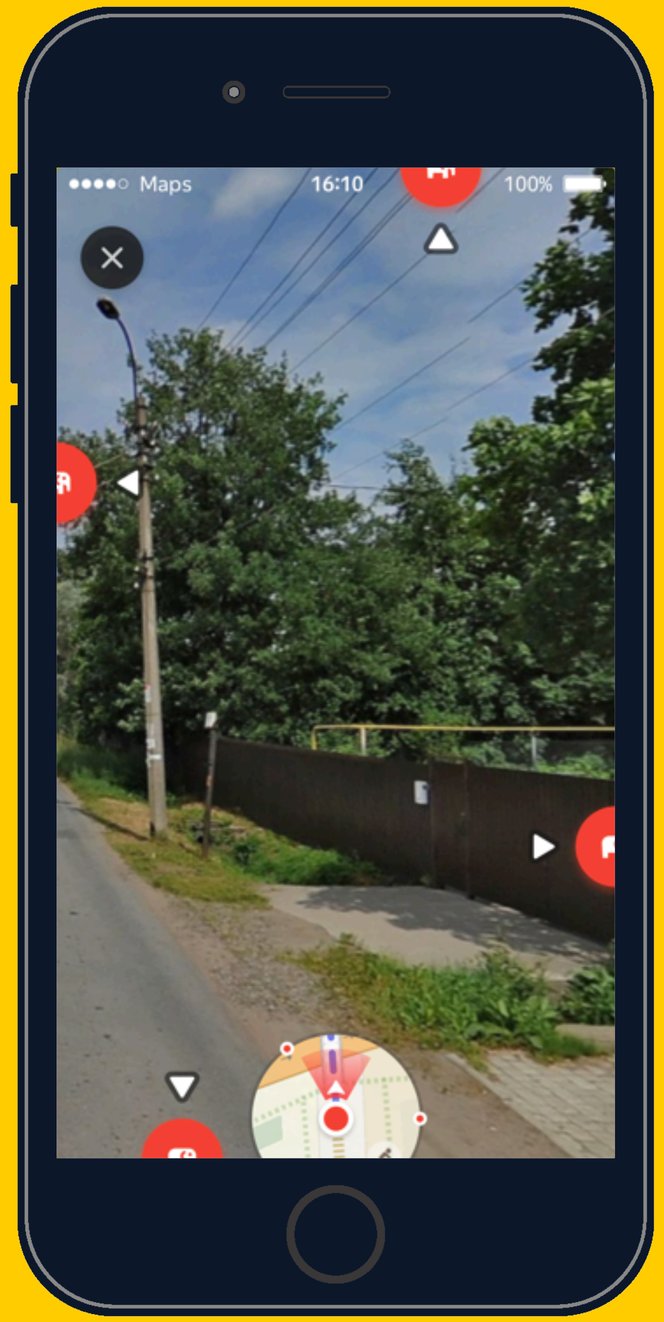
Вспомогательная метка

Вспомогательная метка



Вспомогательная метка

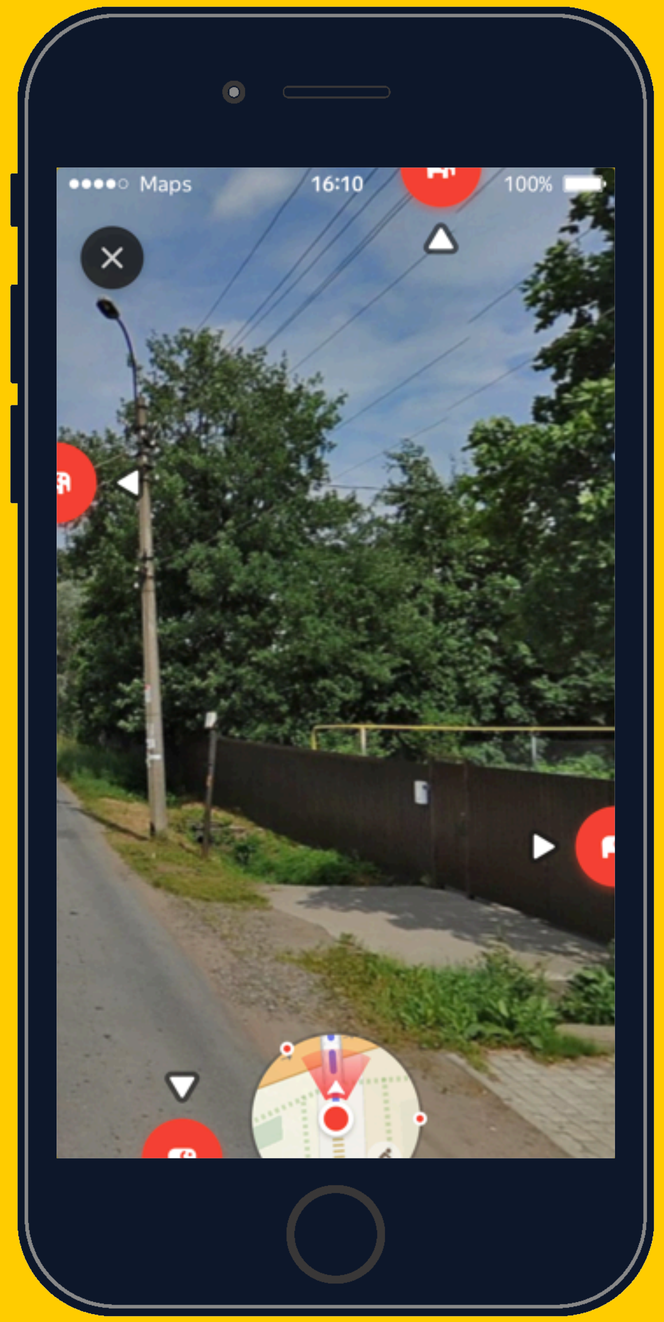
- Позиция на экране



Вспомогательная метка

Позиция на экране

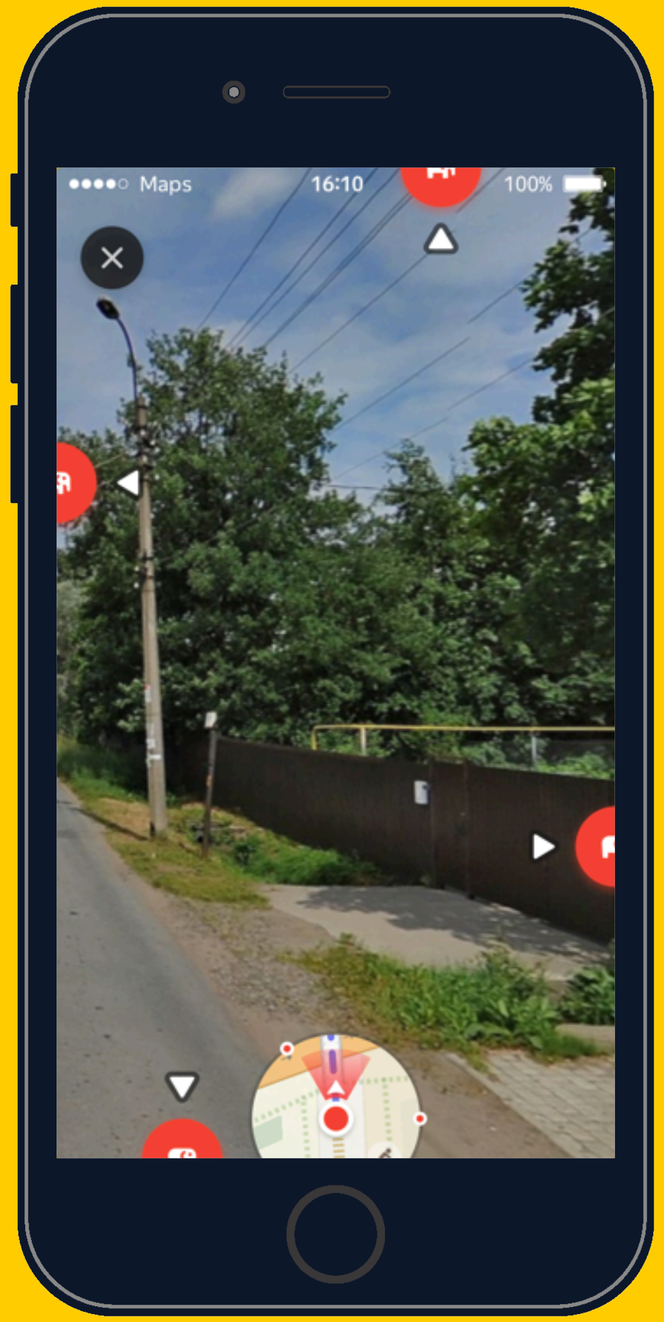
› метка показывает куда нужно развернуть телефон, чтобы увидеть финиш



Вспомогательная метка

Позиция на экране

- метка показывает куда нужно развернуть телефон, чтобы увидеть финиш
- когда финиш расположен сзади, метка «прилипает» к низу экрана

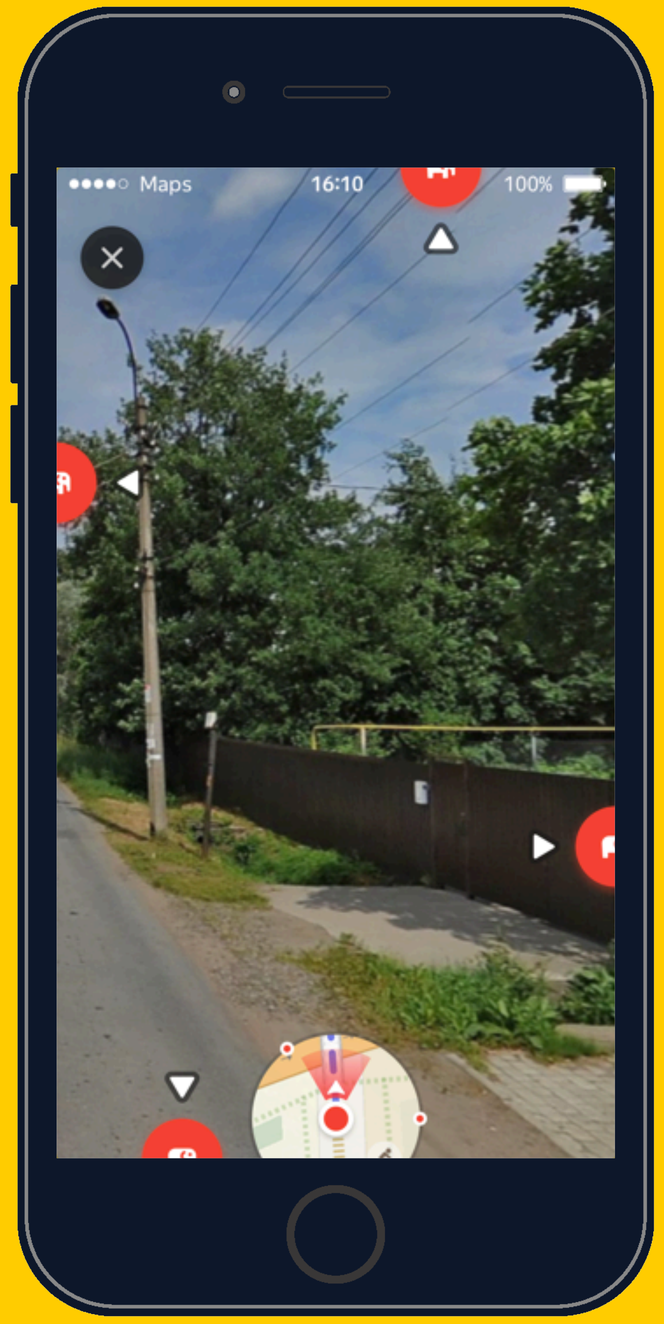


Вспомогательная метка

Позиция на экране

- метка показывает куда нужно развернуть телефон, чтобы увидеть финиш
- когда финиш расположен сзади, метка «прилипает» к низу экрана

Видимость метки



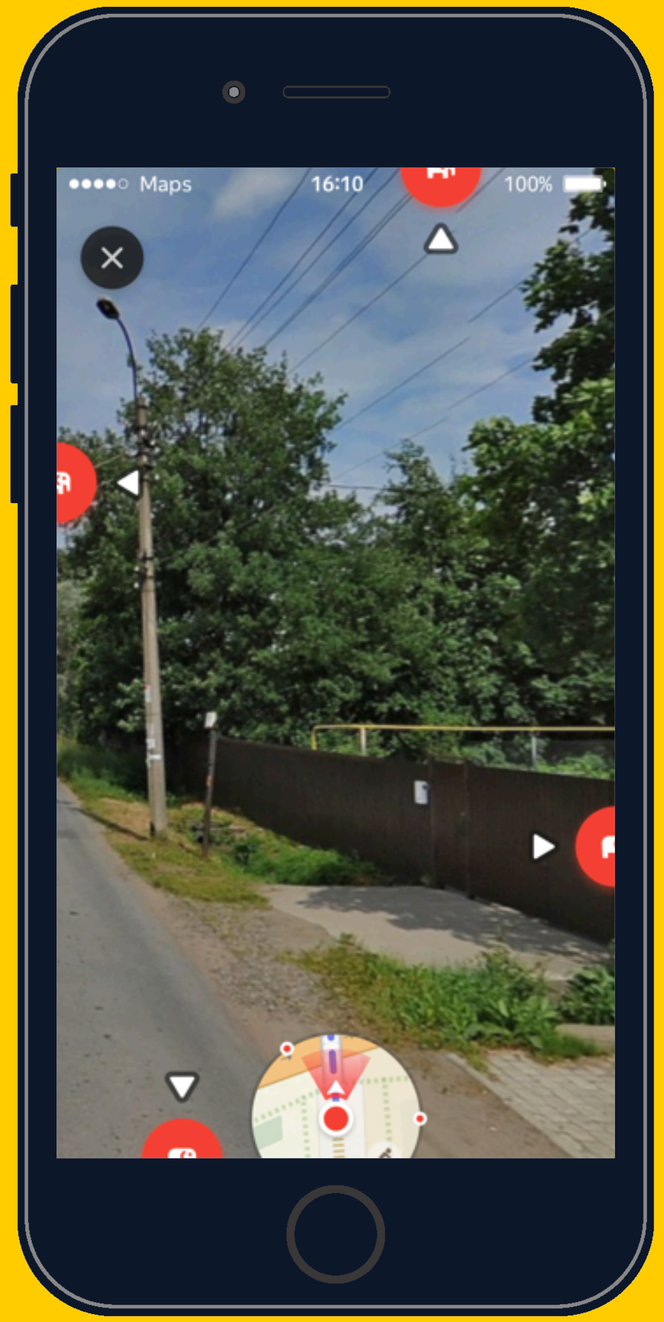
Вспомогательная метка

Позиция на экране

- метка показывает куда нужно развернуть телефон, чтобы увидеть финиш
- когда финиш расположен сзади, метка «прилипает» к низу экрана

Видимость метки

- метка исчезаем когда финиш виден на экране



ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

ВИДЕО
ФОТО
КВАДРАТ

(x,y,z)
(x,y)
ВИДЕО
ФОТО
КВАДРАТ

(x,y)

(x,y)

Проекция на плоскость экрана зависит от ориентации и положения телефона

Как узнать, что положение изменилось

Определение проекции метки финиша

- Входная точка - `SCNSceneRendererDelegate`

```
func renderer(_ renderer: SCNSceneRenderer, didRenderScene scene:
    SCNScene, atTime time: TimeInterval) {
    ...
}
```

Определение проекции метки финиша

```
func findProjection(ofNode node: SCNNode, at scnView: SCNView)
-> CGPoint {
    let nodeWorldPosition = node.worldPosition
    let projection = scnView.projectPoint(nodeWorldPosition)
    return CGPoint(x: CGFloat(projection.x),
                   y: CGFloat(projection.y))
}
```

Определение проекции метки финиша

```
func findProjection(ofNode node: SCNNode, at scnView: SCNView)
-> CGPoint {
    let nodeWorldPosition = node.worldPosition
    let projection = scnView.projectPoint(nodeWorldPosition)
    return CGPoint(x: CGFloat(projection.x),
                   y: CGFloat(projection.y))
}
```

Положение нода относительно камеры

```
func isNodeInFrontOfCamera(_ node: SCNNode, scnView: SCNView) -> Bool {
    guard let pov = scnView.pointOfView else { return false }
    guard let parent = node.parent else { return false }
    let positionInPOV = parent.convertPosition(node.position, to: pov)
    return positionInPOV.z < 0
}
```

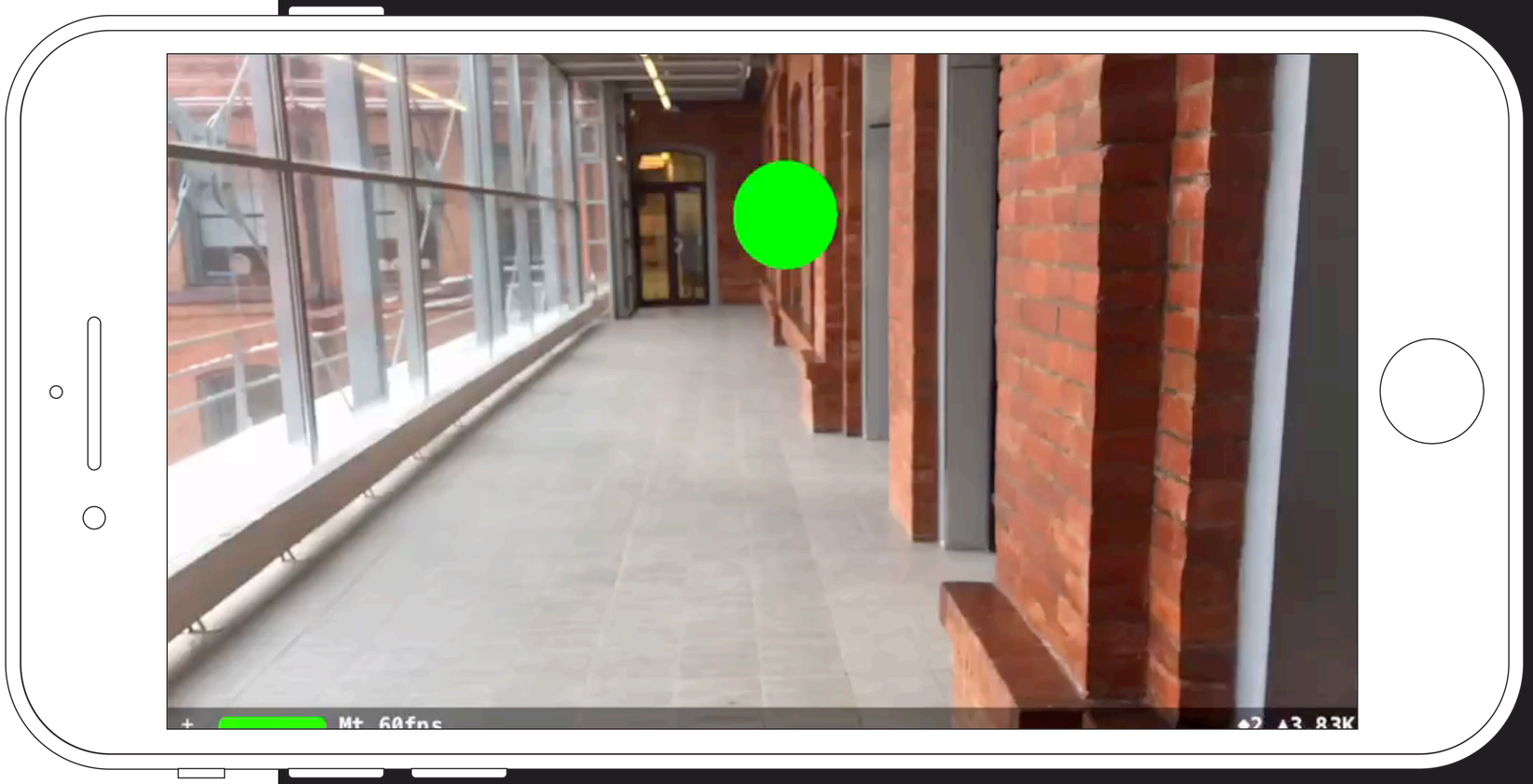
Положение нода относительно камеры

```
func isNodeInFrontOfCamera(_ node: SCNNode, scnView: SCNView) -> Bool {
    guard let pov = scnView.pointOfView else { return false }
    guard let parent = node.parent else { return false }
    let positionInPOV = parent.convertPosition(node.position, to: pov)
    return positionInPOV.z < 0
}
```

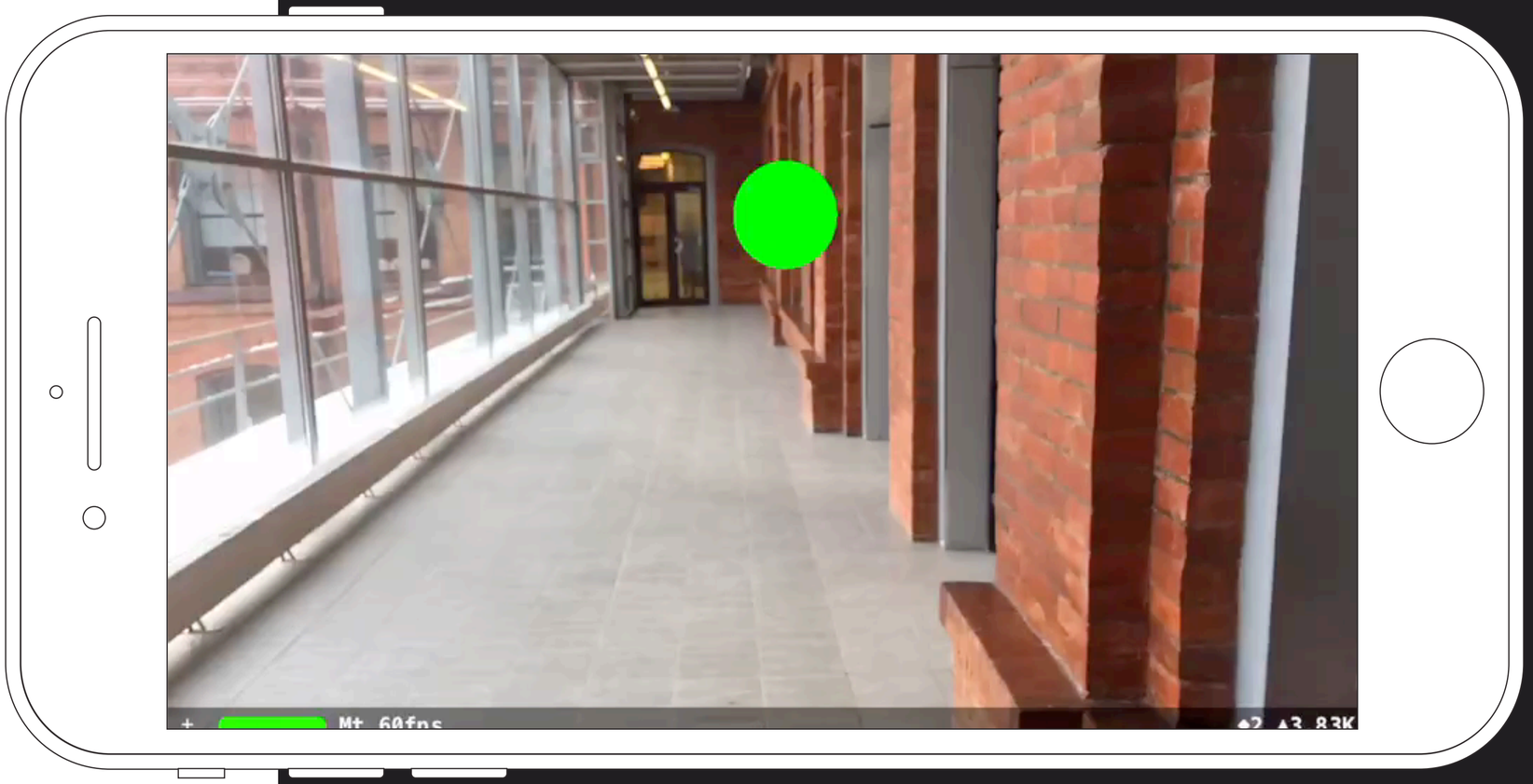
Положение нода относительно камеры

```
func isNodeInFrontOfCamera(_ node: SCNNode, scnView: SCNView) -> Bool {
    guard let pov = scnView.pointOfView else { return false }
    guard let parent = node.parent else { return false }
    let positionInPOV = parent.convertPosition(node.position, to: pov)
    return positionInPOV.z < 0
}
```

Результат



Результат



Полилиния маршрута

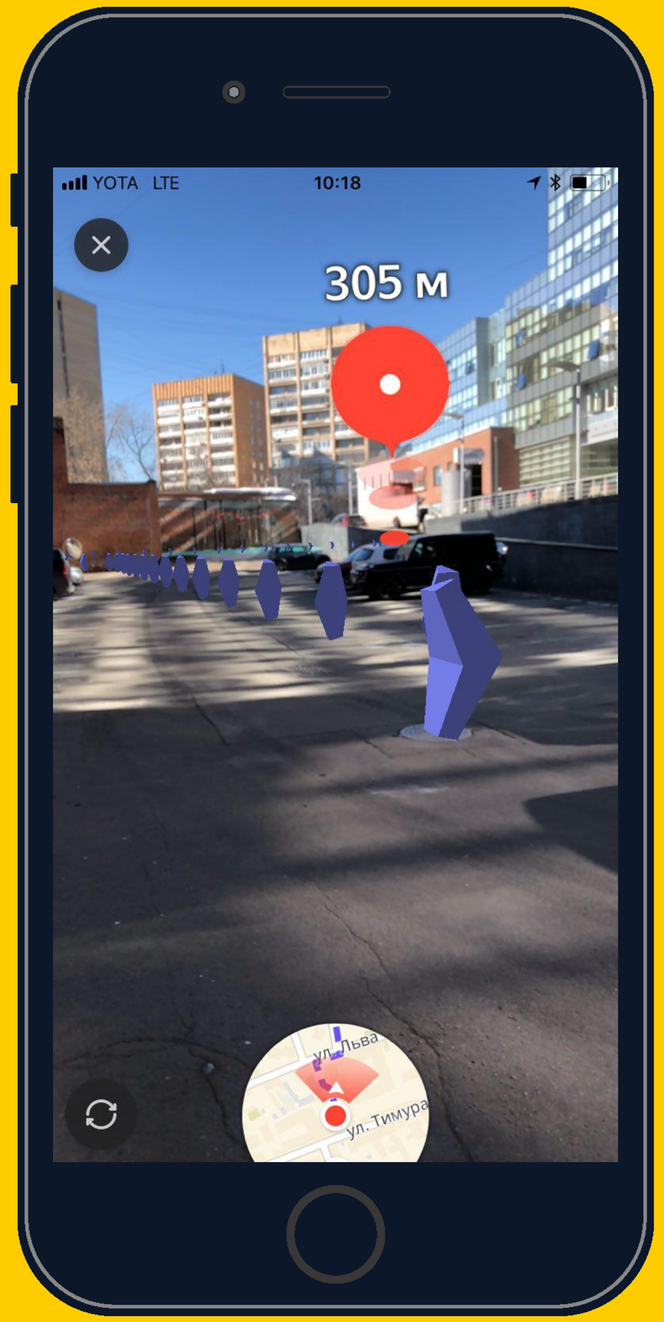
Ограничения

Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов может привести к тормозам

Полилиния маршрута

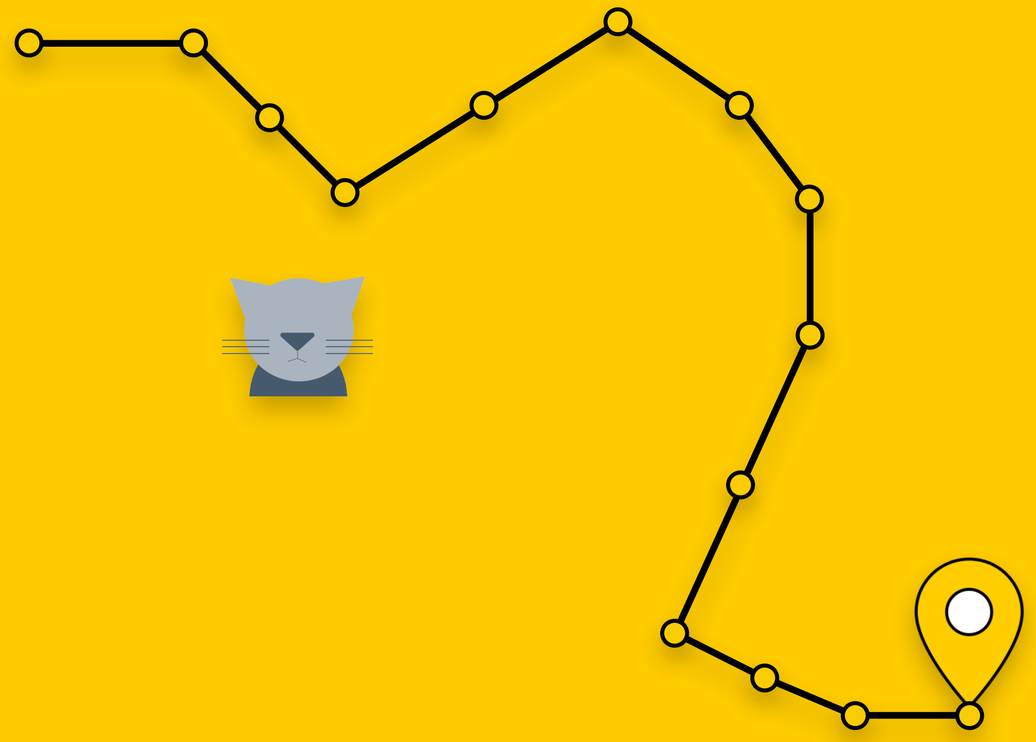
- Выбор участка маршрута, который будем рендерить
- Создание 3D моделей
- Добавление анимаций
- Обновление при движении по маршруту



Выбор точек

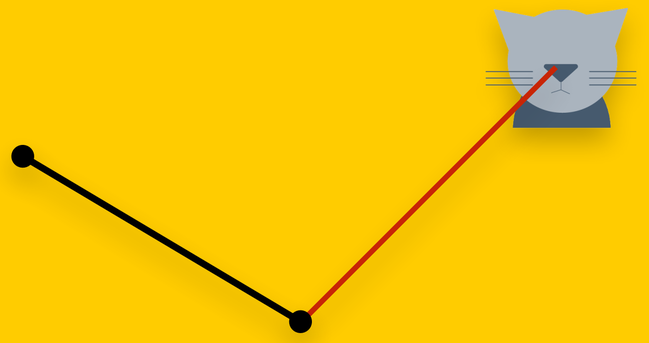
Выбор точек

- Поиск расстояний до сегментов, где расстояние это



Выбор точек

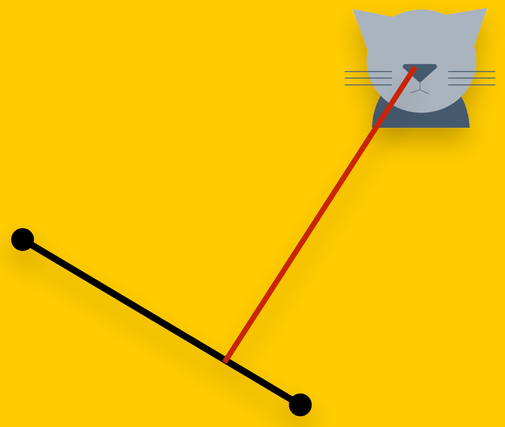
- Поиск расстояний до сегментов, где расстояние это
  - расстояние до ближайшей точки сегмента, если перпендикуляр вне отрезка



Выбор точек

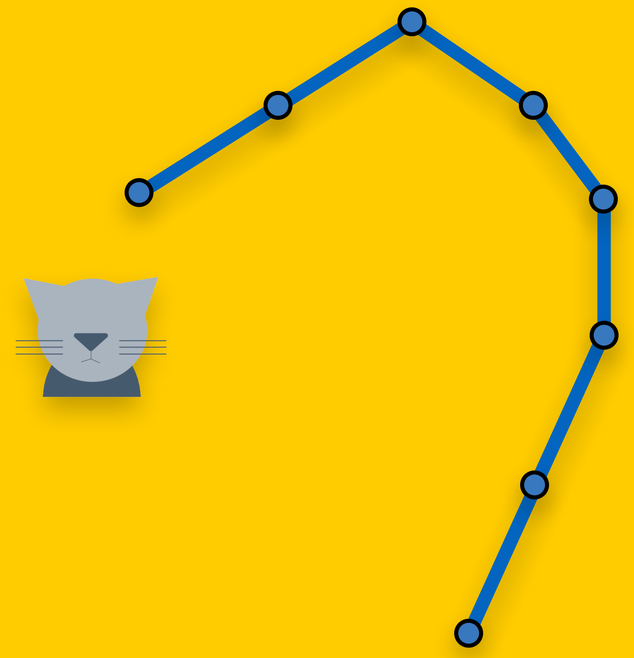
Поиск расстояний до сегментов, где расстояние это

- расстояние до ближайшей точки сегмента, если перпендикуляр вне отрезка
- перпендикуляр к сегменту, если внутри отрезка



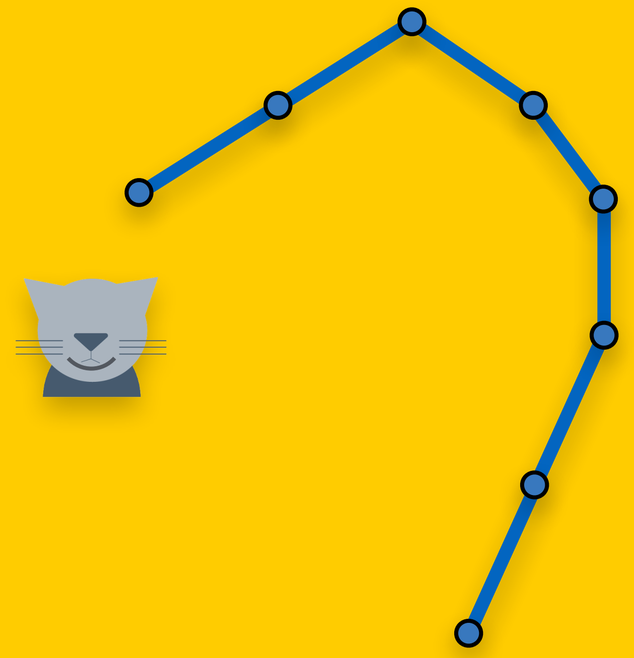
Выбор точек

- Поиск расстояний до сегментов, где расстояние это
  - расстояние до ближайшей точки сегмента, если перпендикуляр вне отрезка
  - перпендикуляр к сегменту, если внутри отрезка
- Начиная с ближайшего сегмента выбираем сегменты до тех пор, пока сумма их длин не превысит 100 м



Выбор точек

- Поиск расстояний до сегментов, где расстояние это
  - расстояние до ближайшей точки сегмента, если перпендикуляр вне отрезка
  - перпендикуляр к сегменту, если внутри отрезка
- Начиная с ближайшего сегмента выбираем сегменты до тех пор, пока сумма их длин не превысит 100 м



3D модель промежуточной точки

3D модель промежуточной точки

Просто взял и загрузил модельку

3D модель промежуточной точки

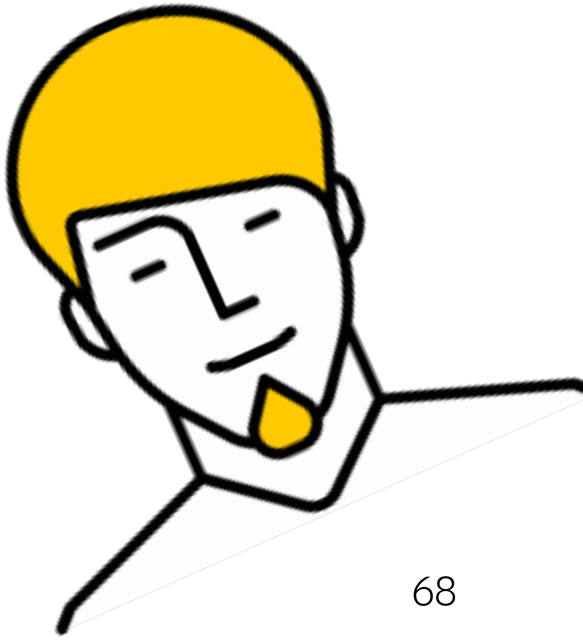
Просто взял и загрузил модельку

- тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

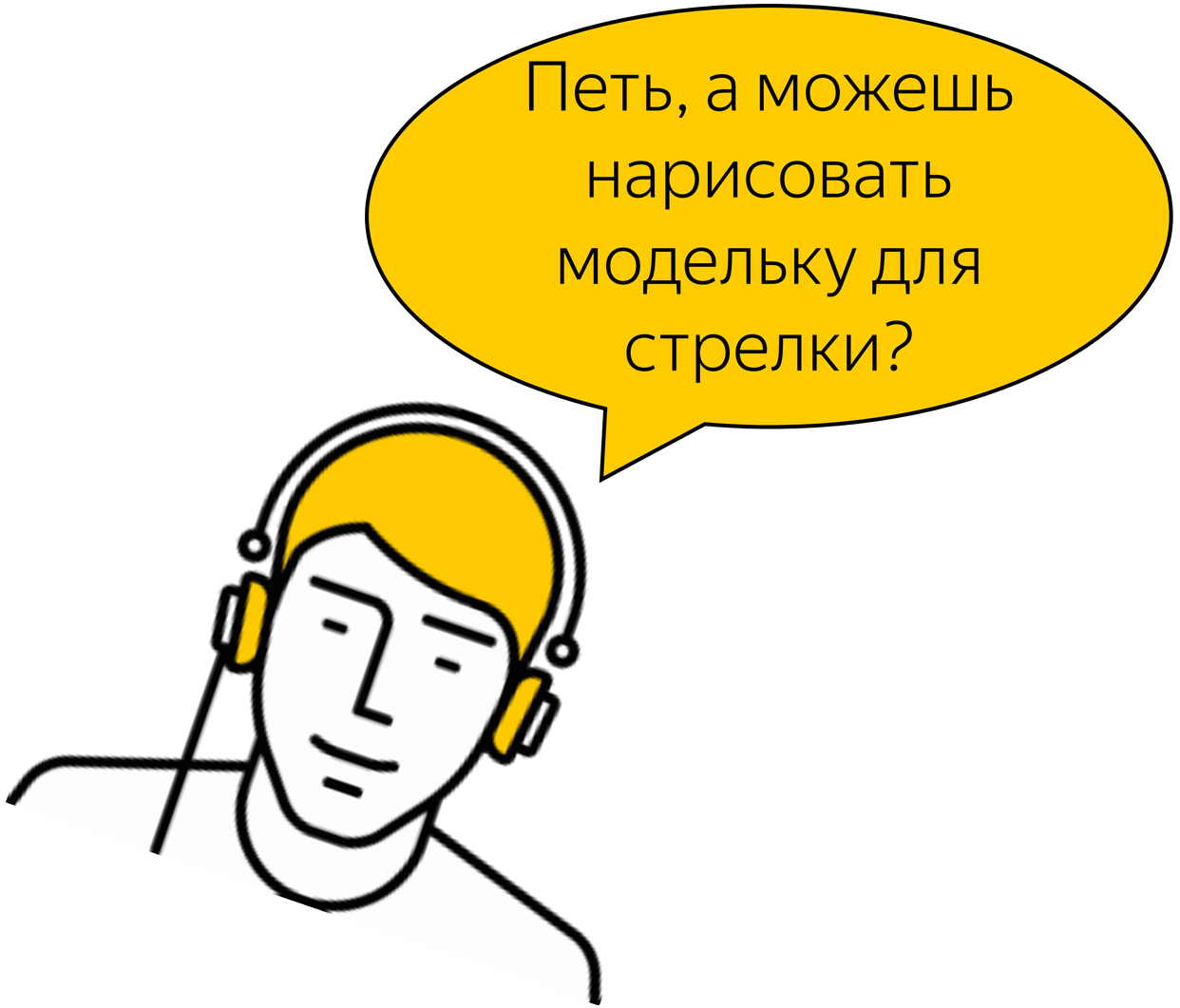
Format	Filename Extension	Supported in
Digital Asset Exchange	.dae	> macOS 10.8
Alembic	.abs	> macOS 10.10
SceneKit compressed scene	.dae or .abc	> macOS 10.10, iOS 8.0
SceneKit archive	.scn	> macOS 10.10, iOS 8.0

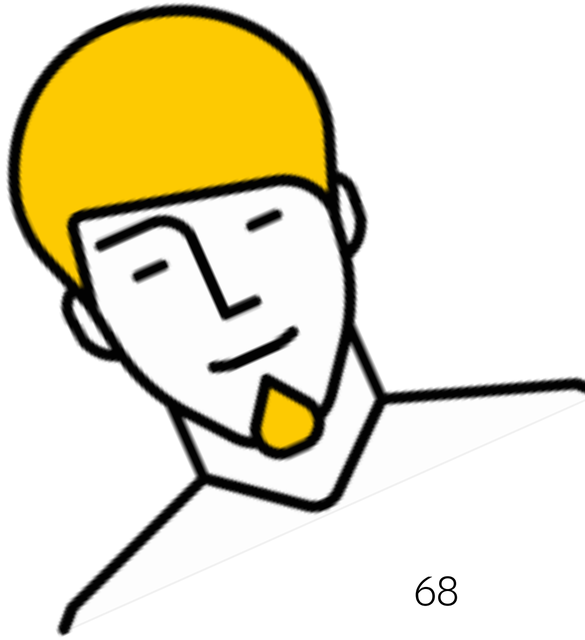
Format	Filename Extension	Supported in
Digital Asset Exchange	.dae	> macOS 10.8
Alembic	.abs	> macOS 10.10
SceneKit compressed scene	.dae or .abc	> macOS 10.10, iOS 8.0
SceneKit archive	.scn	> macOS 10.10, iOS 8.0

Создание 3D модели



Создание 3D модели





Создание 3D модели

3D модель промежуточной точки

Просто взял и загрузил модельку

› тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

3D модель промежуточной точки

Просто взял и загрузил модельку

› тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

Пришлось все делать самим

3D модель промежуточной точки

Просто взял и загрузил модельку

› тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

Пришлось все делать самим

› представить 3D модель в виде треугольников

3D модель промежуточной точки

Просто взял и загрузил модельку

- тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

Пришлось все делать самим

- представить 3D модель в виде треугольников
- найти координаты вершин треугольников в локальной системе координат модели

3D модель промежуточной точки

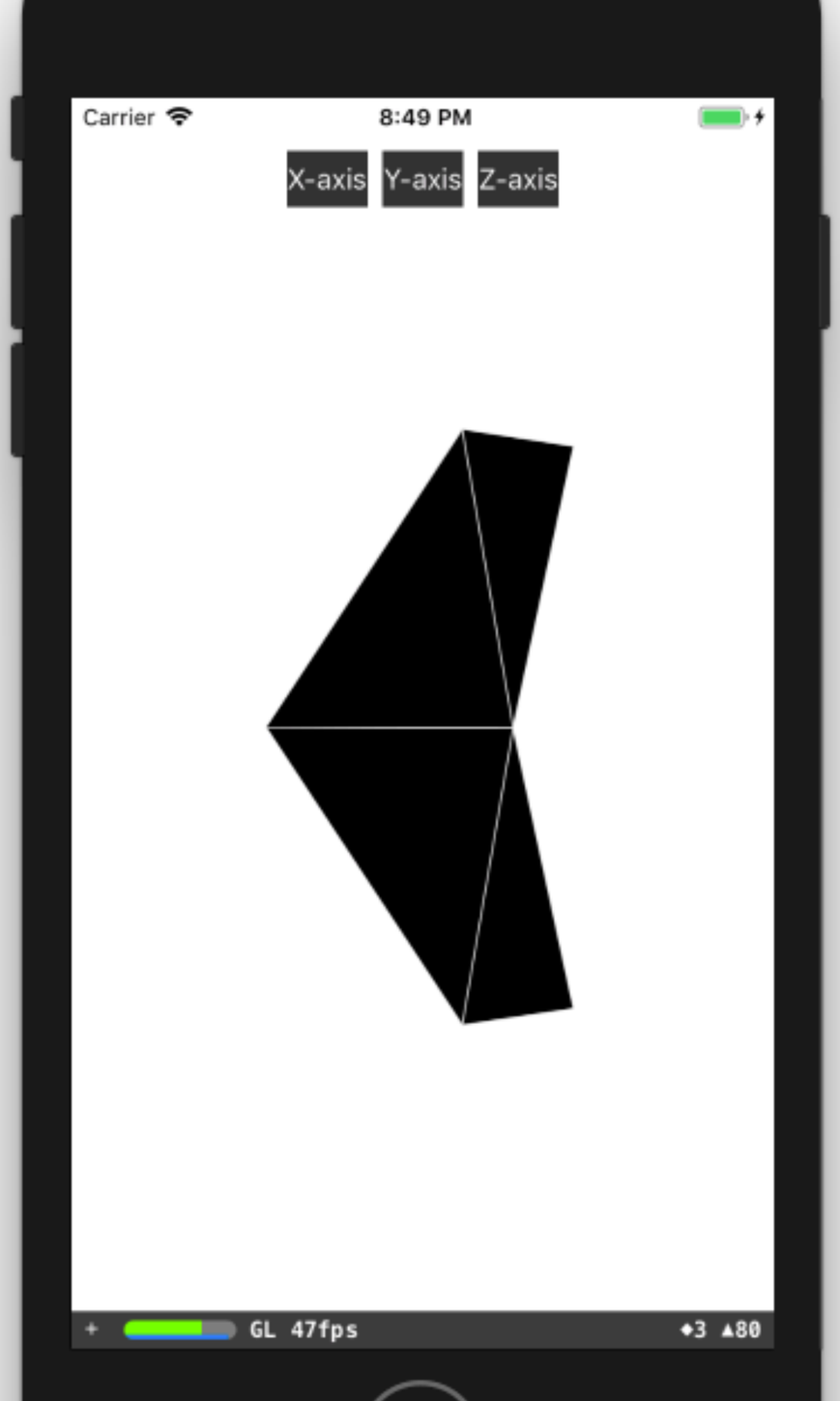
Просто взял и загрузил модельку

- тем более что `SceneKit` поддерживает большинство популярных форматов загрузки сцен и моделей

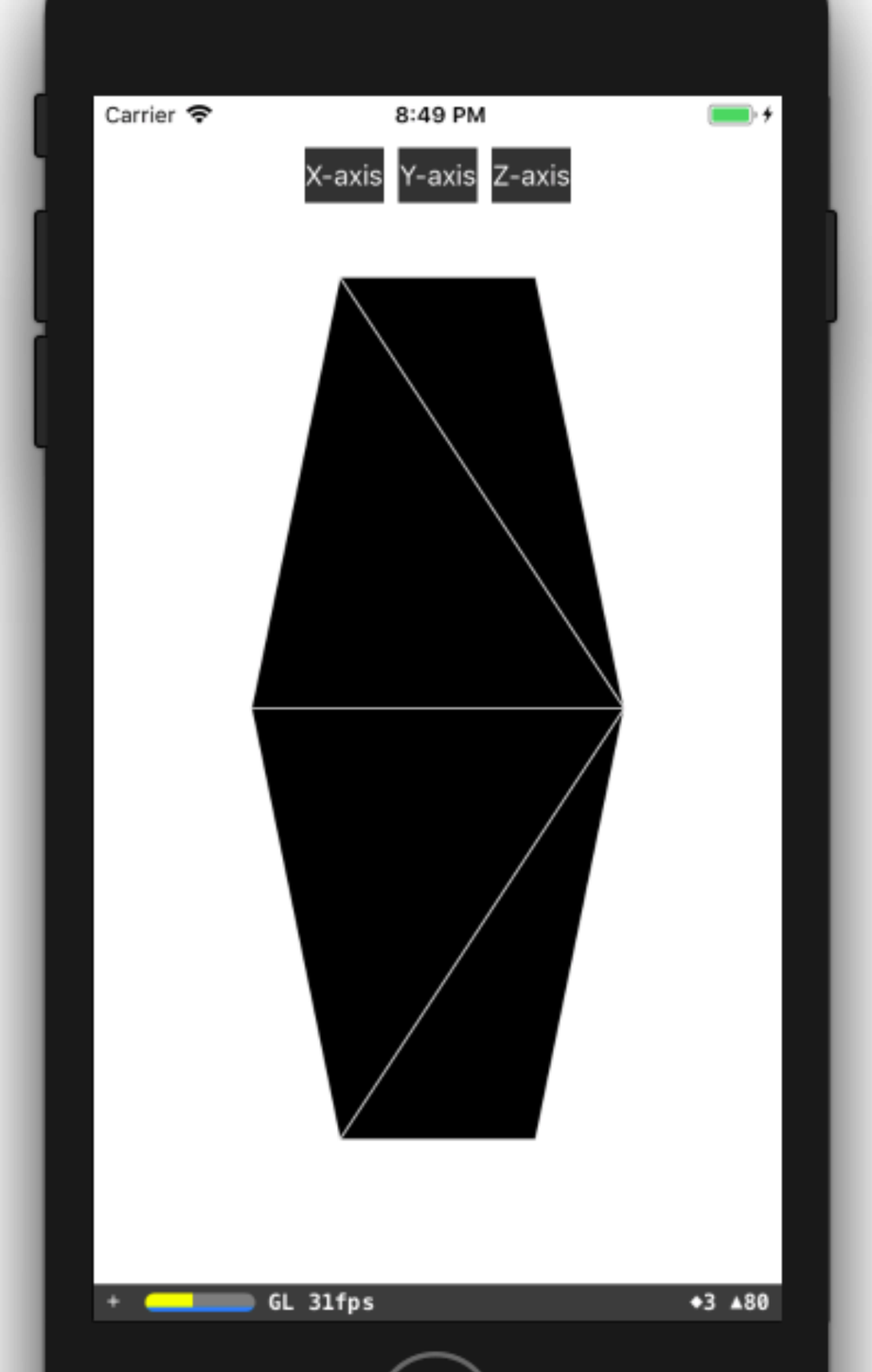
Пришлось все делать самим

- представить 3D модель в виде треугольников
- найти координаты вершин треугольников в локальной системе координат модели
- создать массив индексов для вершин треугольников

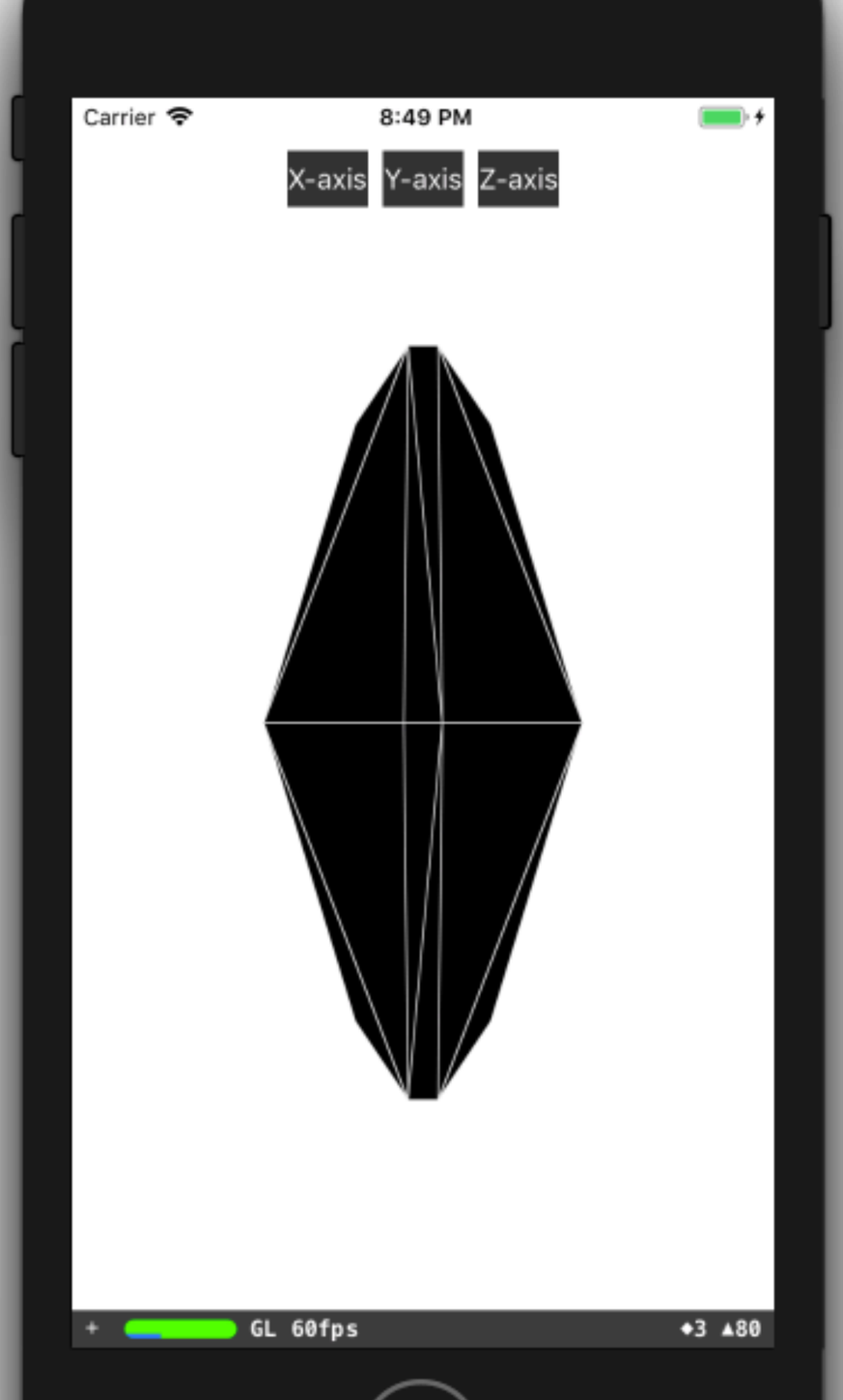
```
let vertices = [
  SCNVector3Make(-0.02,  0.00,  0.00), // 0
  SCNVector3Make(-0.02,  0.50, -0.33), // 1
  SCNVector3Make(-0.10,  0.44, -0.50), // 2
  SCNVector3Make(-0.22,  0.00, -0.39), // 3
  SCNVector3Make(-0.10, -0.44, -0.50), // 4
  SCNVector3Make(-0.02, -0.50, -0.33), // 5

  SCNVector3Make( 0.02,  0.00,  0.00), // 6
  SCNVector3Make( 0.02,  0.50, -0.33), // 7
  SCNVector3Make( 0.10,  0.44, -0.50), // 8
  SCNVector3Make( 0.22,  0.00, -0.39), // 9
  SCNVector3Make( 0.10, -0.44, -0.50), // 10
  SCNVector3Make( 0.02, -0.50, -0.33), // 11
]

let indices = [
  0,3,5, 3,4,5, 1,2,3, 0,1,3,
  10,9,11, 6,11,9, 6,9,7, 9,8,7,
  6,5,11, 6,0,5, 6,1,0, 6,7,1,
  11,5,4, 11,4,10, 9,4,3, 9,10,4,
  9,3,2, 9,2,8, 8,2,1, 8,1,7
]
```



```
let vertices = [
  SCNVector3Make(-0.02,  0.00,  0.00), // 0
  SCNVector3Make(-0.02,  0.50, -0.33), // 1
  SCNVector3Make(-0.10,  0.44, -0.50), // 2
  SCNVector3Make(-0.22,  0.00, -0.39), // 3
  SCNVector3Make(-0.10, -0.44, -0.50), // 4
  SCNVector3Make(-0.02, -0.50, -0.33), // 5

  SCNVector3Make( 0.02,  0.00,  0.00), // 6
  SCNVector3Make( 0.02,  0.50, -0.33), // 7
  SCNVector3Make( 0.10,  0.44, -0.50), // 8
  SCNVector3Make( 0.22,  0.00, -0.39), // 9
  SCNVector3Make( 0.10, -0.44, -0.50), // 10
  SCNVector3Make( 0.02, -0.50, -0.33), // 11
]

let indices = [
  0,3,5, 3,4,5, 1,2,3, 0,1,3,
  10,9,11, 6,11,9, 6,9,7, 9,8,7,
  6,5,11, 6,0,5, 6,1,0, 6,7,1,
  11,5,4, 11,4,10, 9,4,3, 9,10,4,
  9,3,2, 9,2,8, 8,2,1, 8,1,7
]
```



```swift
let vertices = [
  SCNVector3Make(-0.02,  0.00,  0.00), // 0
  SCNVector3Make(-0.02,  0.50, -0.33), // 1
  SCNVector3Make(-0.10,  0.44, -0.50), // 2
  SCNVector3Make(-0.22,  0.00, -0.39), // 3
  SCNVector3Make(-0.10, -0.44, -0.50), // 4
  SCNVector3Make(-0.02, -0.50, -0.33), // 5

  SCNVector3Make( 0.02,  0.00,  0.00), // 6
  SCNVector3Make( 0.02,  0.50, -0.33), // 7
  SCNVector3Make( 0.10,  0.44, -0.50), // 8
  SCNVector3Make( 0.22,  0.00, -0.39), // 9
  SCNVector3Make( 0.10, -0.44, -0.50), // 10
  SCNVector3Make( 0.02, -0.50, -0.33), // 11
]

let indices = [
  0,3,5, 3,4,5, 1,2,3, 0,1,3,
  10,9,11, 6,11,9, 6,9,7, 9,8,7,
  6,5,11, 6,0,5, 6,1,0, 6,7,1,
  11,5,4, 11,4,10, 9,4,3, 9,10,4,
  9,3,2, 9,2,8, 8,2,1, 8,1,7
]
```



Модель стрелки

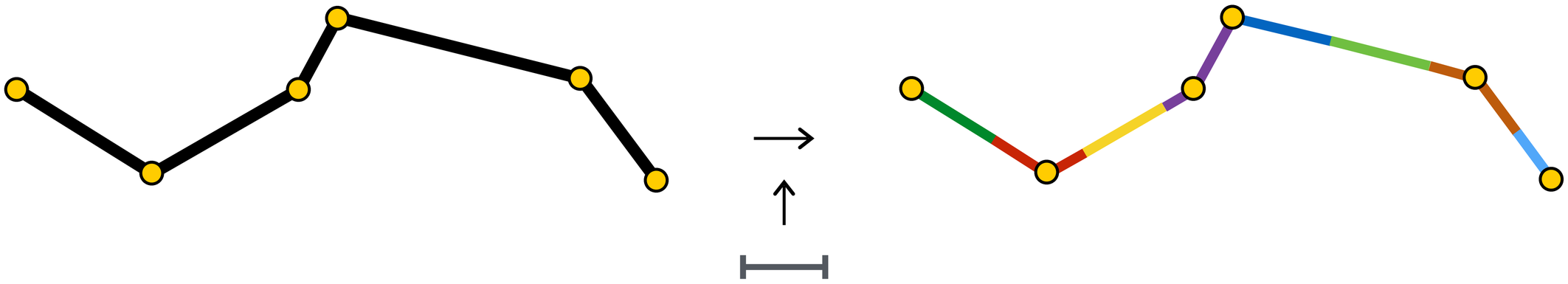
Модель стрелки

Анимация линии маршрута

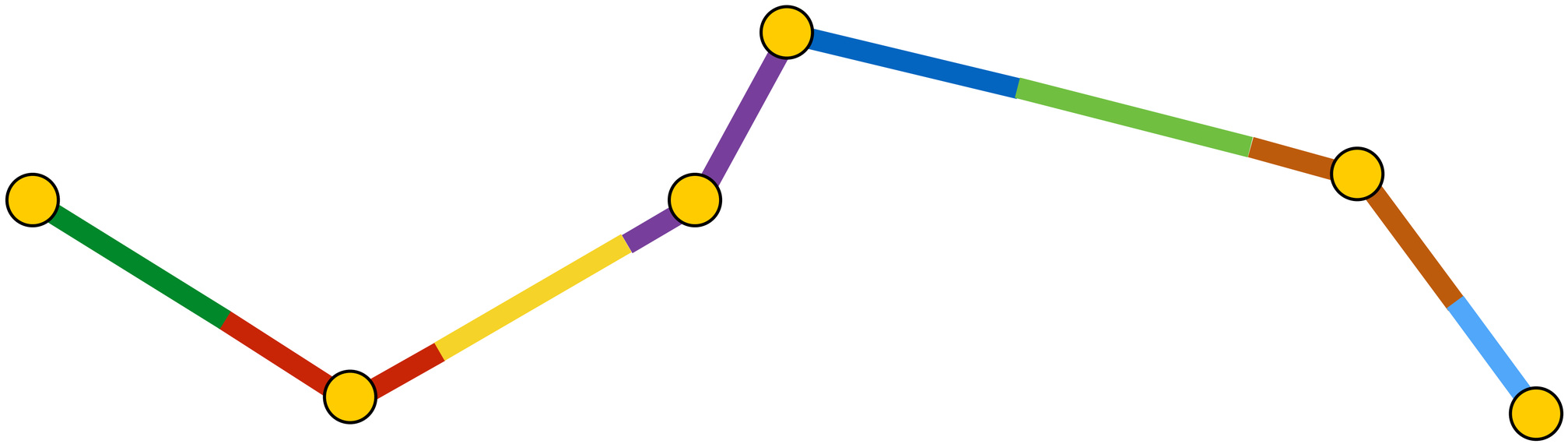
1. Разделение маршрута на участки равной длины*
2. Создание стрелки для каждого участка
3. Добавление каждой стрелки анимации движения вдоль участка

* Расстояние между стрелками в процессе анимации

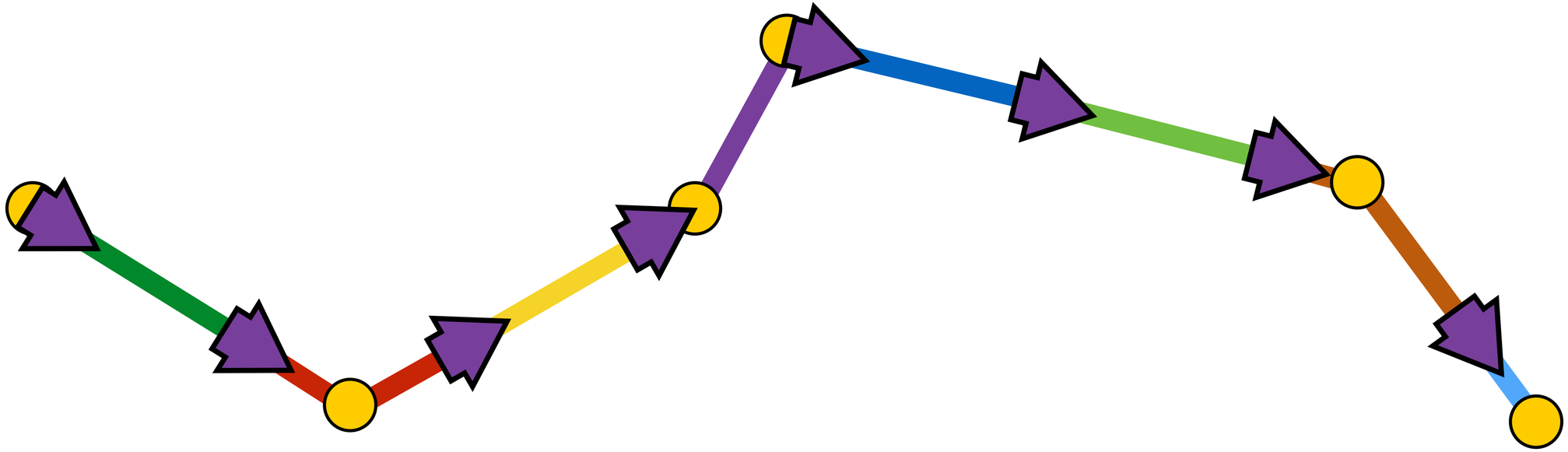
Разбиение маршрута на участки



Начальное положение



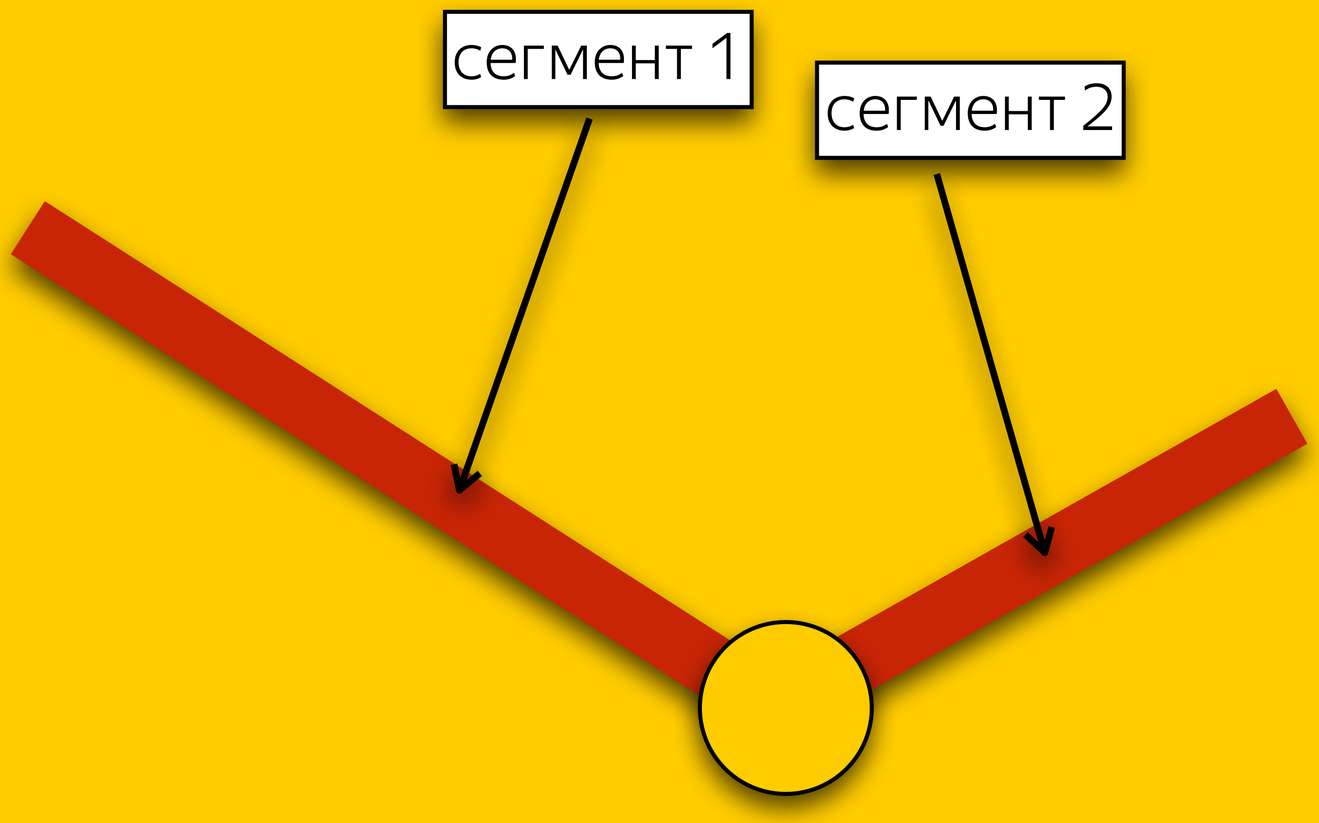
Начальное положение



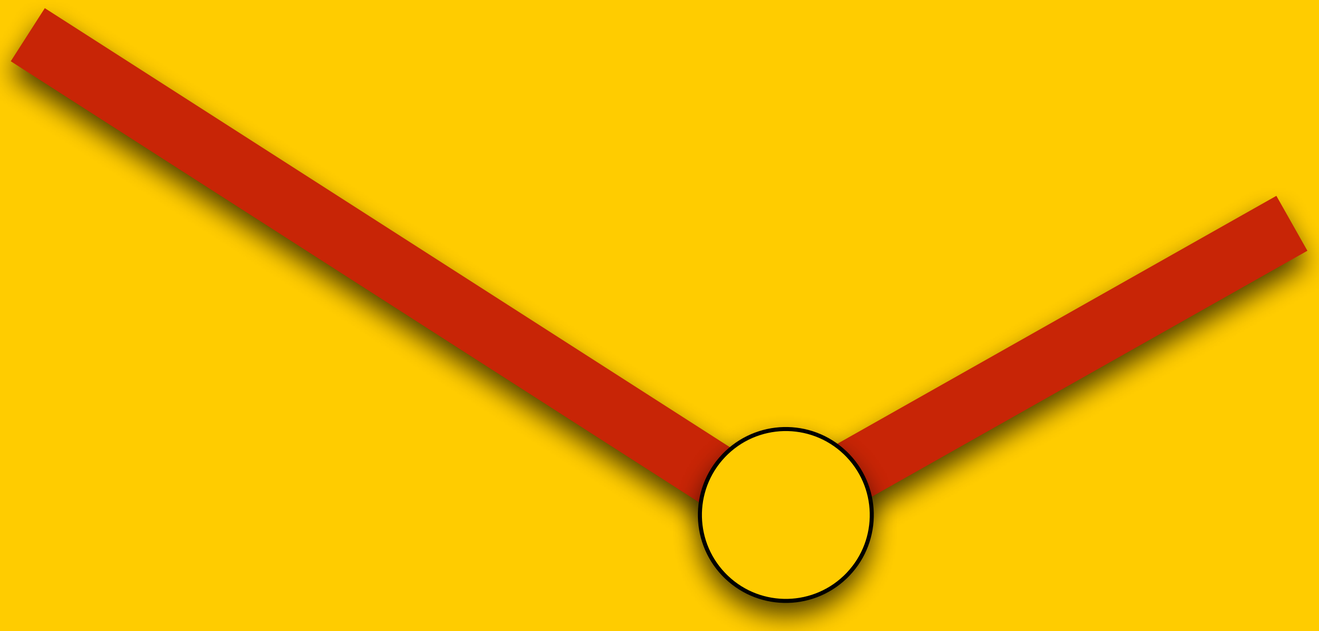
Модель анимации

```
struct Animation {
    struct Step {
        let start: CGPoint
        let end: CGPoint

        init(start: CGPoint, end: CGPoint) {
            self.start = start
            self.end = end
        }
    }

    var steps: [Step]
}
```

Анимация стрелки

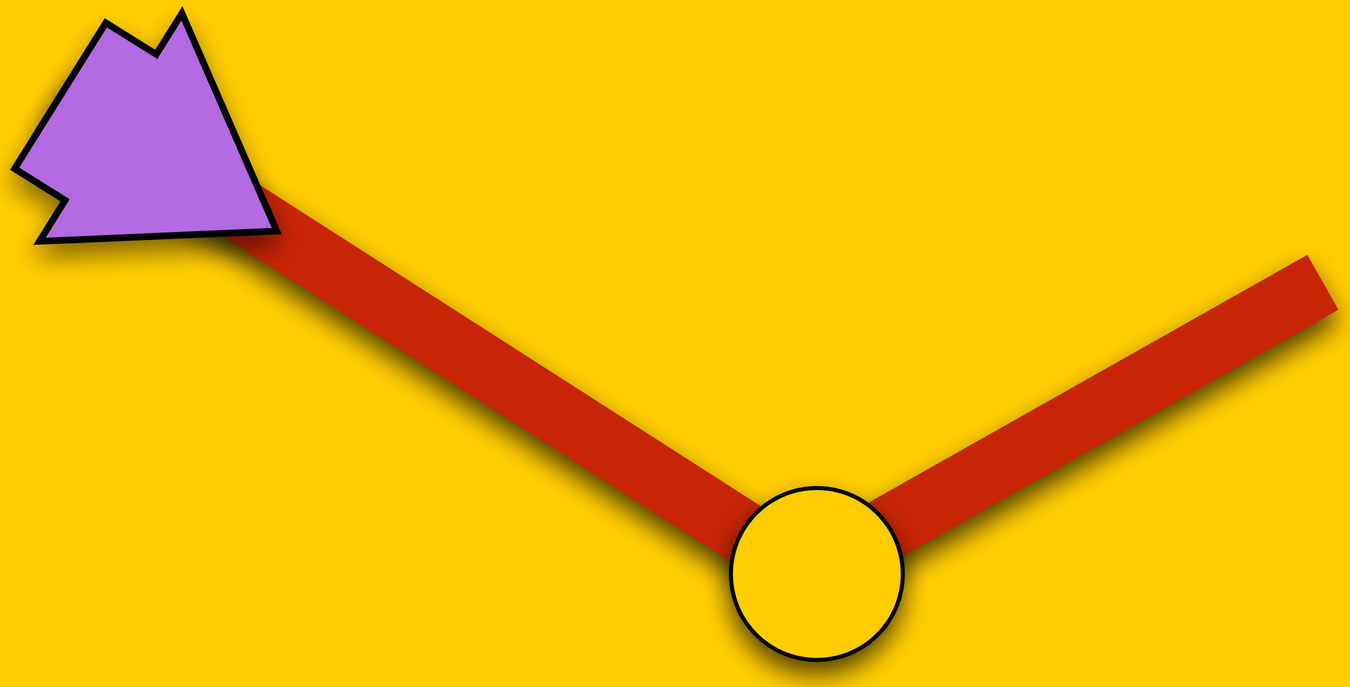


Анимация стрелки



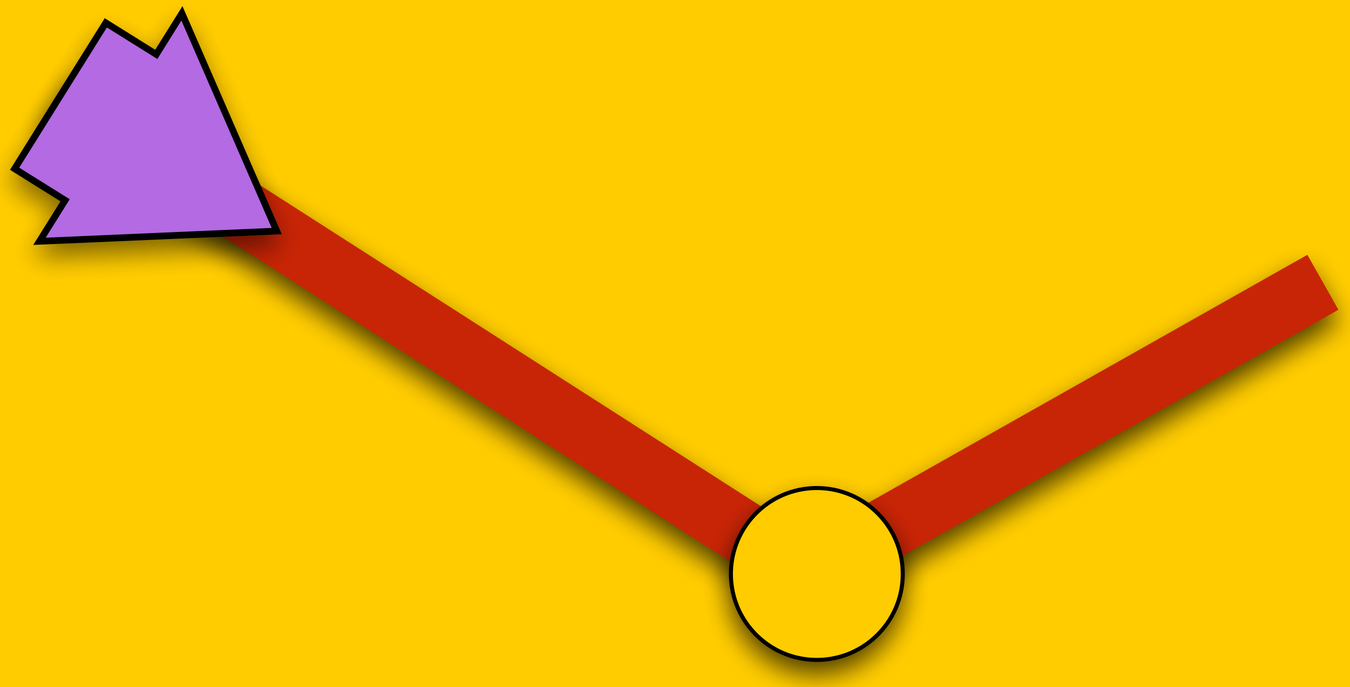
Анимация стрелки

- появление в начальной позиции с начальным углом



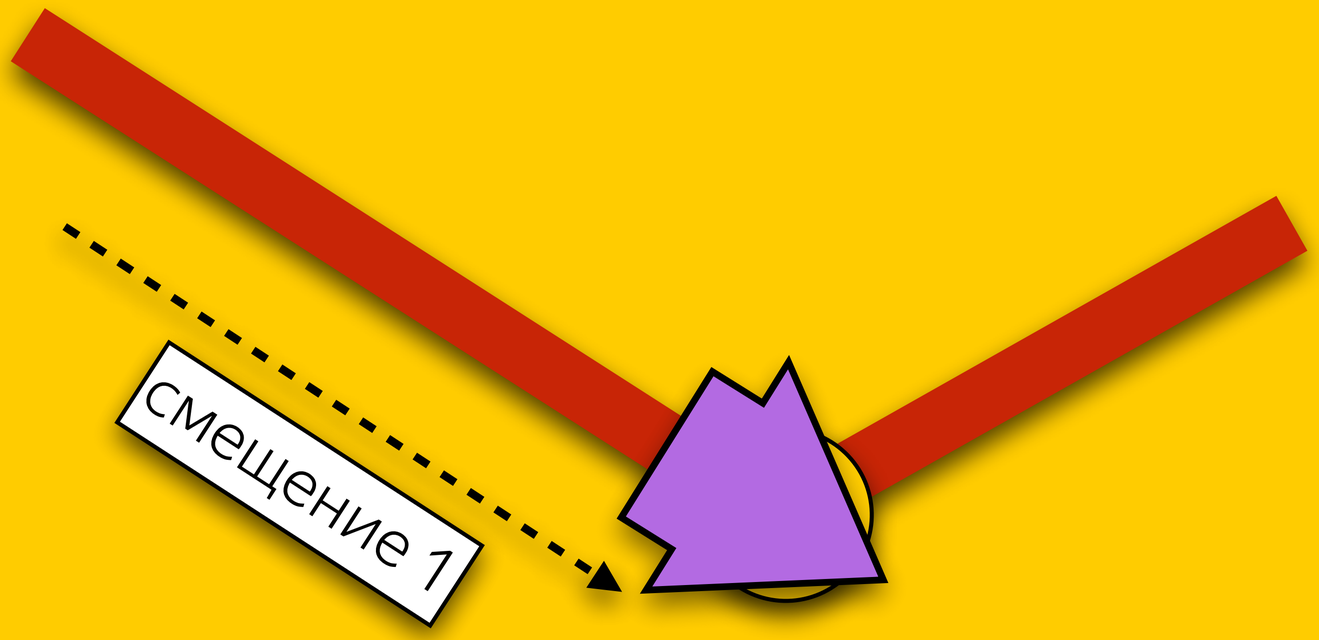
Анимация стрелки

- появление в начальной позиции с начальным углом
- последовательность смещений вдоль сегментов с поворотами в точках соединения сегментов



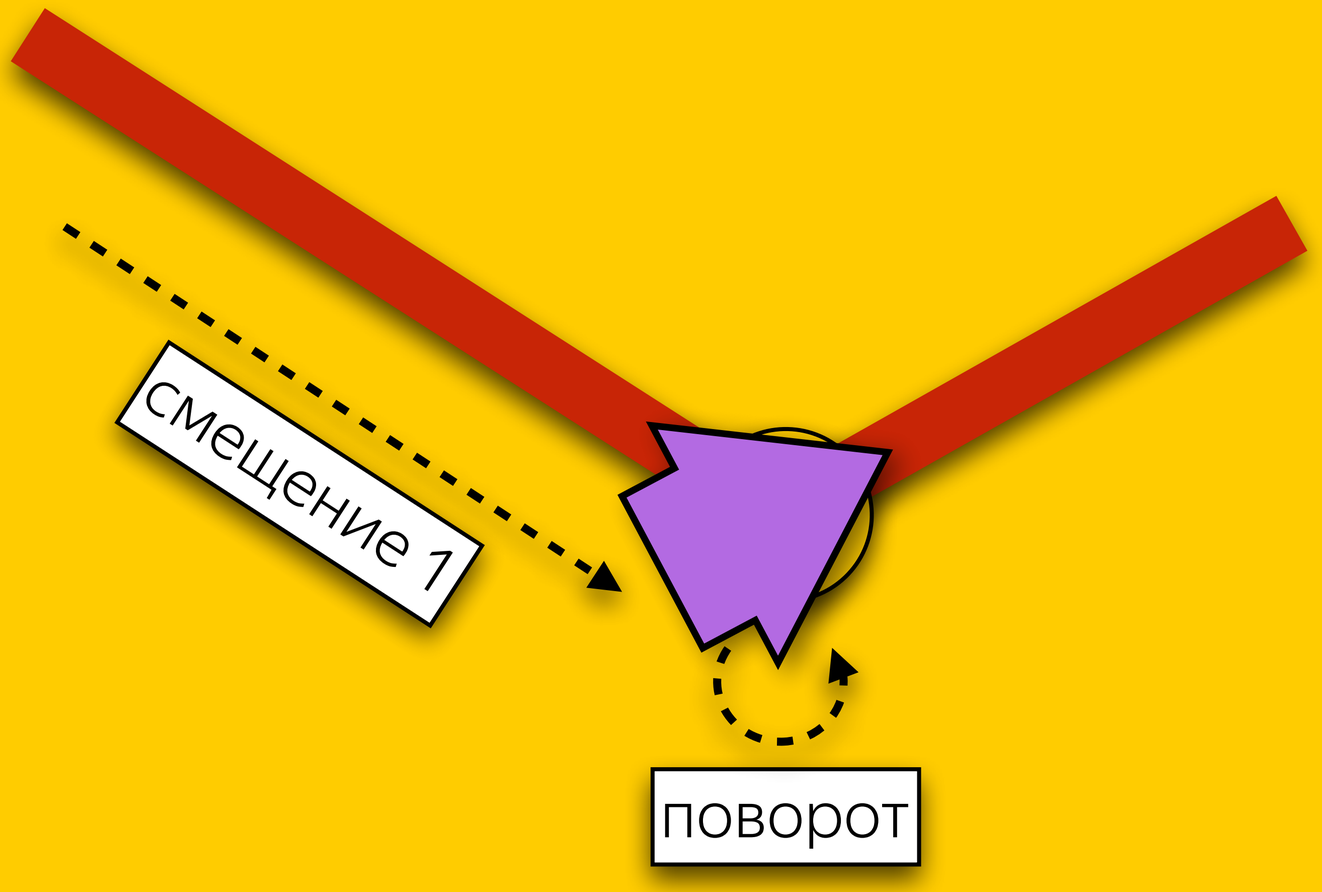
Анимация стрелки

- появление в начальной позиции с начальным углом
- последовательность смещений вдоль сегментов с поворотами в точках соединения сегментов



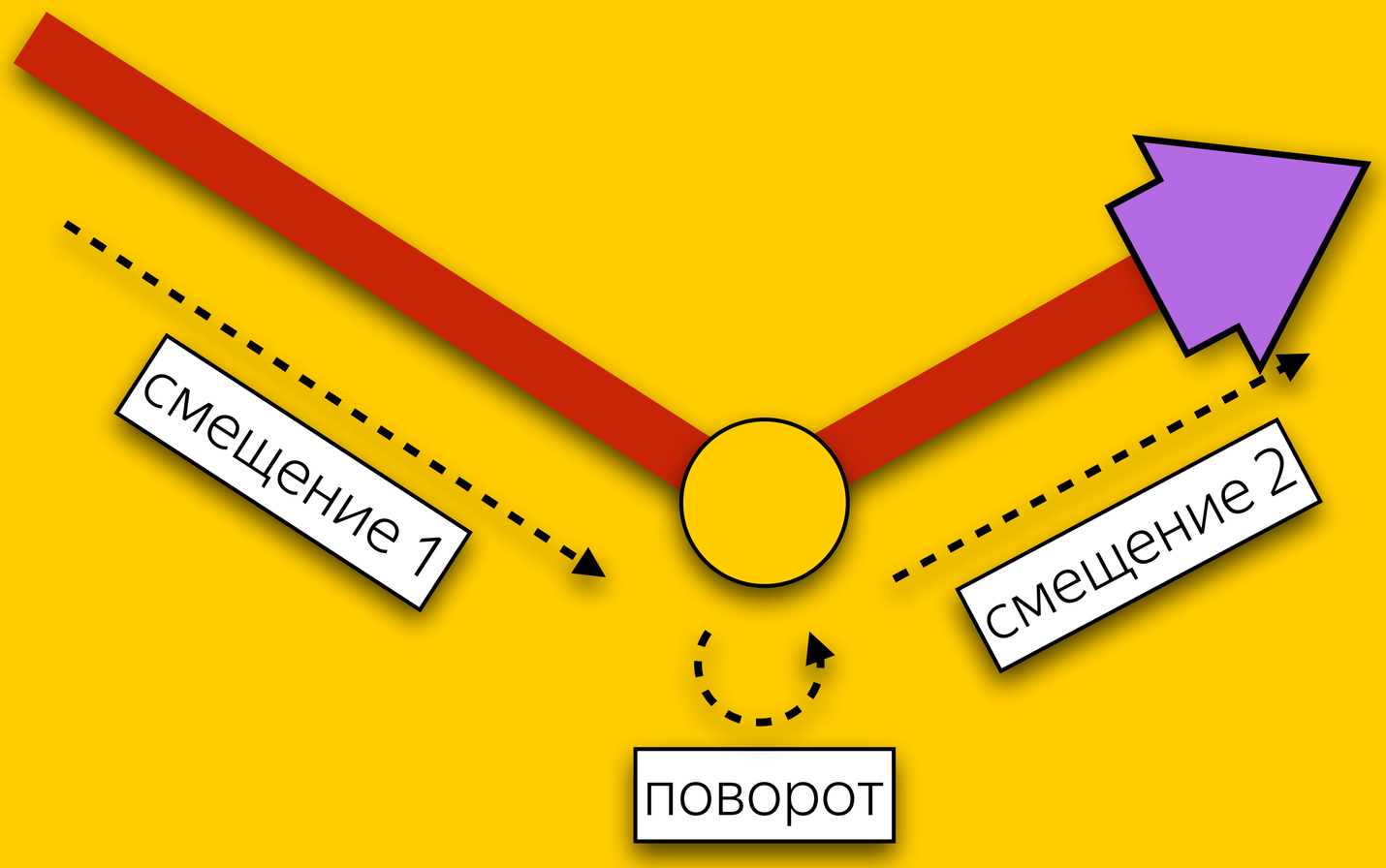
Анимация стрелки

- появление в начальной позиции с начальным углом
- последовательность смещений вдоль сегментов с поворотами в точках соединения сегментов



Анимация стрелки

- появление в начальной позиции с начальным углом
- последовательность смещений вдоль сегментов с поворотами в точках соединения сегментов



SCNAction

- объект анимации
- может включать дочерние элементы
  - *Sequence* - дочерние экшены запускаются последовательно
  - *Group*- дочерние экшены запускаются параллельно
  - *Repeating* - дочерний экшен перезапускается после окончания

Итоговая анимация

```
let reset = createResetAction(firstStep: firstStep)
var stepActions: [SCNAction] = []
for stepIndex in 0..<animation.steps.count {
    let step = animation.steps[stepIndex]
    let stepAction = createAction(forStep: step, ...)
    stepActions.append(stepAction)
}
arrow.runAction(SCNAction.repeatForever(
    SCNAction.sequence([reset] + stepActions))
)
```

Итоговая анимация

```
let reset = createResetAction(firstStep: firstStep)
var stepActions: [SCNAction] = []
for stepIndex in 0..<animation.steps.count {
    let step = animation.steps[stepIndex]
    let stepAction = createAction(forStep: step, ...)
    stepActions.append(stepAction)
}
arrow.runAction(SCNAction.repeatForever(
    SCNAction.sequence([reset] + stepActions))
)
```

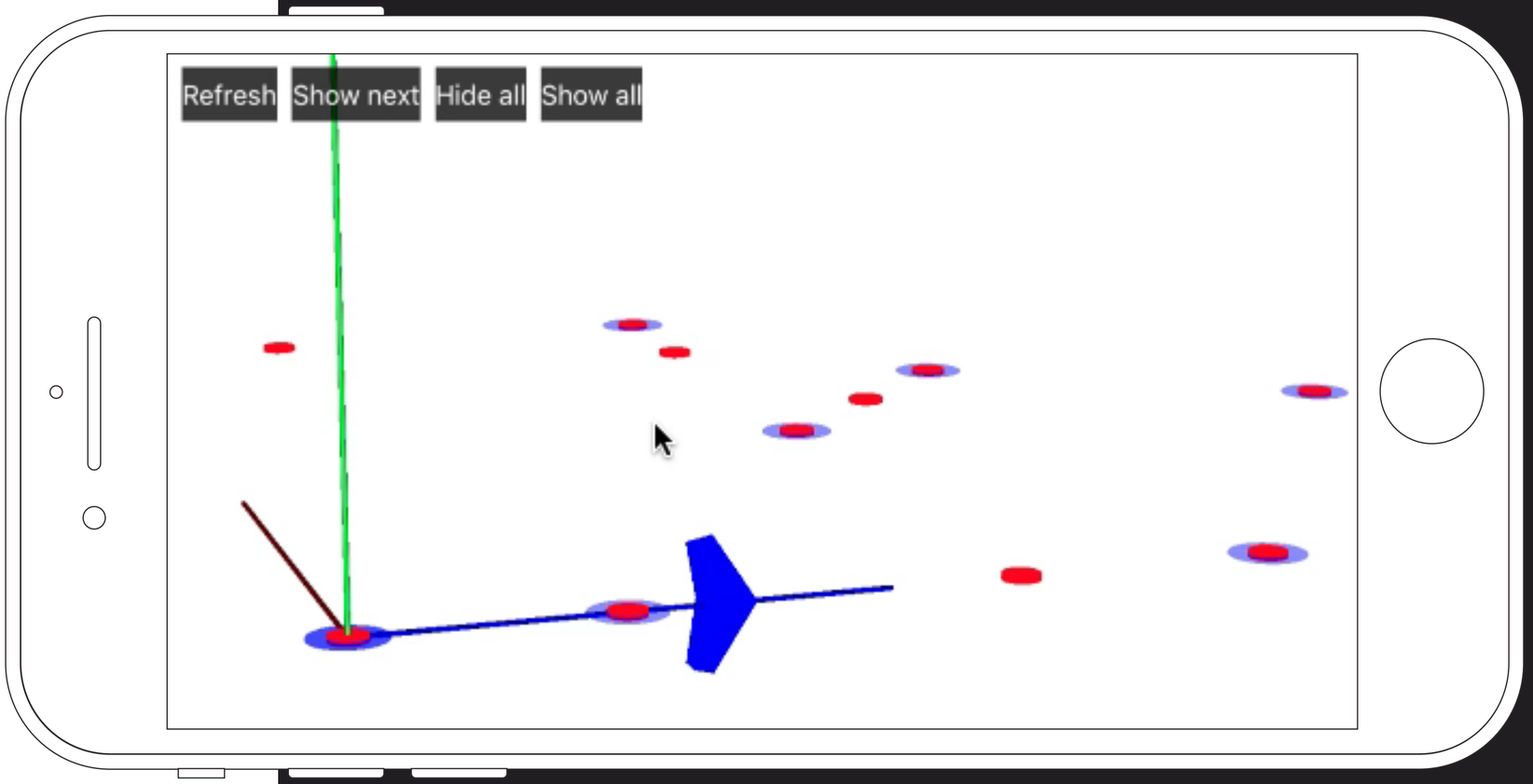
Итоговая анимация

```
let reset = createResetAction(firstStep: firstStep)
var stepActions: [SCNAction] = []
for stepIndex in 0..<animation.steps.count {
    let step = animation.steps[stepIndex]
    let stepAction = createAction(forStep: step, ...)
    stepActions.append(stepAction)
}
arrow.runAction(SCNAction.repeatForever(
    SCNAction.sequence([reset] + stepActions))
)
```

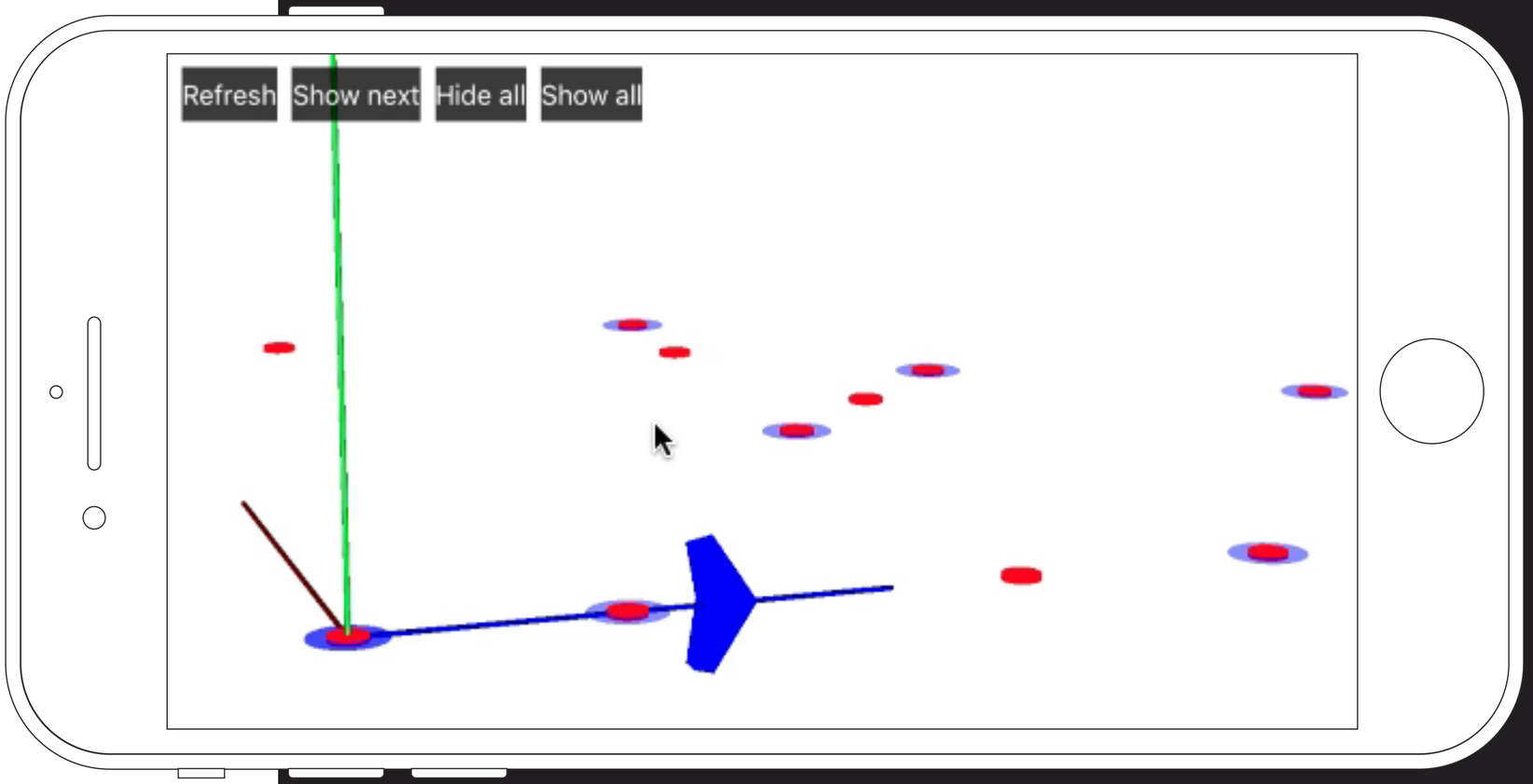
Итоговая анимация

```
let reset = createResetAction(firstStep: firstStep)
var stepActions: [SCNAction] = []
for stepIndex in 0..<animation.steps.count {
    let step = animation.steps[stepIndex]
    let stepAction = createAction(forStep: step, ...)
    stepActions.append(stepAction)
}
arrow.runAction(SCNAction.repeatForever(
    SCNAction.sequence([reset] + stepActions))
)
```

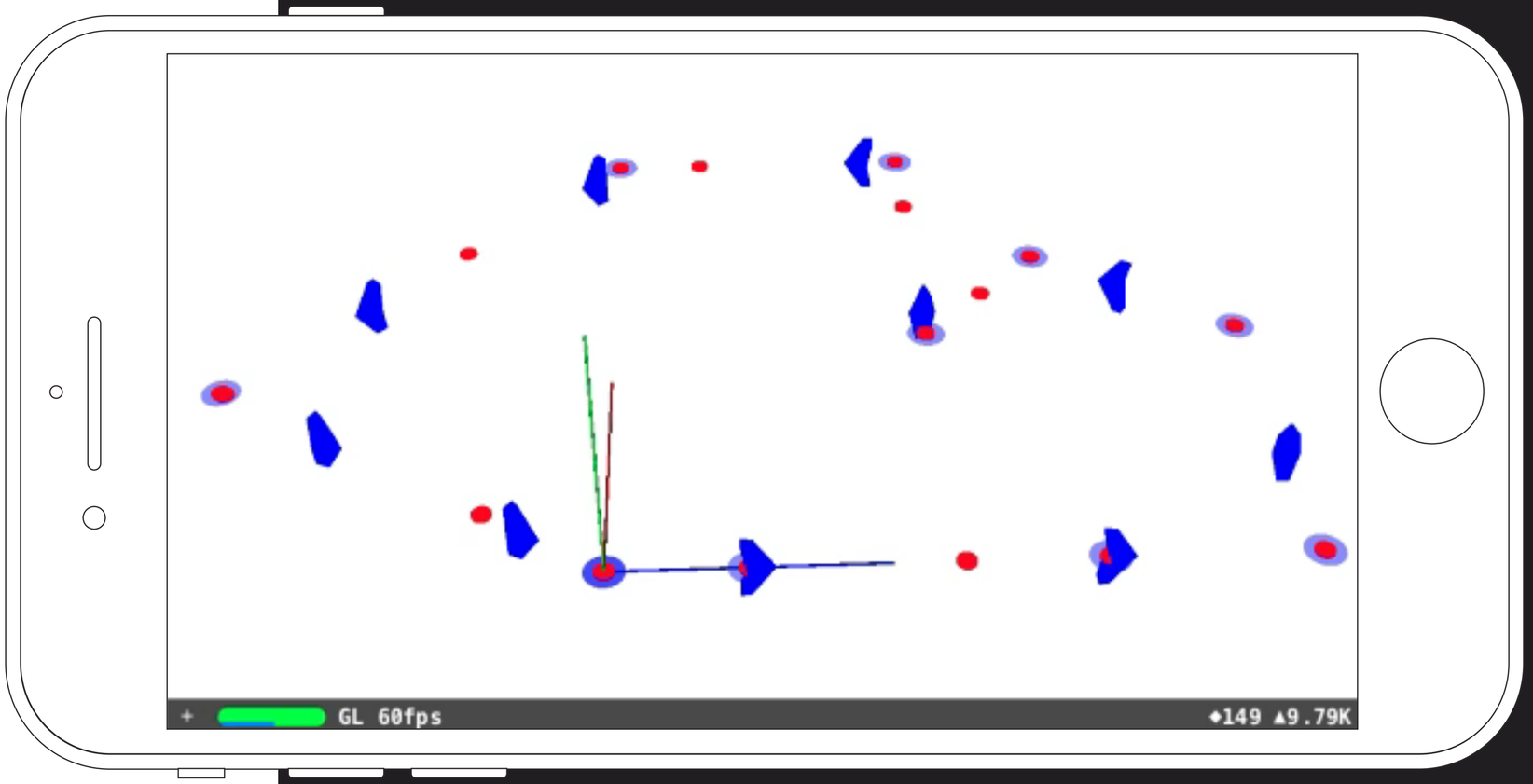
Анимация



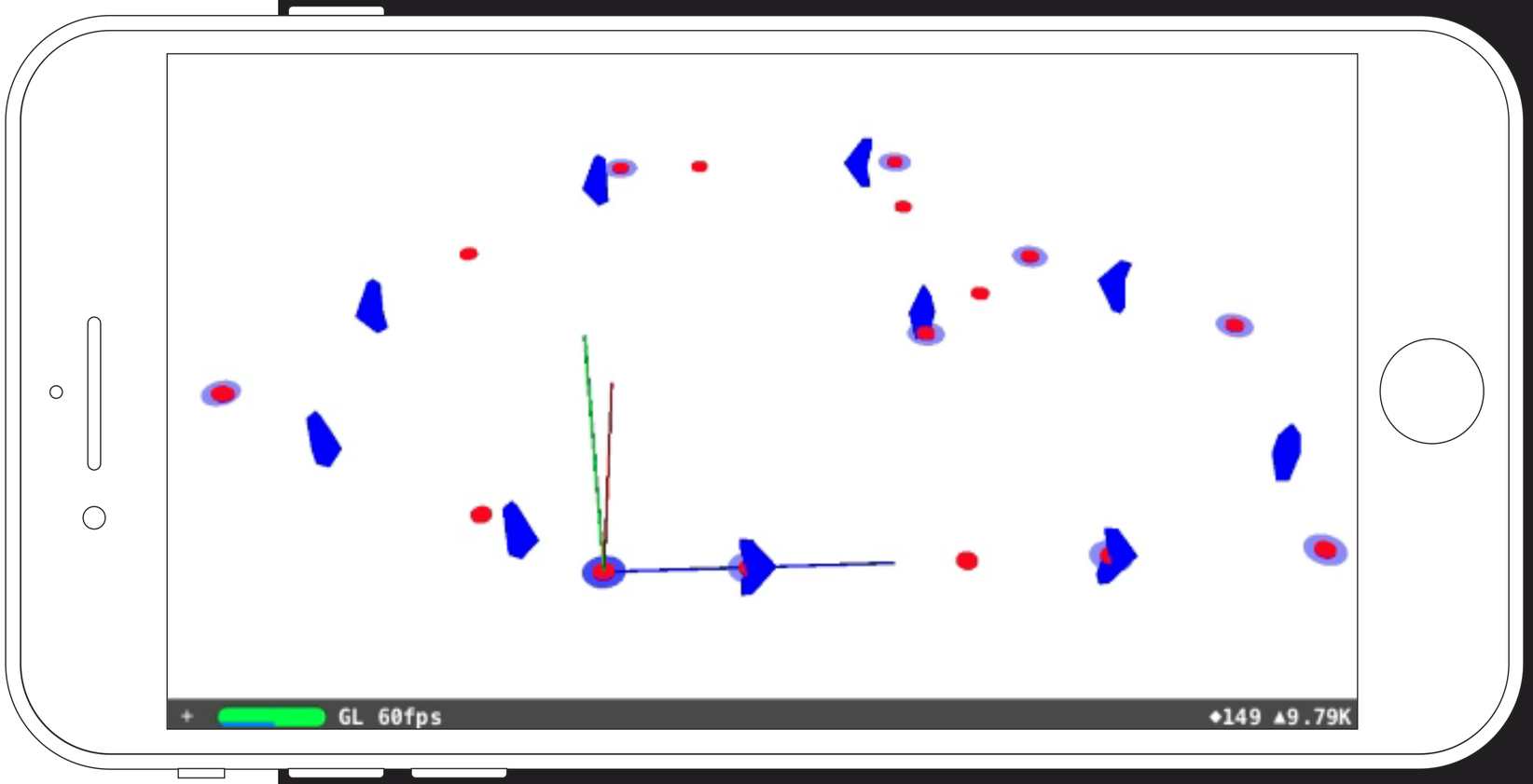
Анимация



Результат



Результат



ARKit + CoreLocation

ARKit создан для дополненной реальности в пределах твоей комнаты*

* из-за погрешностей, накапливаемых со временем

Системы координат

Геодезические - CoreLocation. **Декартовы** - ARKit (SceneKit)

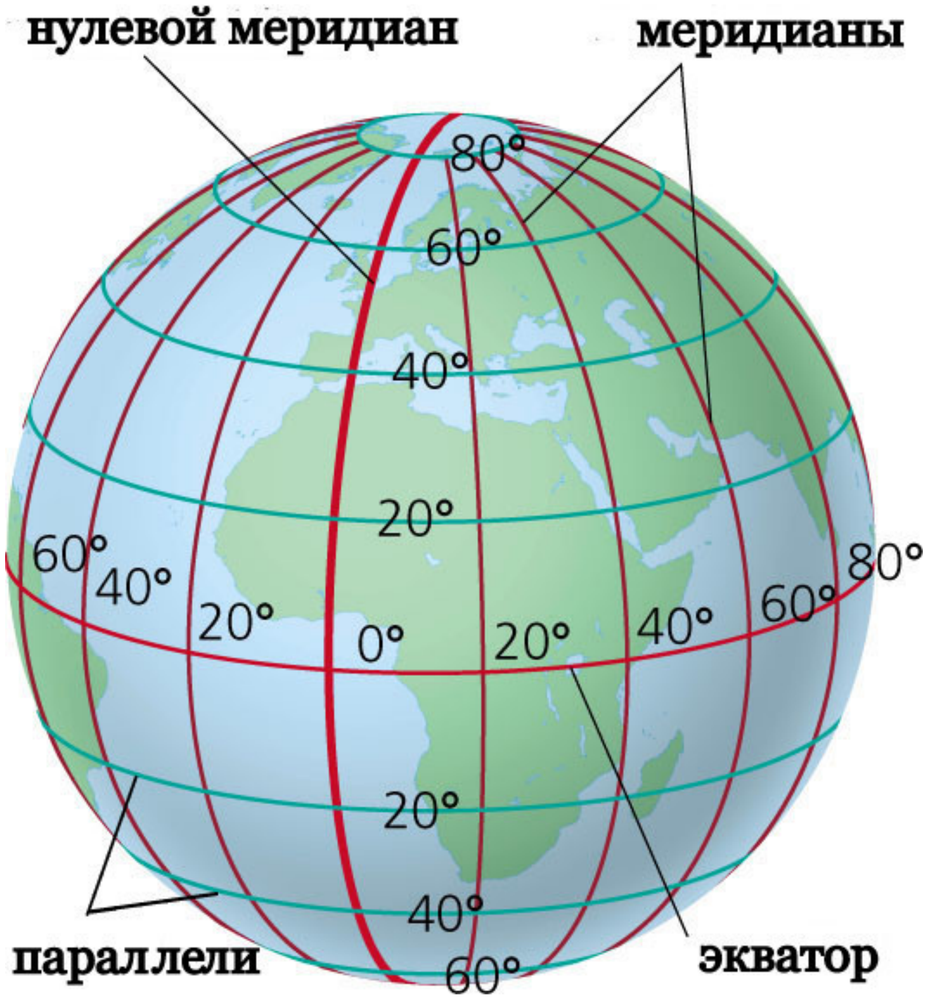
Не учитываем

- высоту над уровнем моря в данных CoreLocation
- компоненту Y в позиции на сцене

Возможна конвертация из декартовых в географические и обратно

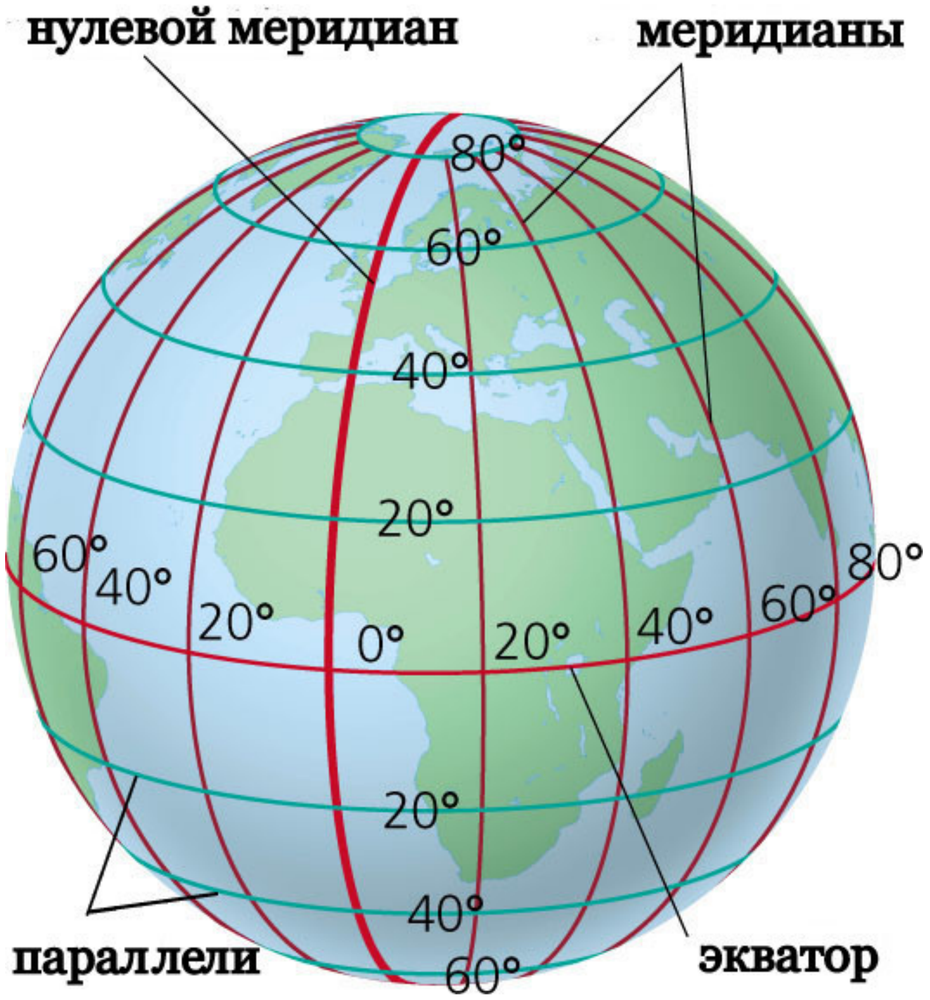
- в ARKit координаты измеряются в метрах
- смещение между двумя географическими координатами можно перевести в метры

КМБ по географии



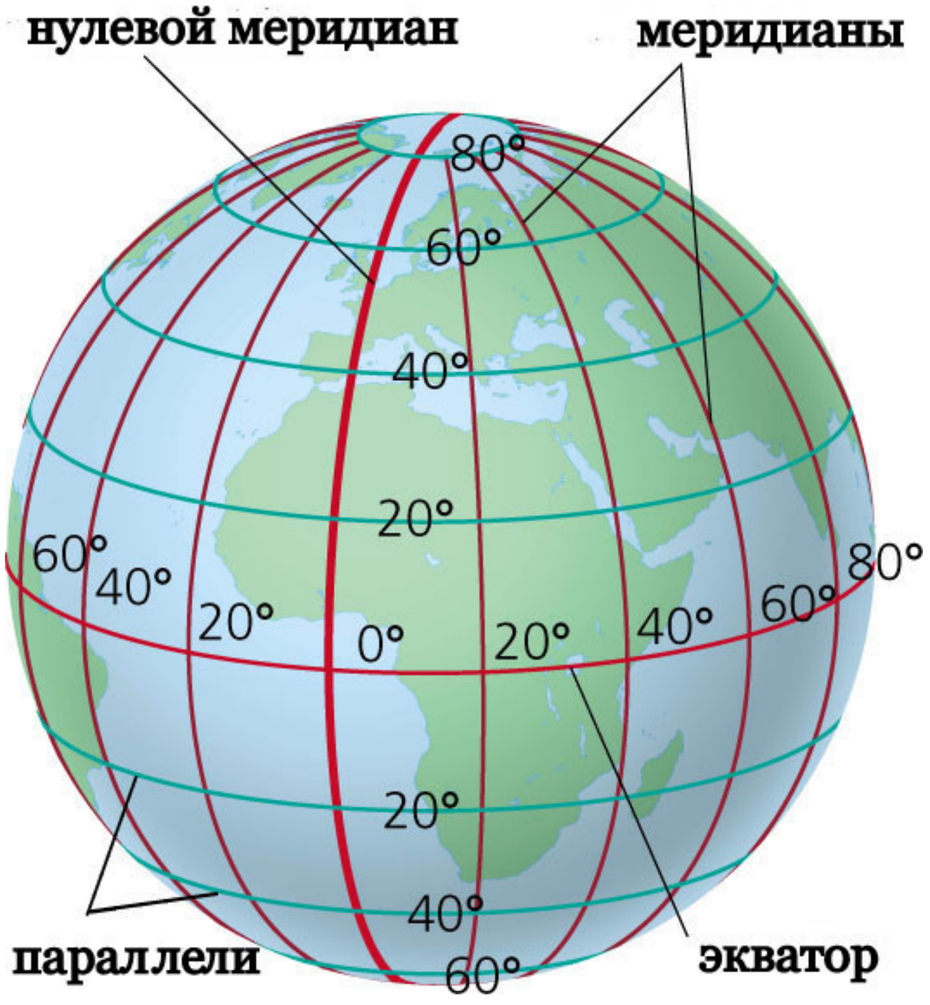
КМБ по географии

- Параллель – линия с градусным значением широты



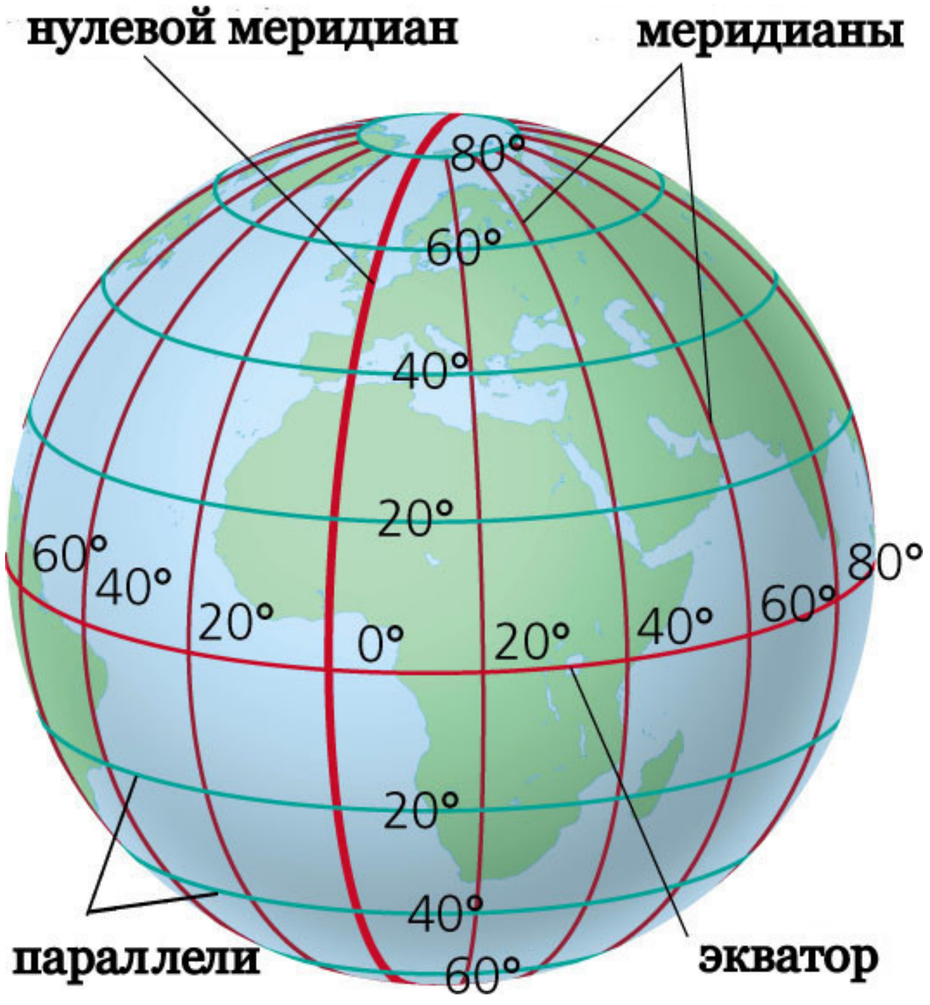
КМБ по географии

- Параллель – линия с градусным значением широты
  - длины различных параллелей различны



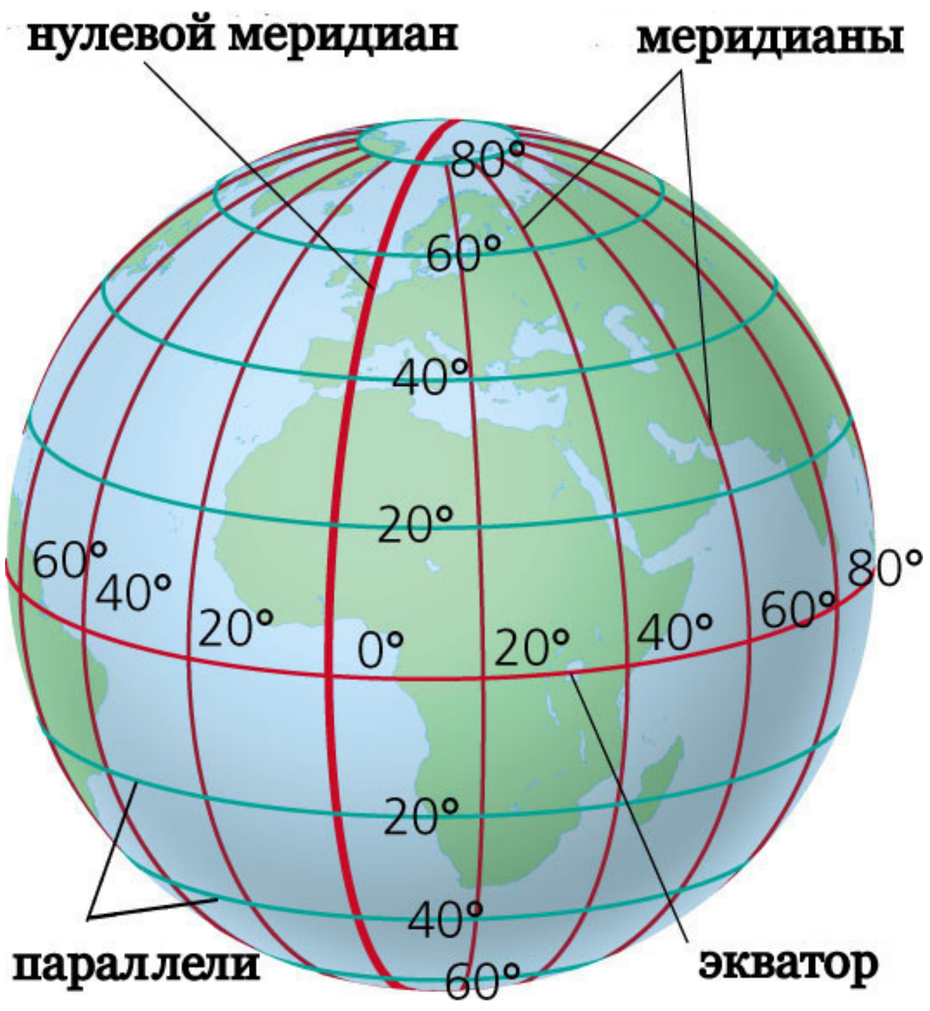
КМБ по географии

- Параллель – линия с градусным значением широты
  - длины различных параллелей различны
- Меридиан – линия с градусным значением долготы



КМБ по географии

- Параллель – линия с градусным значением широты
  - длины различных параллелей различны
- Меридиан – линия с градусным значением долготы
  - длины всех меридианов одинаковы



Переход между системами к-т

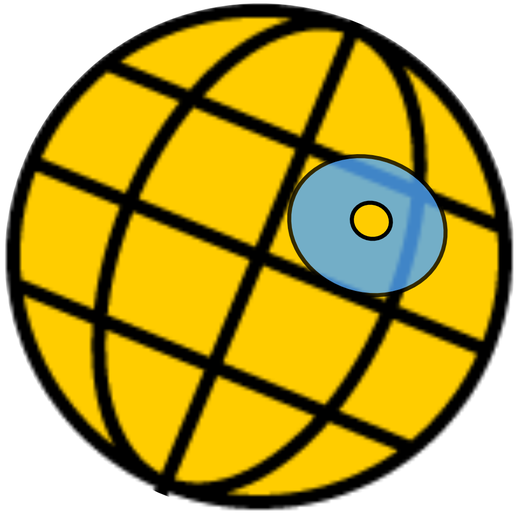
```
Δx = Δlon * metersInOneLonDegree(lat₀)
Δy = Δlat * metersInOneLatDegree
```

$$\text{кол-во метров в } 1^0 \text{ долготы на широте } \alpha = \frac{2\pi R_{\text{земли}} \cos(\alpha)}{360^0}$$

$$\text{кол-во метров в } 1^0 \text{ широты} = \frac{2\pi R_{\text{земли}}}{360^0}$$

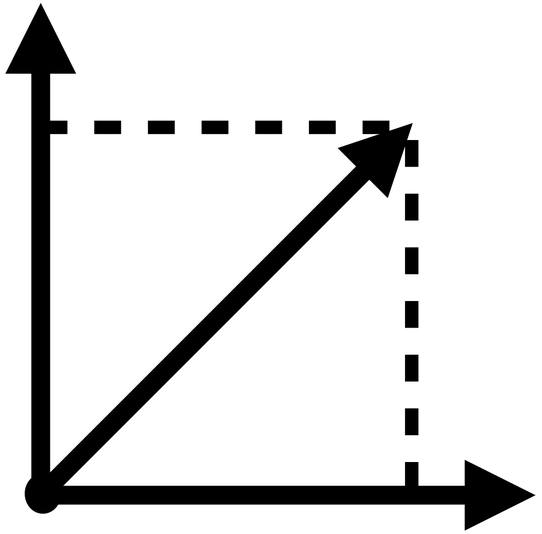
Эстимейт

(lat,lon,altitude)



+

(x,y,z)



Размещения объектов на сцене

Размещения объектов на сцене

Имея географическую координату объекта для определения его положения на сцене необходимо

Размещения объектов на сцене

Имея географическую координату объекта для определения его положения на сцене необходимо

- получить якорный эстимейт, относительно которого будут производиться дальнейшие расчеты

Размещения объектов на сцене

Имея географическую координату объекта для определения его положения на сцене необходимо

- получить якорный эстимейт, относительно которого будут производиться дальнейшие расчеты
- рассчитать смещение в метрах между географической к-той этого эстимейта и географической к-той объекта

Размещения объектов на сцене

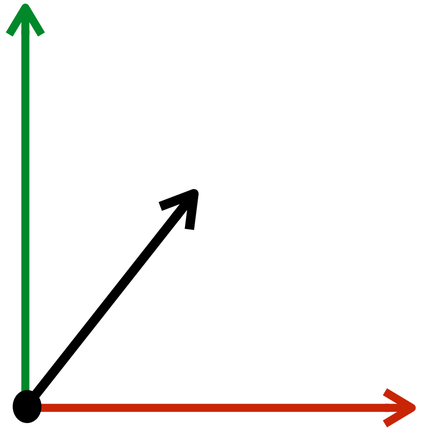
Имея географическую координату объекта для определения его положения на сцене необходимо

- получить якорный эстимейт, относительно которого будут производиться дальнейшие расчеты
- рассчитать смещение в метрах между географической к-той этого эстимейта и географической к-той объекта
- применить найденное смещение к позиции на сцене якорного эстимейта

Определение позиции объекта на сцене

Рассчитаем позицию объекта через смещение в географических к-тах

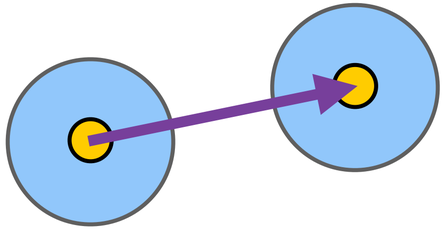
позиция якорного эстимейта



(x, z)

+

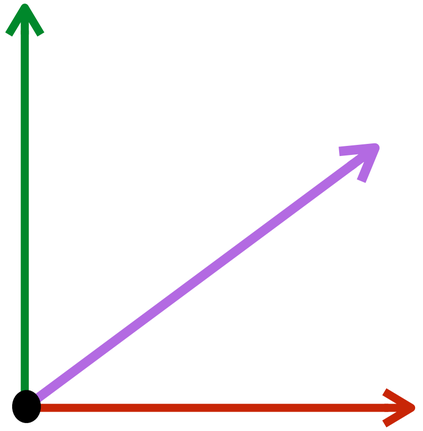
смещение в гео. координатах



Δ(lat, lon) -> Δ(x, z)

=

позиция объекта



(x+Δx, z+Δz)

Выбор якорного эстимейта

Наивный способ

- якорный эстимейт - например, полученный из последнего обновления координаты от CoreLocation

ARKit+CoreLocation

- якорный эстимейт - лучший эстимейт

Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов приводит к тормозам

Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов приводит к тормозам
- Погрешность данных CoreLocation

Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов приводит к тормозам
- Погрешность данных CoreLocation
- Точность снижается при удалении на расстояние > 100-150 м

ARKit+CoreLocation создан для уточнения текущего гео-местоположения за счет данных ARKit о смещении на сцене

https://github.com/ProjectDent/ARKit-CoreLocation

Концепция ARKit+CoreLocation

Концепция ARKit+CoreLocation

Объединение данных GRS (`CoreLocation`) с точным перемещением в 3D (`ARKit`) для уточнения местоположения

Концепция ARKit+CoreLocation

Объединение данных GRS (`CoreLocation`) с точным перемещением в 3D (`ARKit`) для уточнения местоположения

- в момент получения нового местоположения сохраним это значение и положение на 3D сцене как новый эстимейт

Концепция ARKit+CoreLocation

Объединение данных GRS (`CoreLocation`) с точным перемещением в 3D (`ARKit`) для уточнения местоположения

- в момент получения нового местоположения сохраним это значение и положение на 3D сцене как новый эстимейт
- сохраняем эстимейты в «массив»

Концепция ARKit+CoreLocation

Объединение данных GRS (`CoreLocation`) с точным перемещением в 3D (`ARKit`) для уточнения местоположения

- в момент получения нового местоположения сохраним это значение и положение на 3D сцене как новый эстимейт
- сохраняем эстимейты в «массив»
- для определения своего местоположения или позиции объекта на сцене, используем **лучший эстимейт**

Лучший эстимейт

Алгоритм

- лучший по точности*
- самый новый** среди эстимейтов с одинаковой точностью
- выбирается из эстимейтов, находящиеся в радиусе 100 м от текущего местоположения

Обновление

- при получении нового значения `CLLocation`
- по таймеру

* `horizontalAccuracy`, ** `timestamp`

Использование лучшего эстимейта уменьшает погрешность определения позиции объекта на сцене

Корректировка направления осей

Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов приводит к тормозам
- Погрешность данных CoreLocation
- Точность снижается при удалении на расстояние > 100-150 м

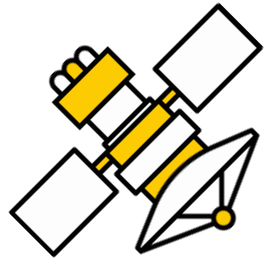
Ограничения

- Отрисовка десятков тысяч объектов для длинных маршрутов приводит к тормозам
- Погрешность данных CoreLocation
- Точность снижается при удалении на расстояние > 100-150 м
- Из-за погрешности компаса оси системы к-т сцены направлены не всегда так, как указано в документации*

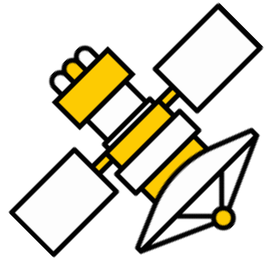
* Ось Z должны быть направлена на юг, но чаще всего это не так

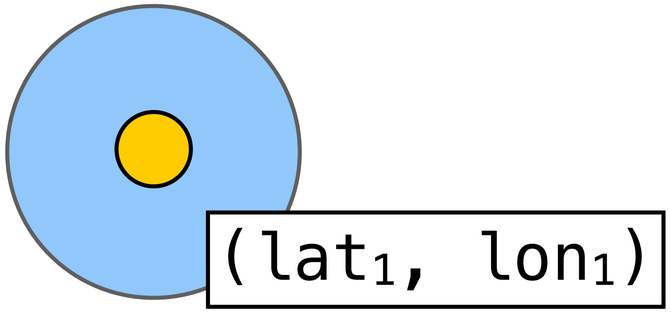
Корректировка осей. Проблема

Корректировка осей. Проблема



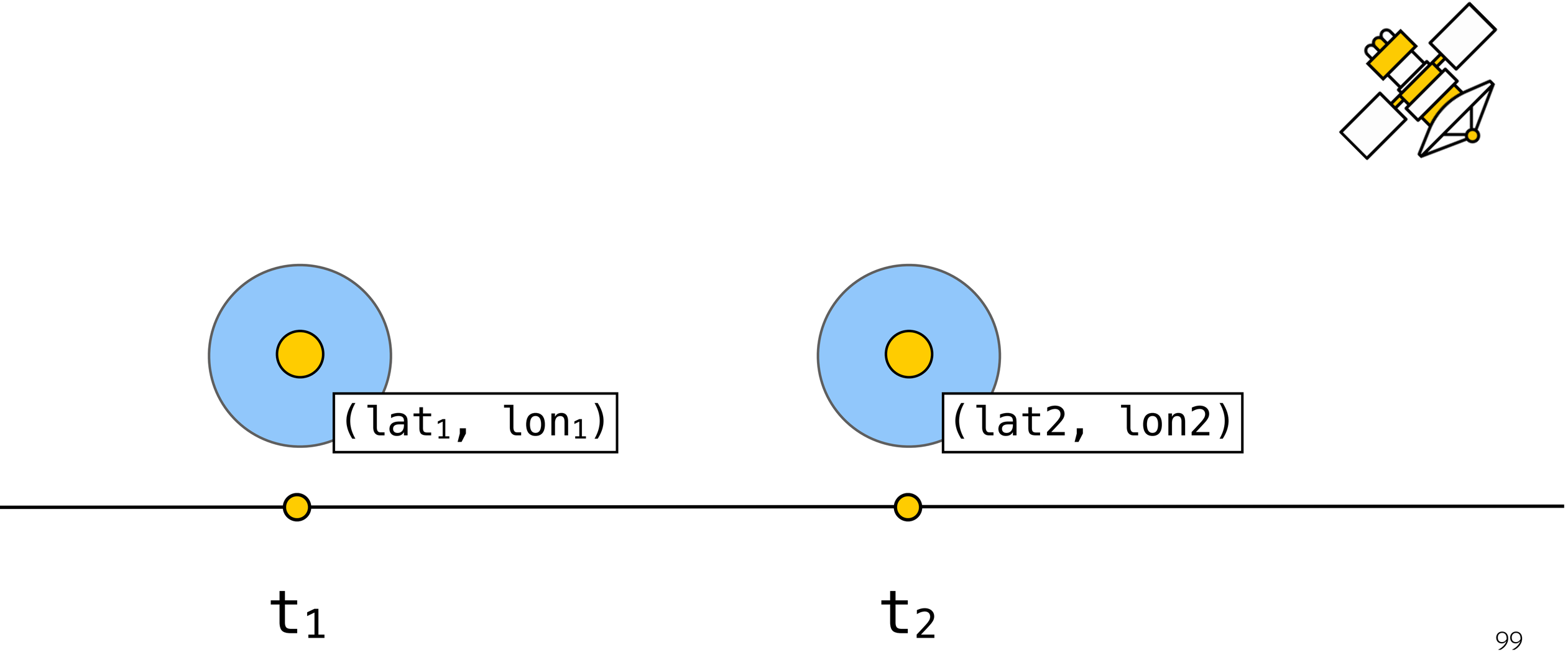
Корректировка осей. Проблема



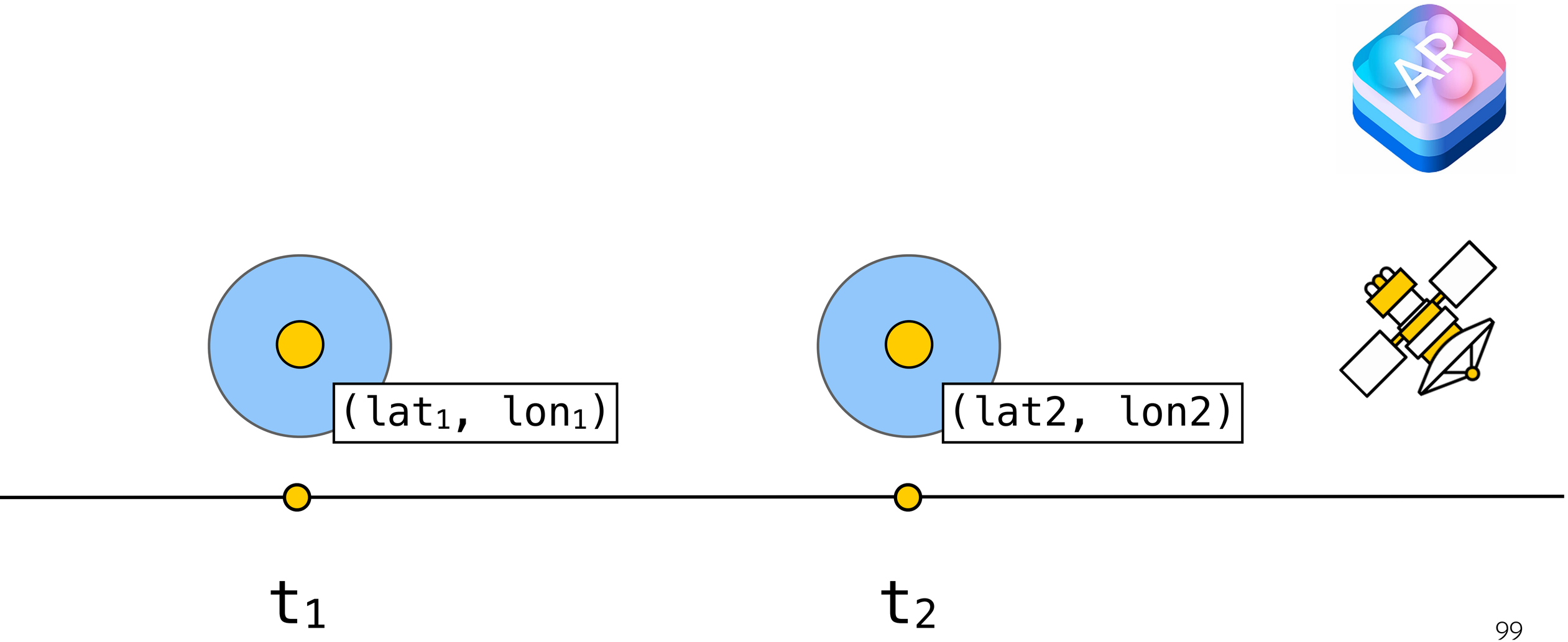


t_1

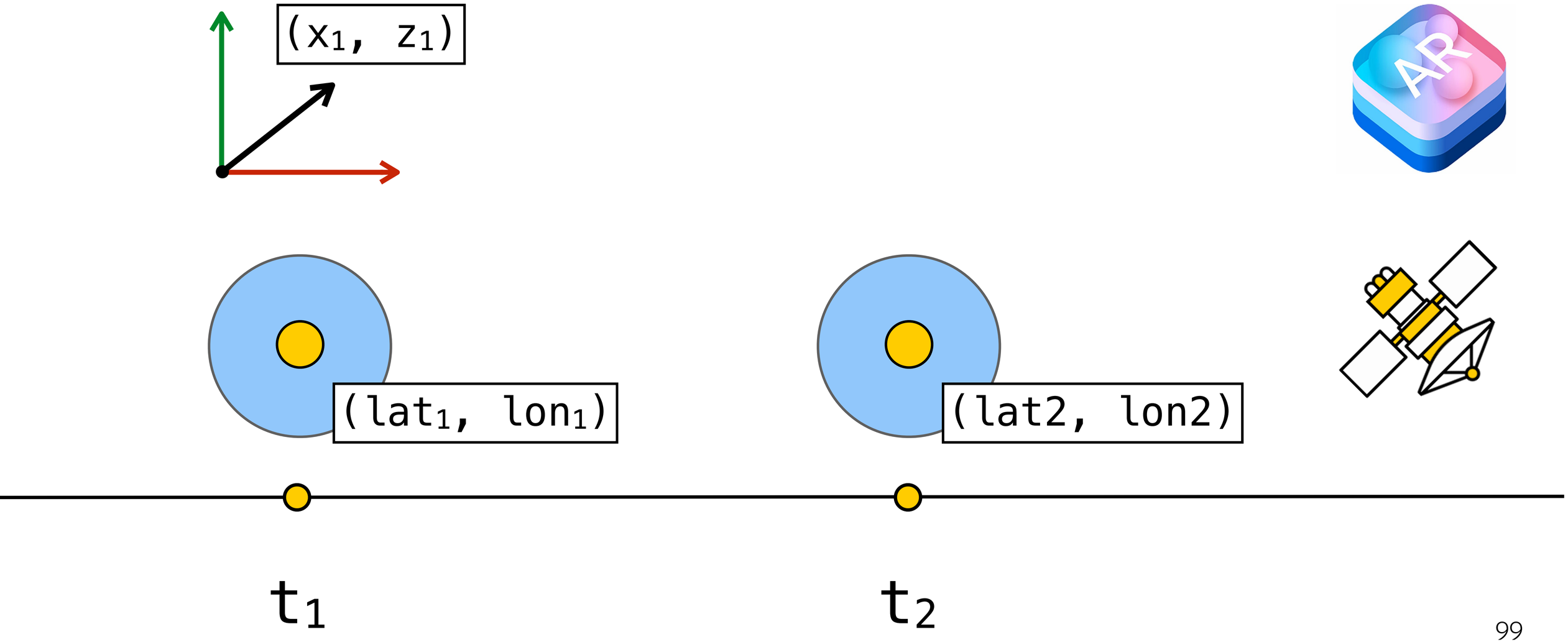
Корректировка осей. Проблема



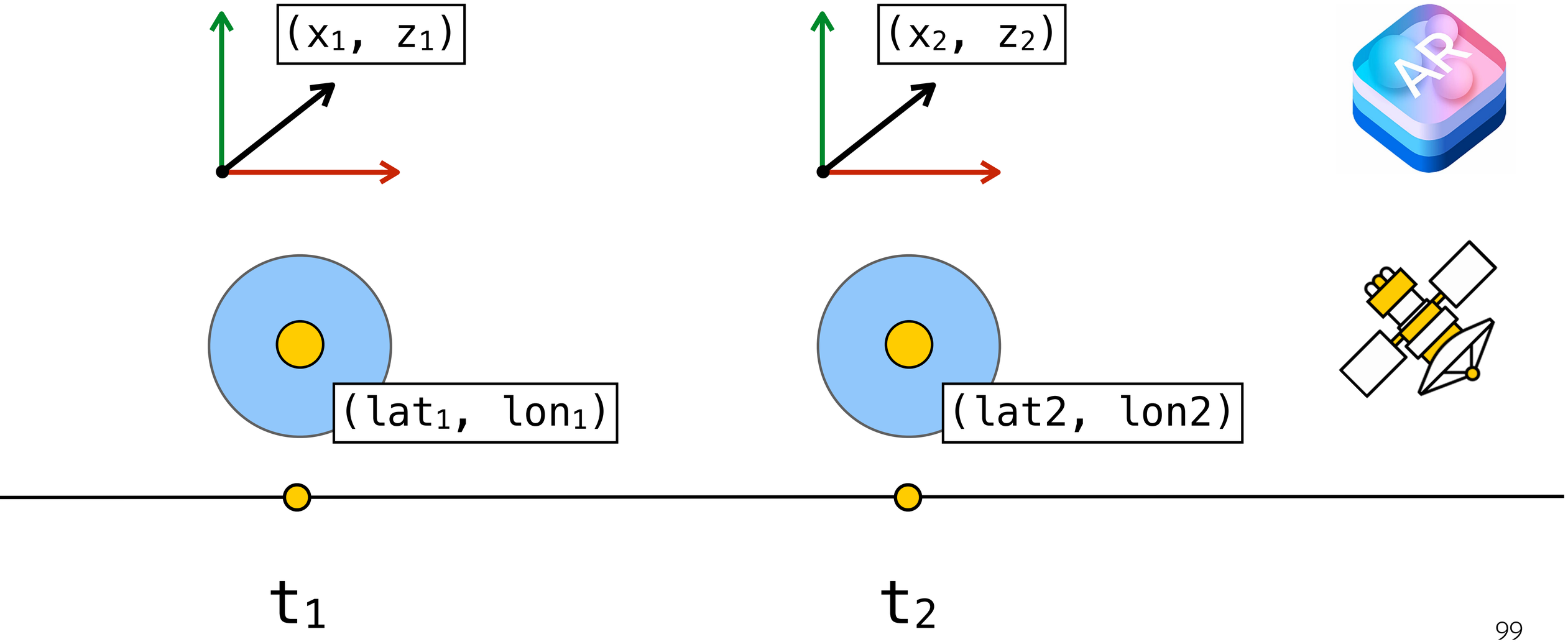
Корректировка осей. Проблема



Корректировка осей. Проблема

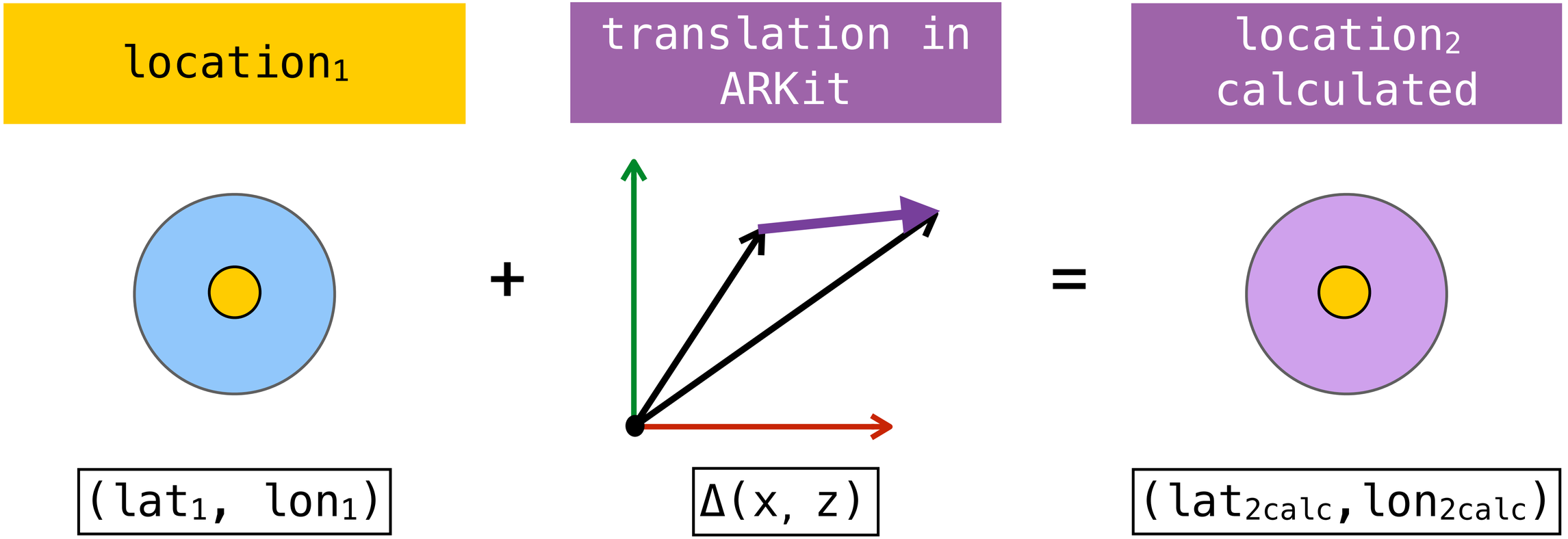


Корректировка осей. Проблема



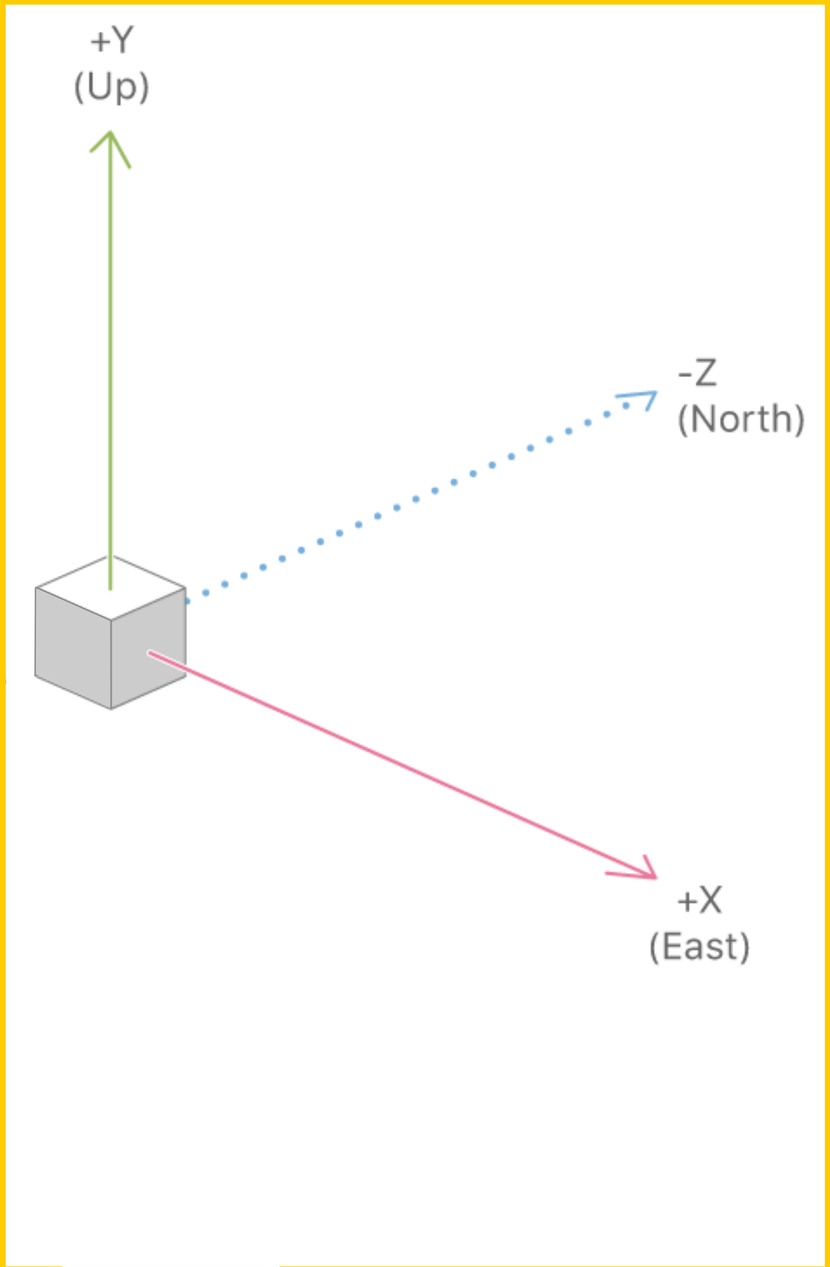
Корректировка осей. Проблема

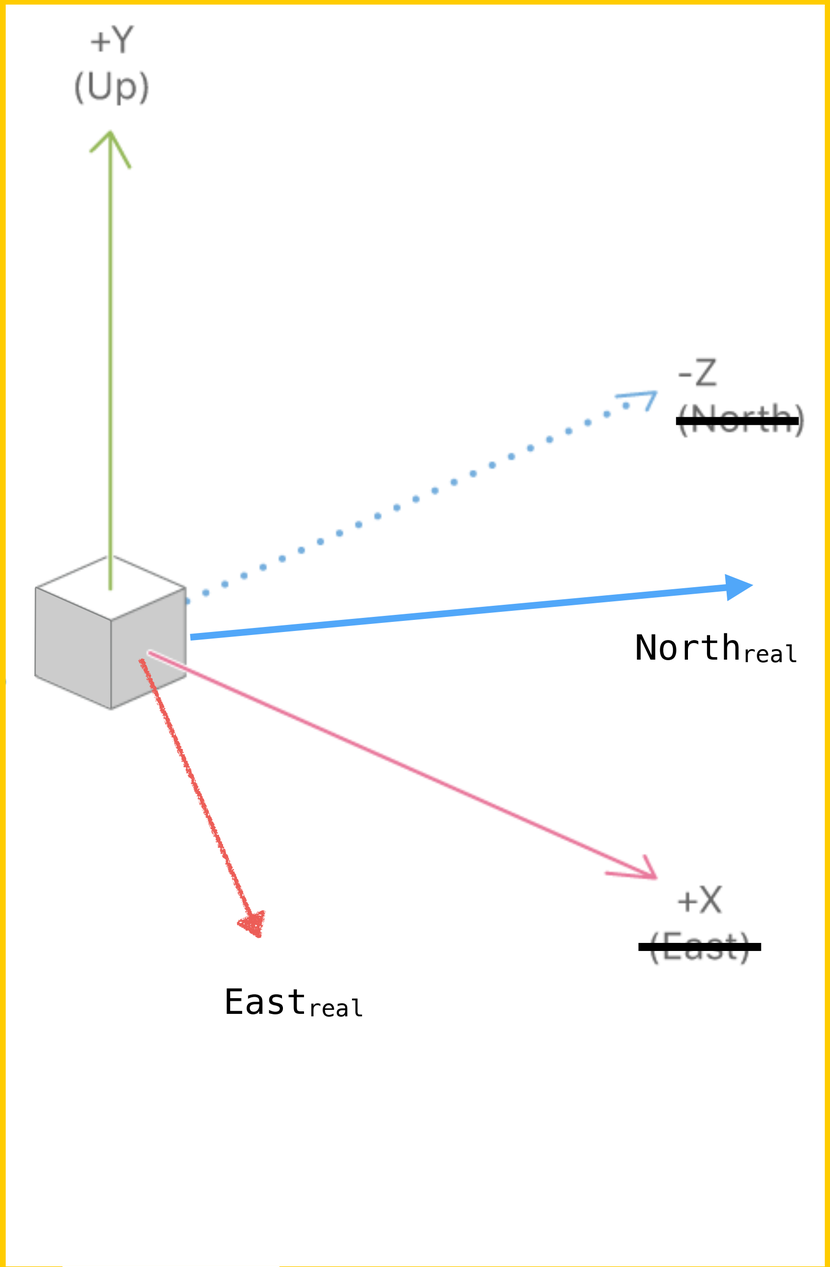
Рассчитаем положение точки 2 через смещение в ARKit

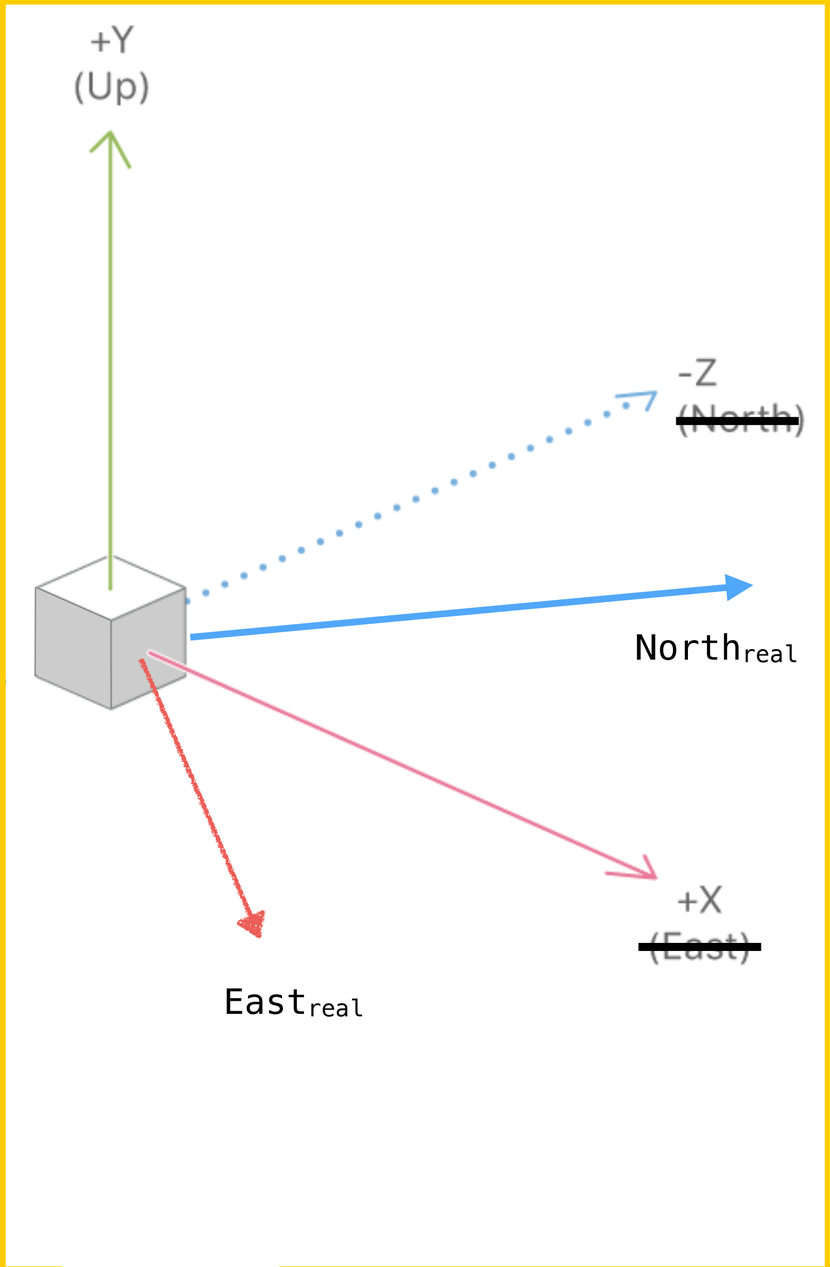


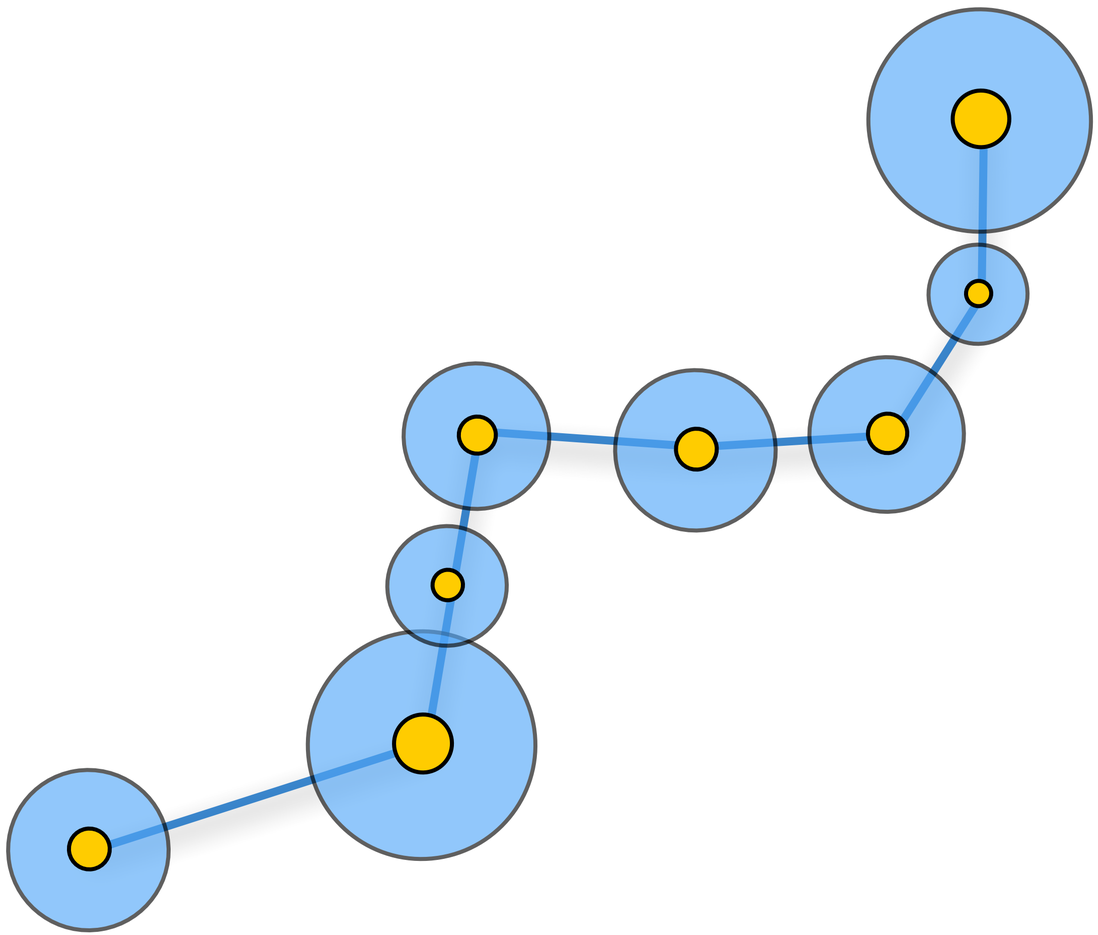
Рассчитанная
с использованием смещения
в ARKit географическая к-та
должна совпадать с к-той
полученной от CoreLocation

... это так, только если направления осей совпадают

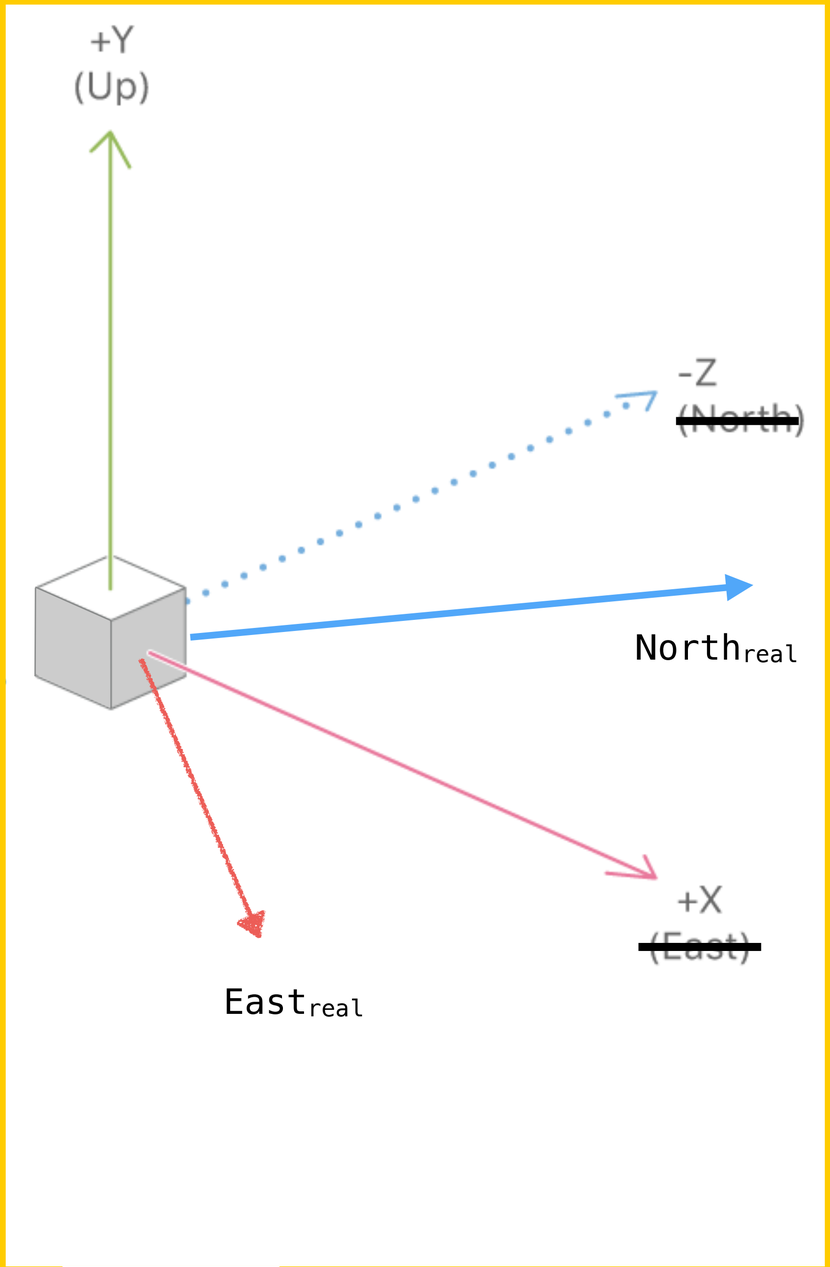

+Y
(Up)
-Z
(North)
+X
(East)

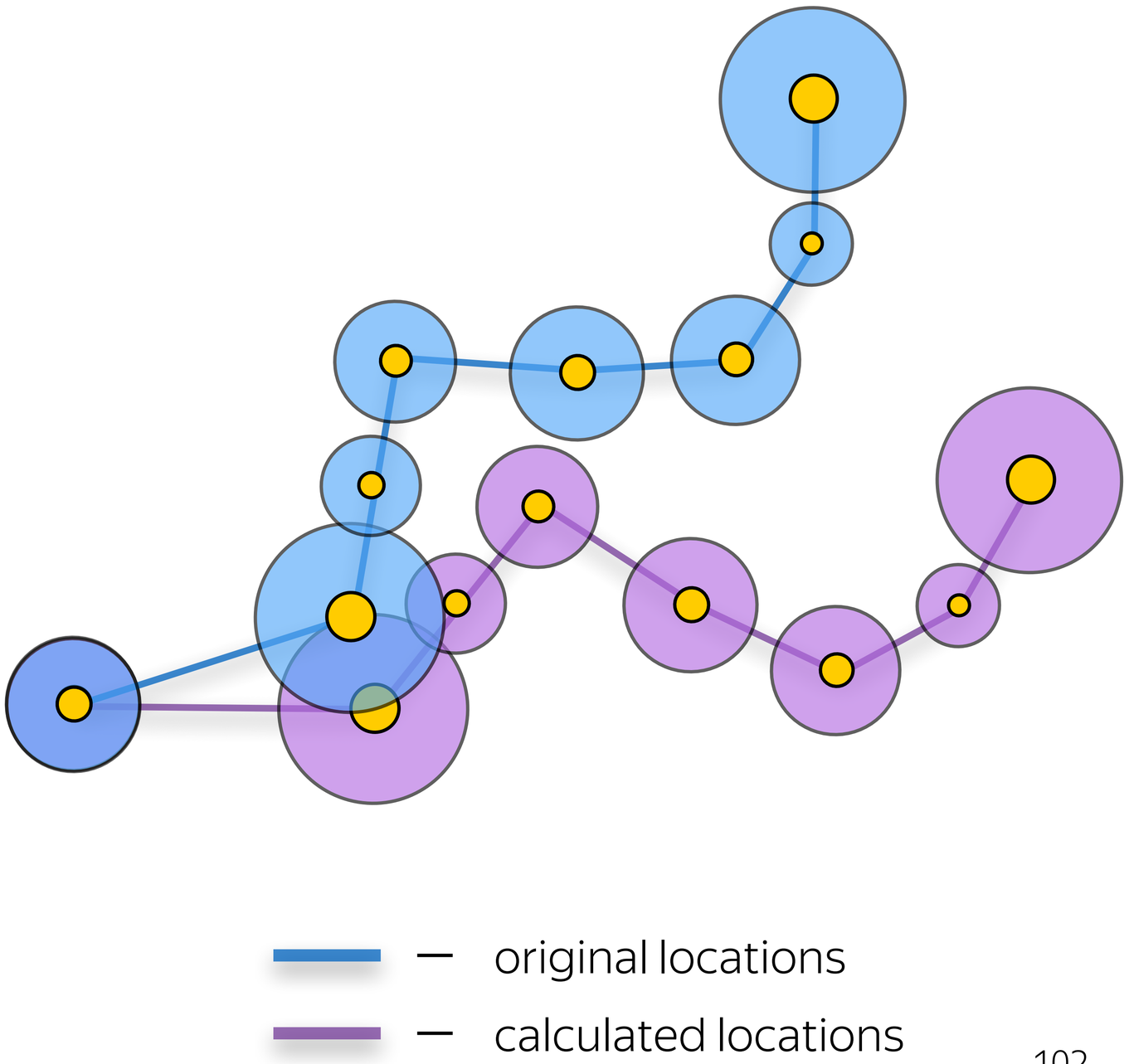

+Y
(Up)
-Z
~~(North)~~
North$_{real}$
+X
~~(East)~~
East$_{real}$

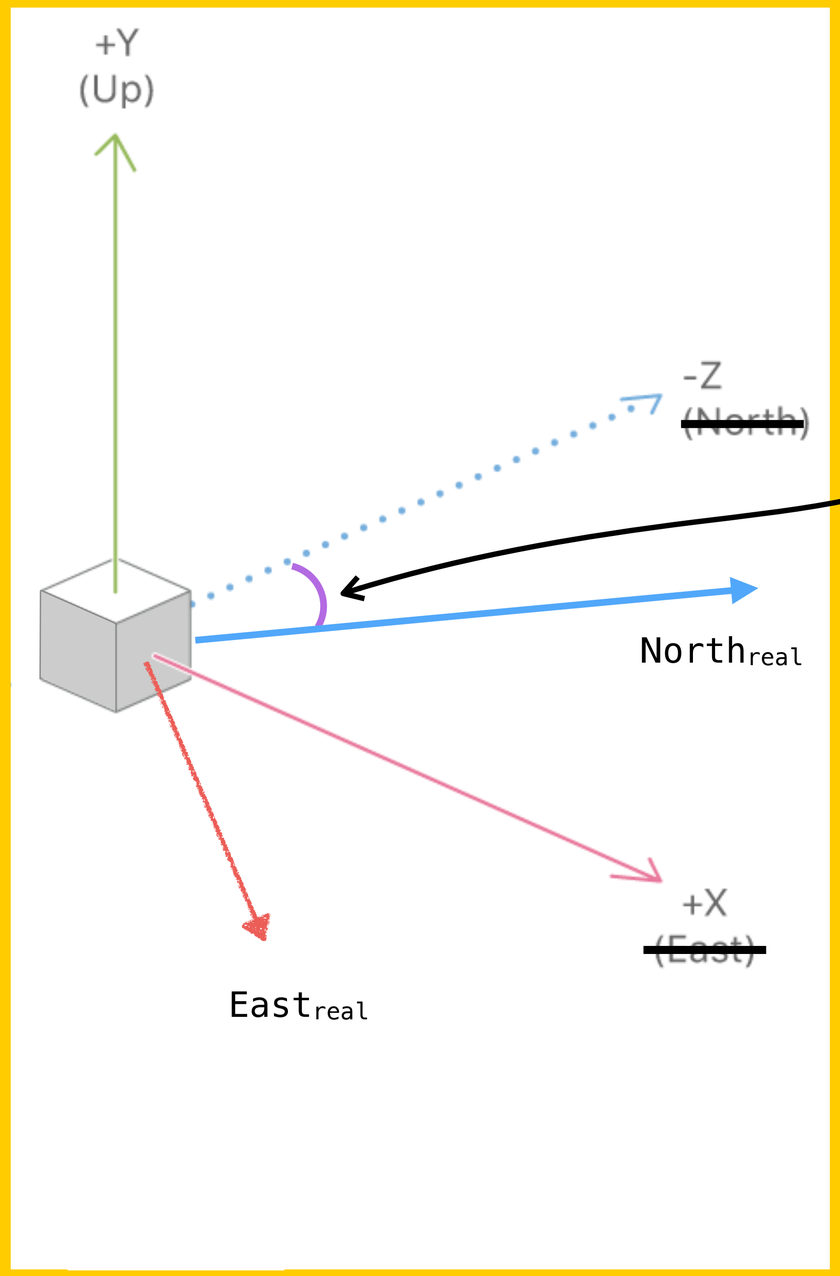




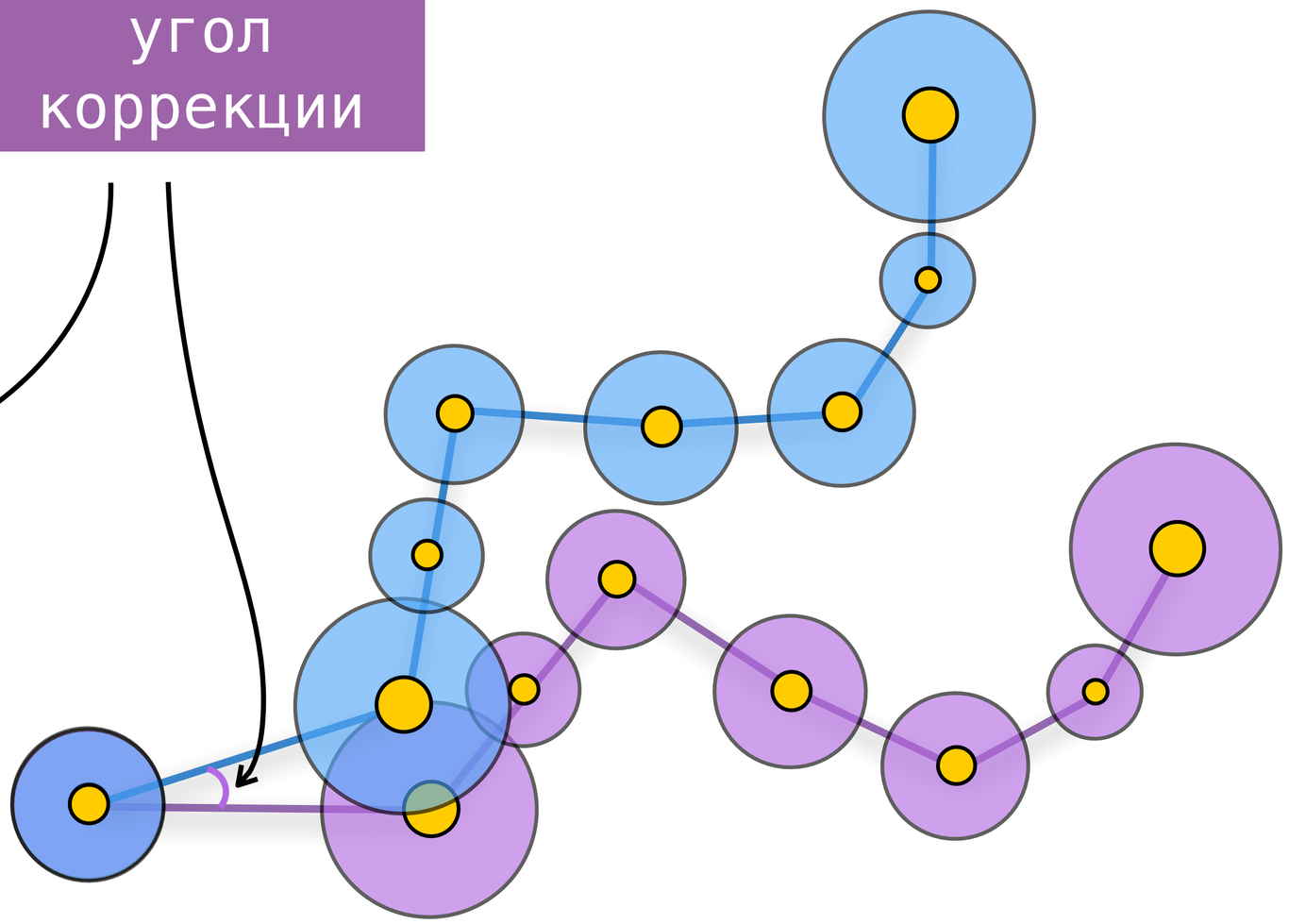
— original locations


+Y
(Up)
-Z
~~(North)~~
North$_{real}$
+X
~~(East)~~
East$_{real}$


— original locations
— calculated locations



угол коррекции



— original locations

— calculated locations

Алгоритм корректировки

Алгоритм корректировки

1. Фильтрация эстимейтов

Алгоритм корректировки

1. Фильтрация эстимейтов
2. Создаем буфер по времени или количеству

Алгоритм корректировки

1. Фильтрация эстимейтов
2. Создаем буфер по времени или количеству
3. Выбор пар эстимейтов

Алгоритм корректировки

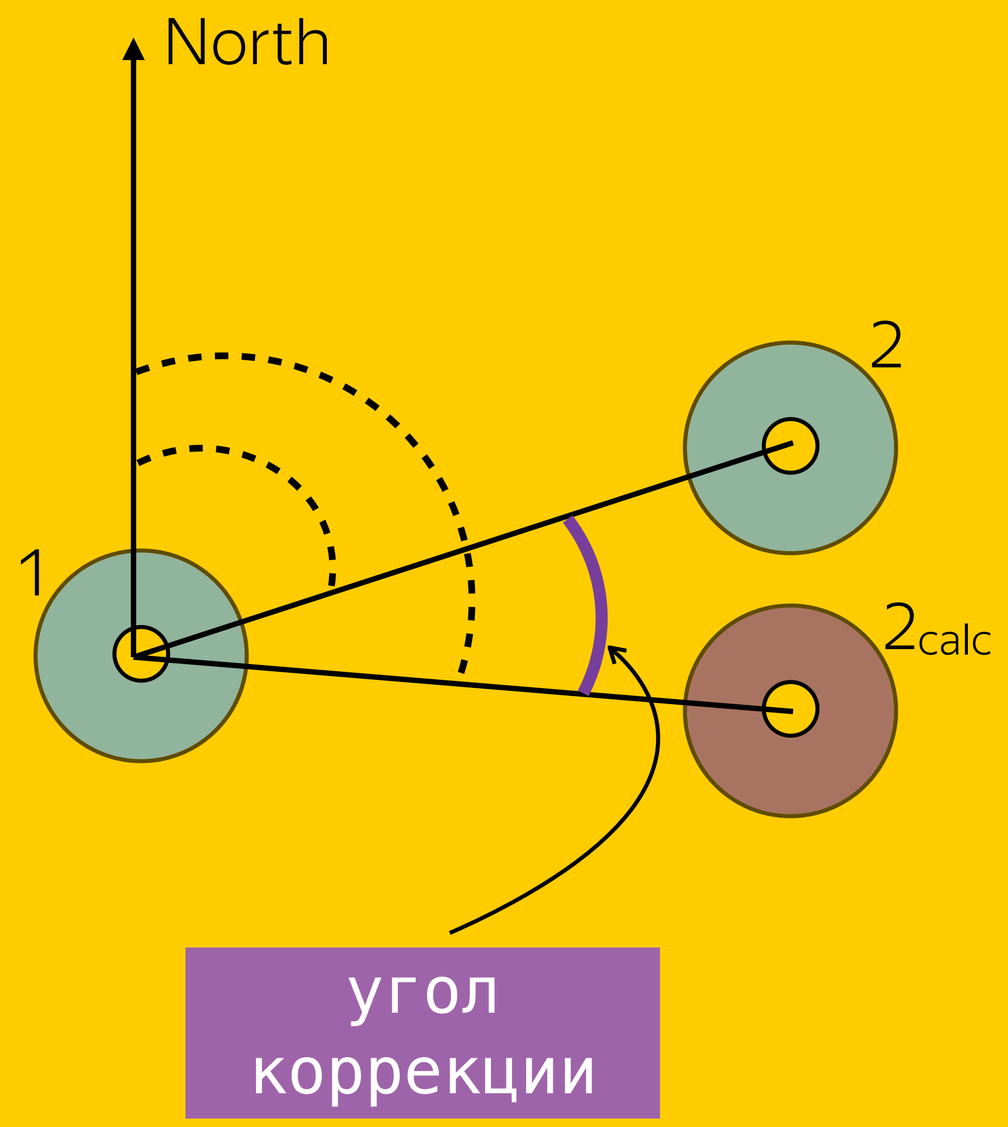
1. Фильтрация эстимейтов
2. Создаем буфер по времени или количеству
3. Выбор пар эстимейтов
4. Расчет угла коррекции для каждой пары

Алгоритм корректировки

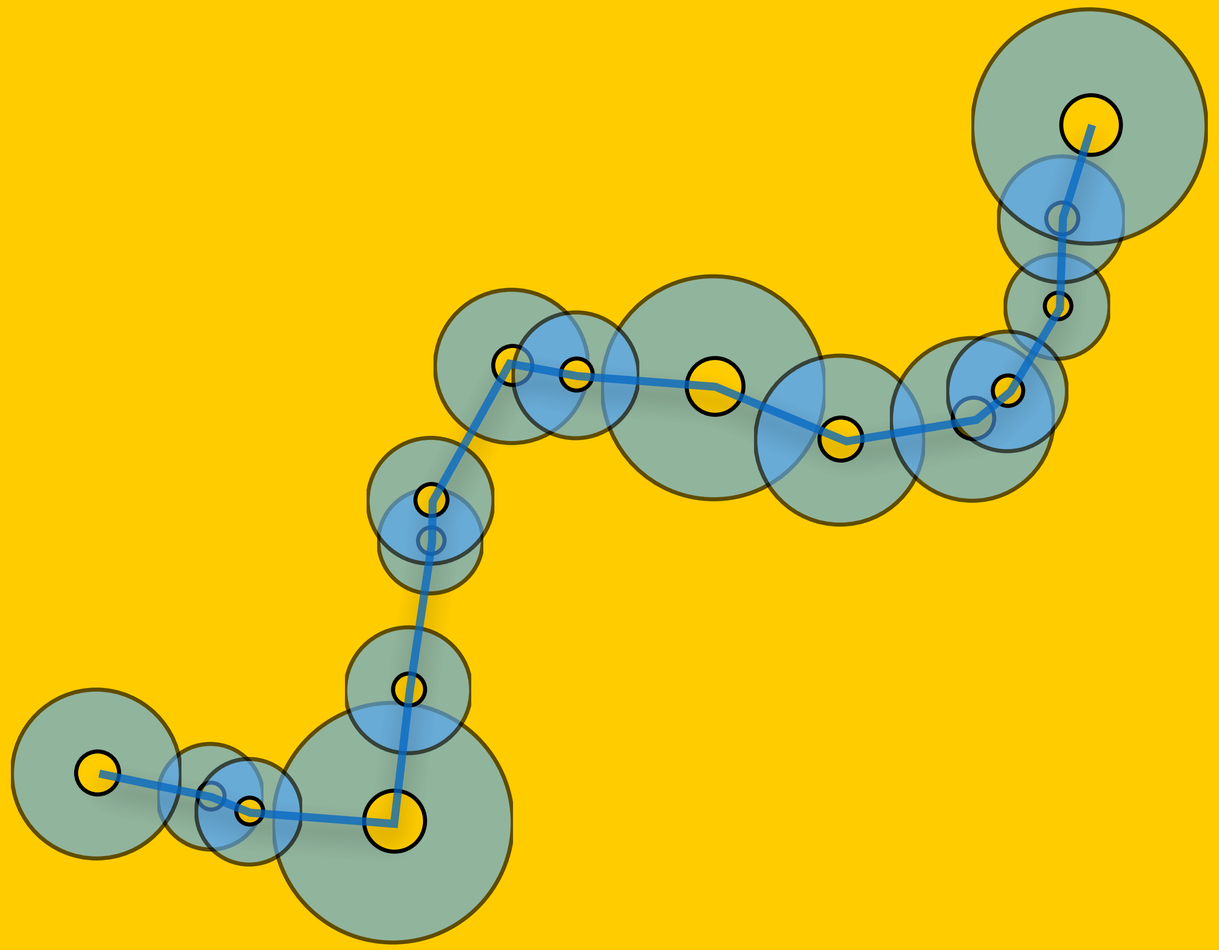
1. Фильтрация эстимейтов
2. Создаем буфер по времени или количеству
3. Выбор пар эстимейтов
4. Расчет угла коррекции для каждой пары
5. Усредняем угол по всем парам

Вычисление угла корректировки

- находим географическую к-ту точки 2 используя смещение в ARKit
- находим разность углов initialBearing(1, 2) и initialBearing(1, 2_{calc})

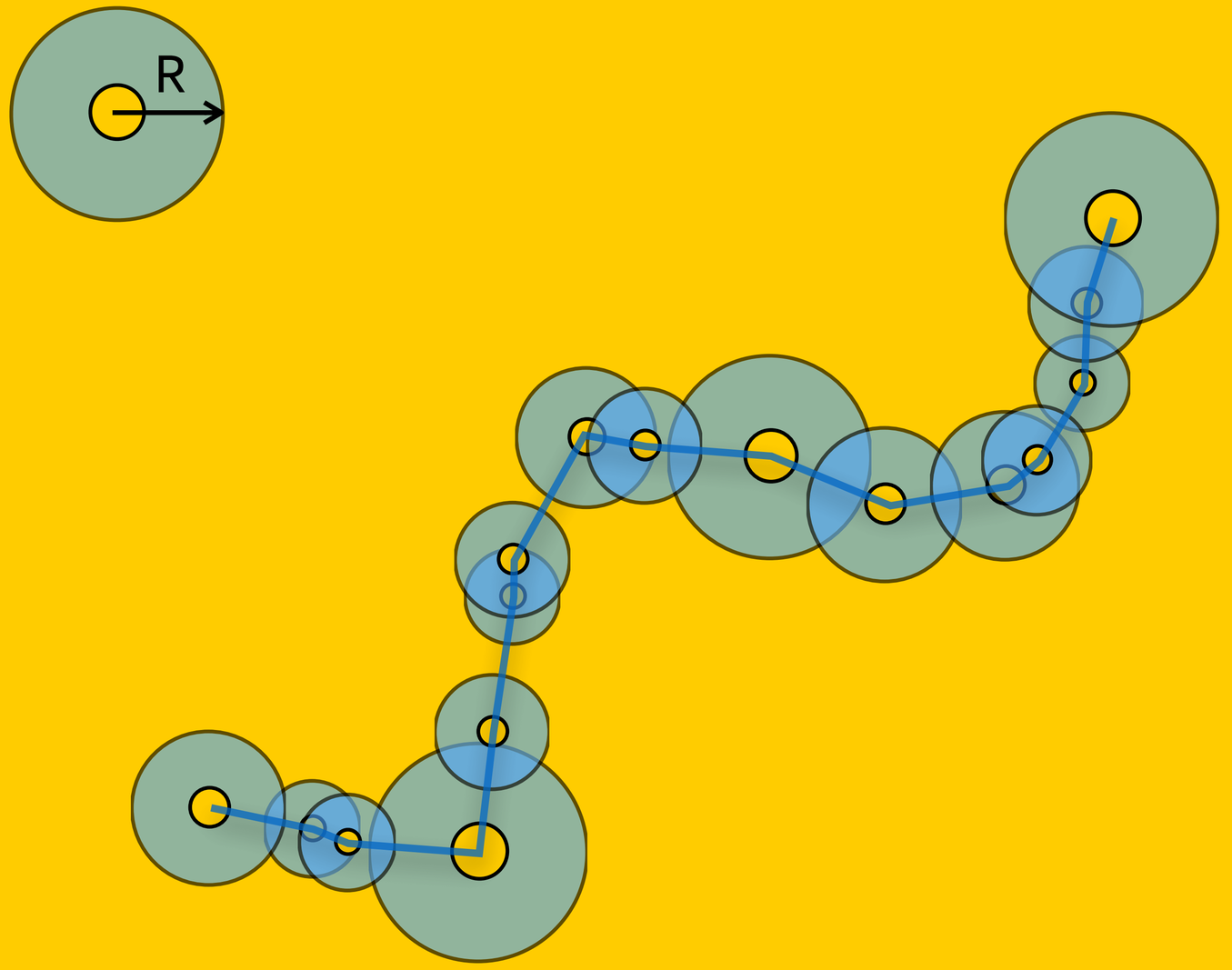


Фильтрация эстимейтов



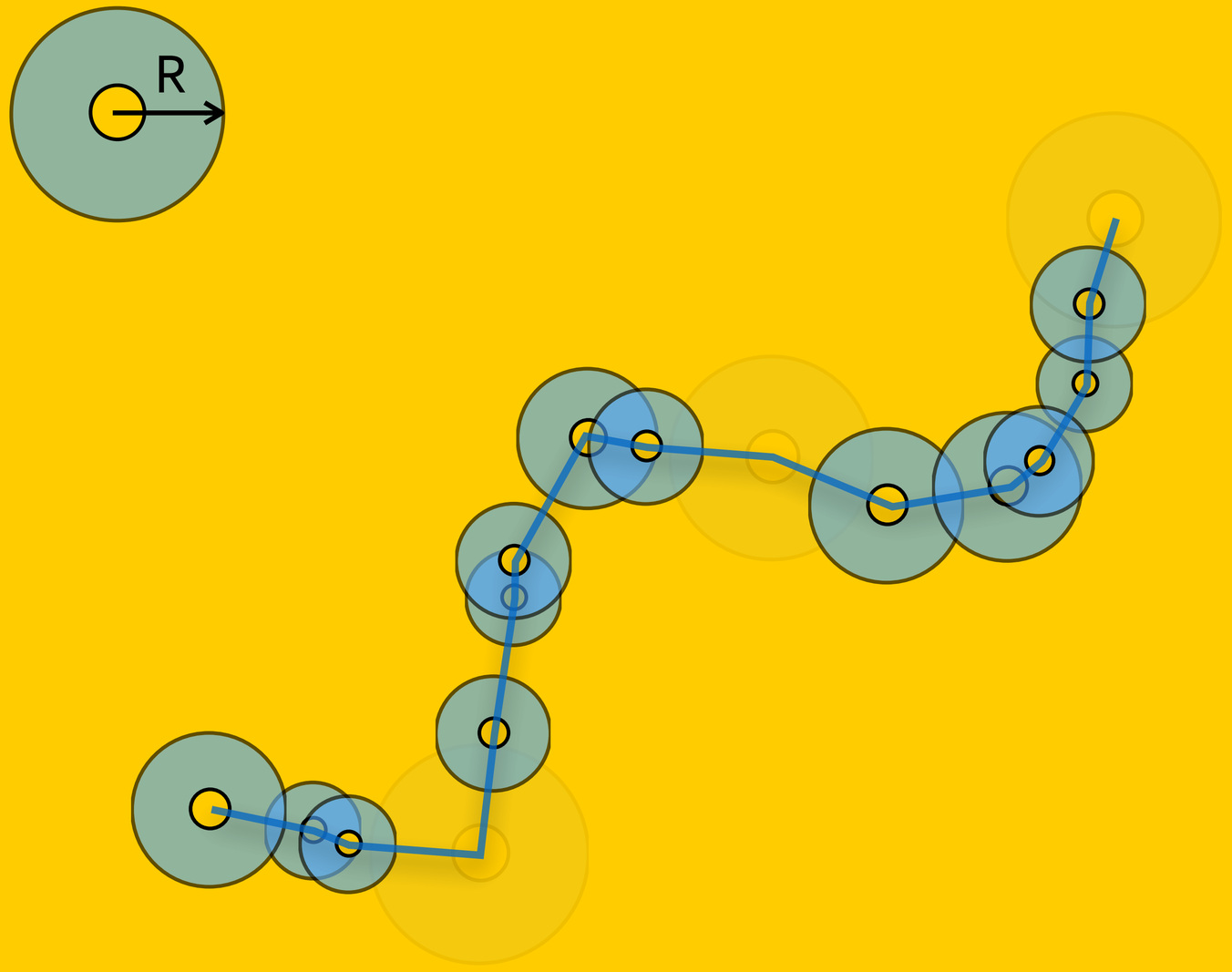
Фильтрация эстимейтов

- по точности географической координаты



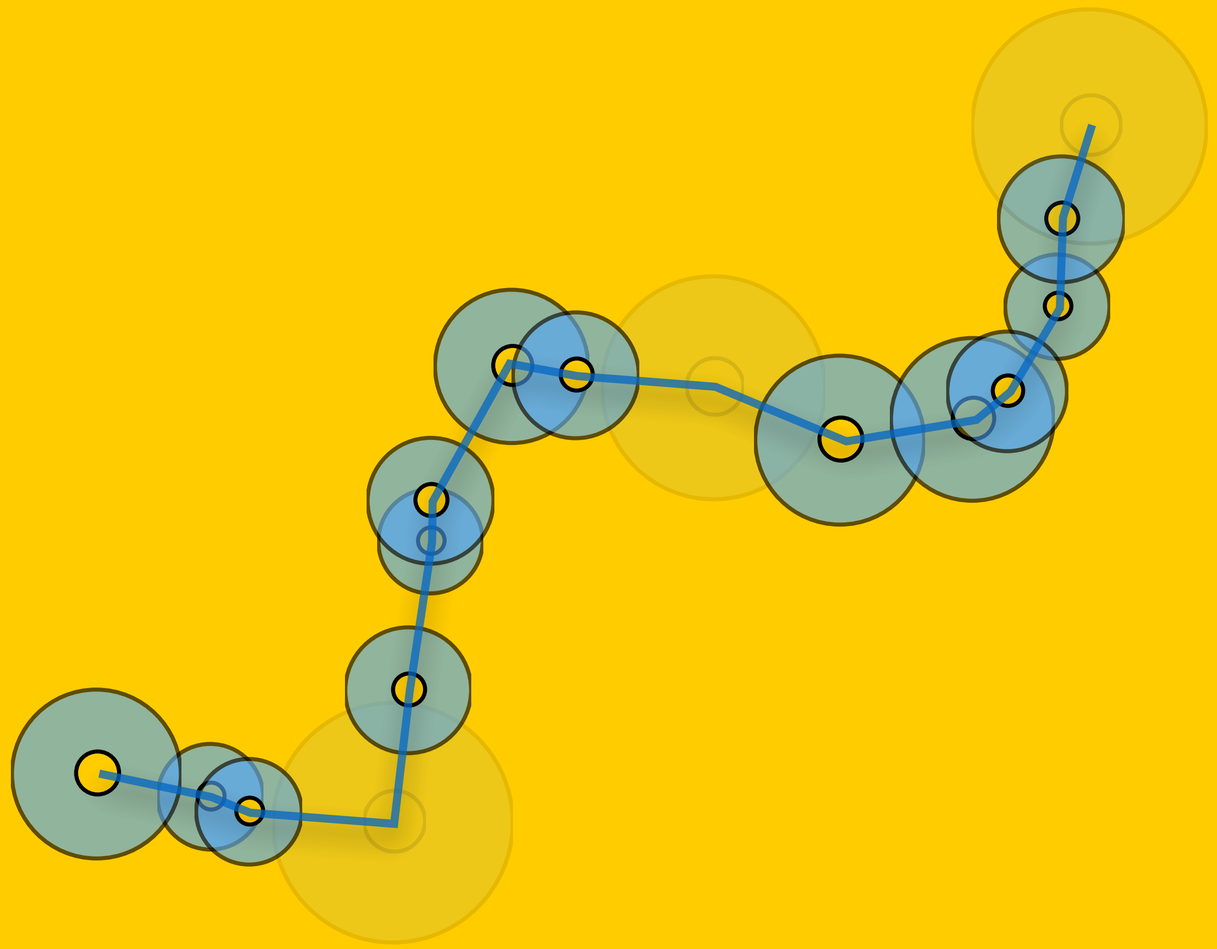
Фильтрация эстимейтов

- по точности географической координаты



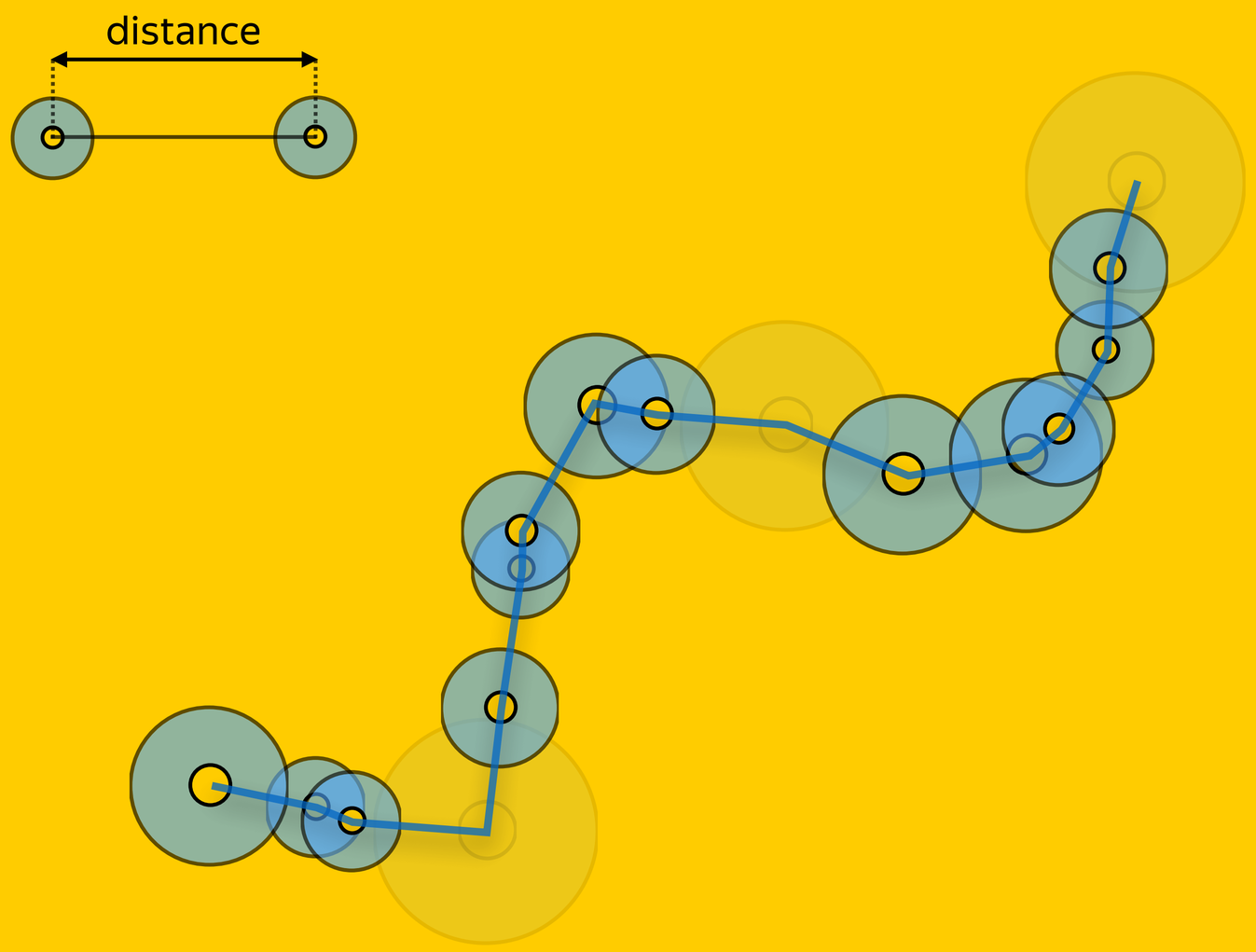
Фильтрация эстимейтов

- по точности географической координаты



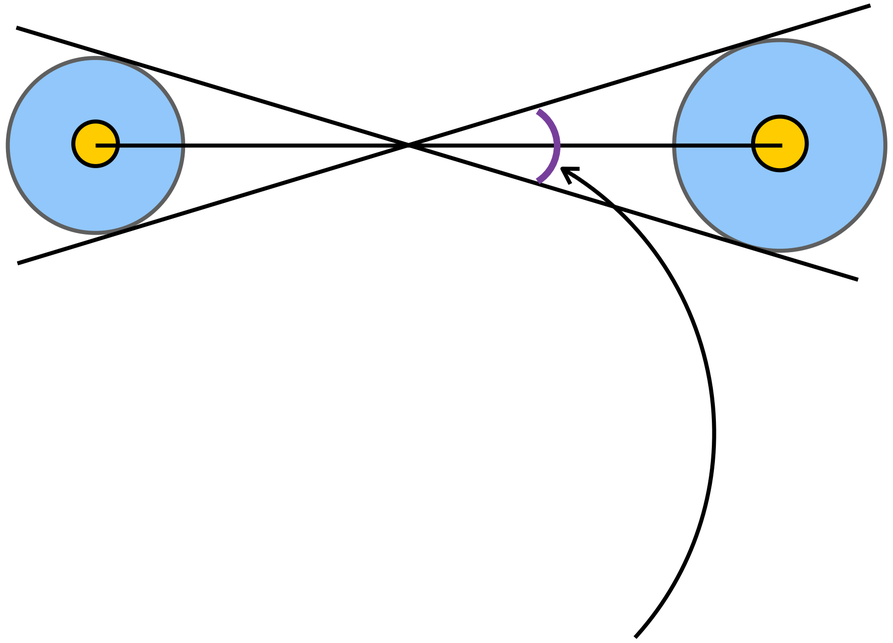
Фильтрация эстимейтов

- по точности географической координаты
- по расстоянию между эстимейтами

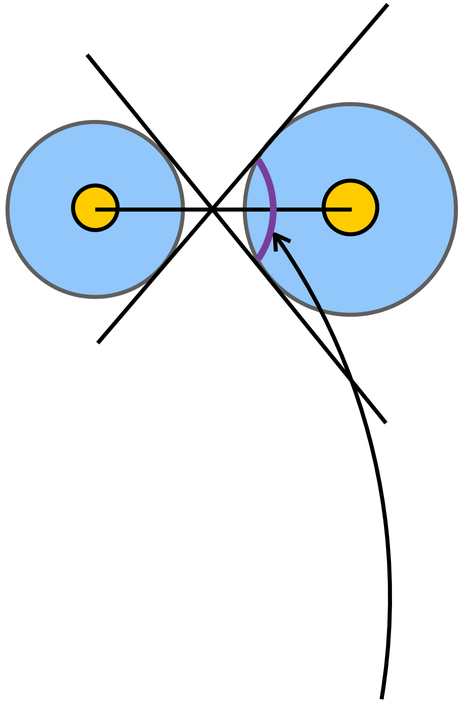


Почему?

Почему?



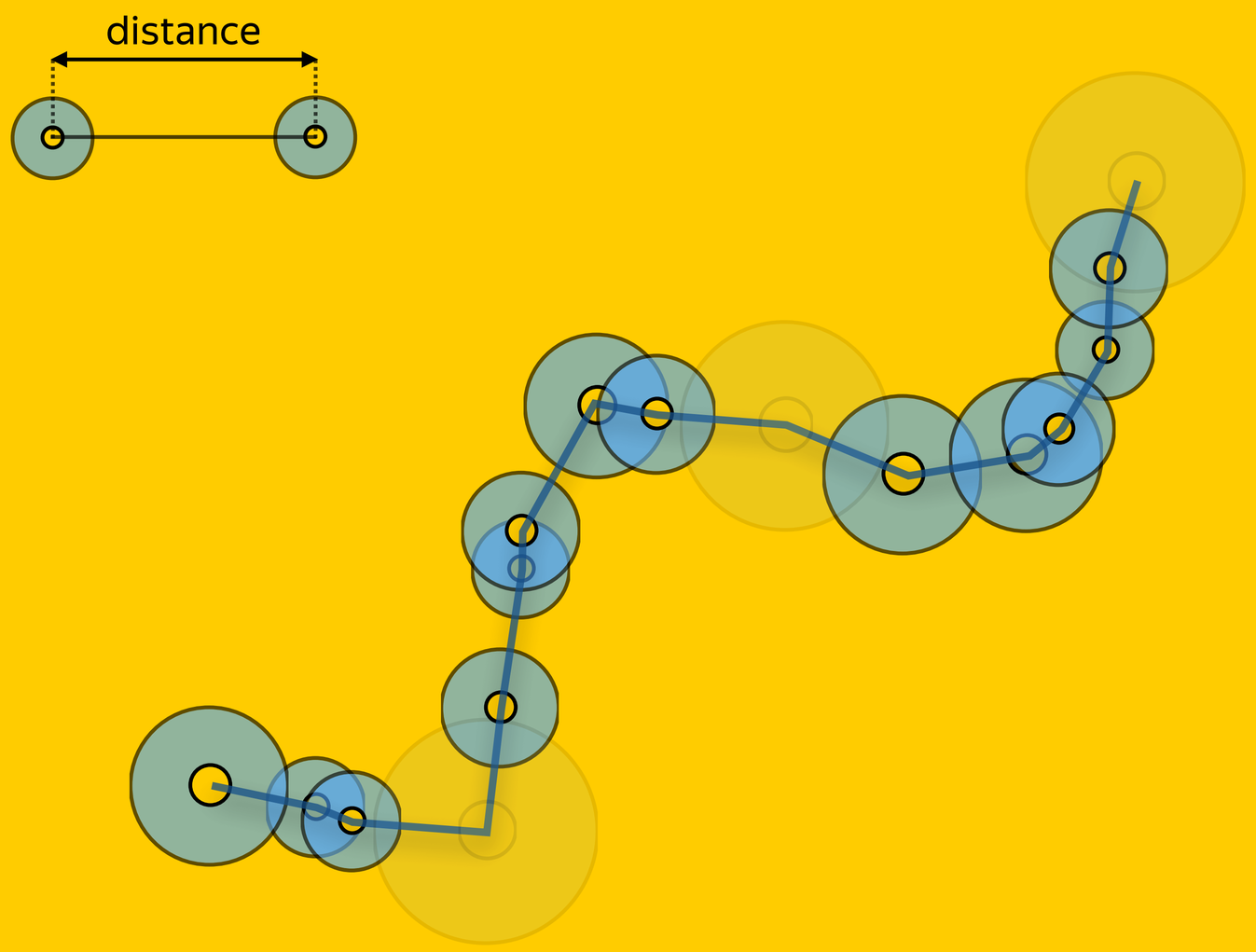
vs



погрешность < погрешность

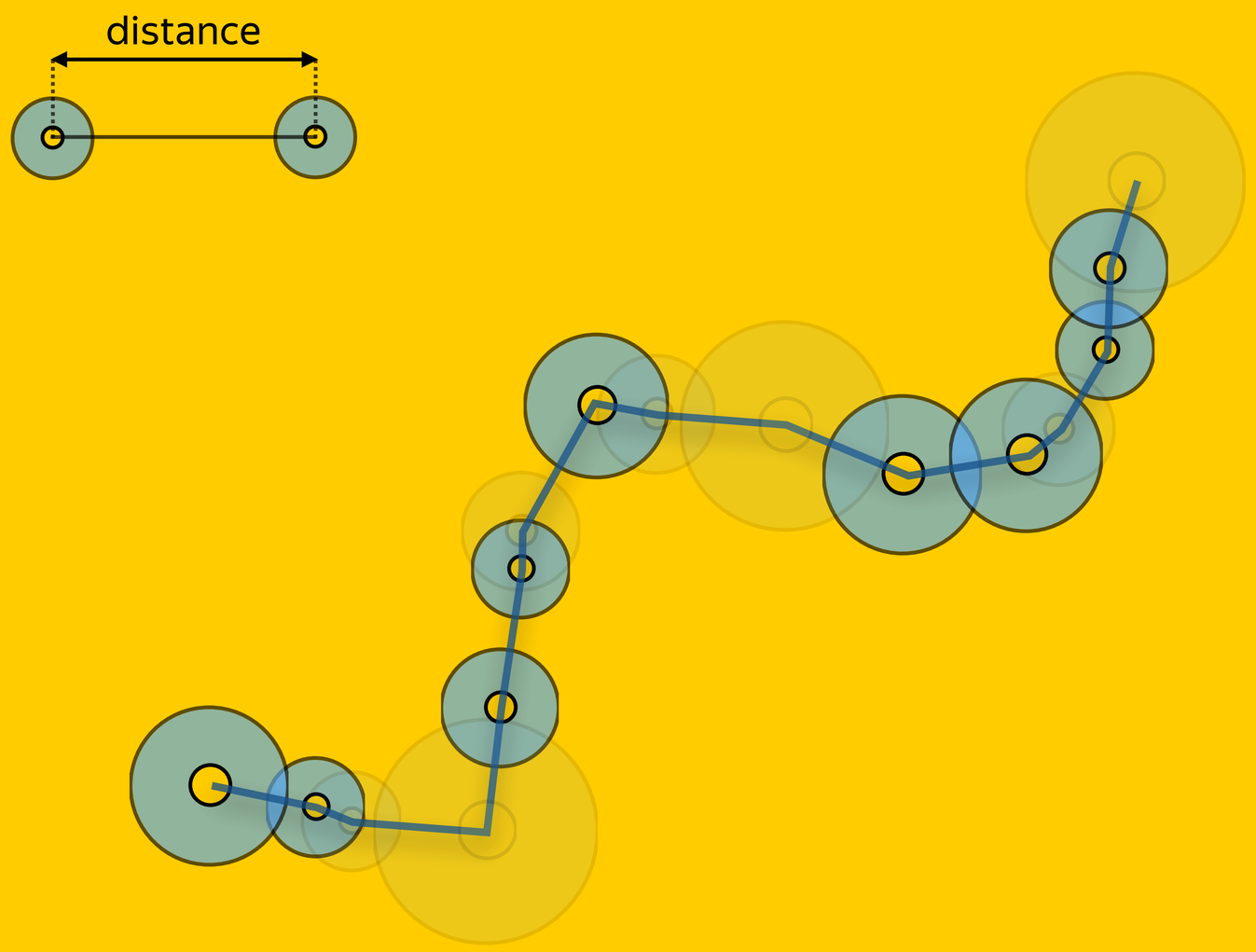
Фильтрация эстимейтов

- по точности географической координаты
- по расстоянию между эстимейтами

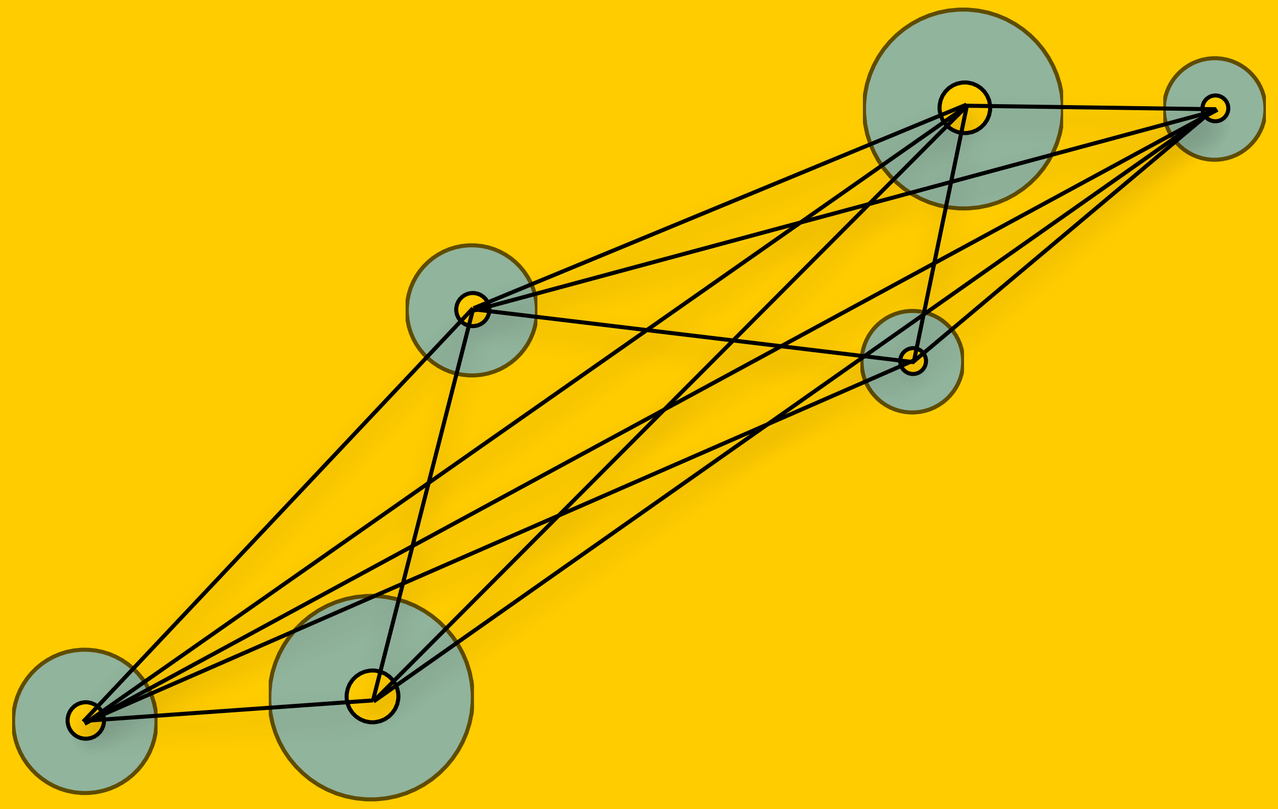


Фильтрация эстимейтов

- по точности географической координаты
- по расстоянию между эстимейтами

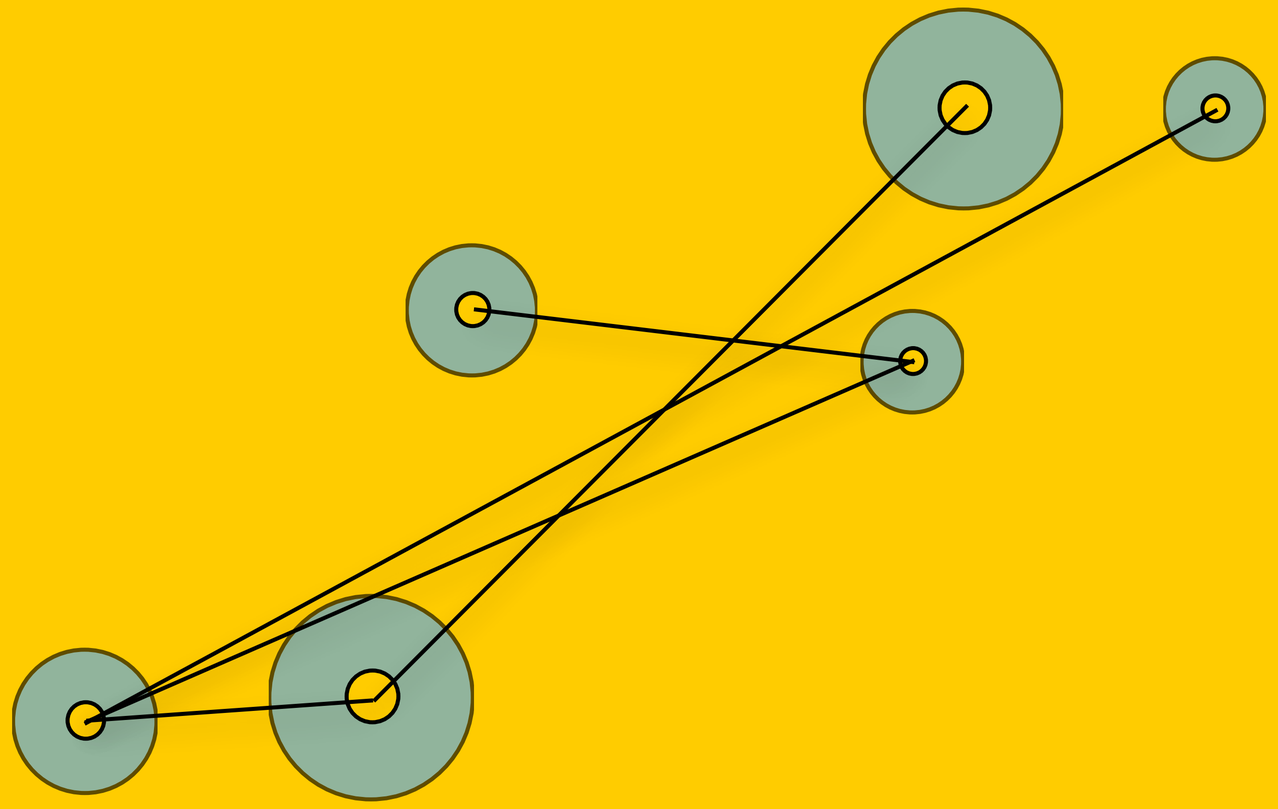


Выбор пар



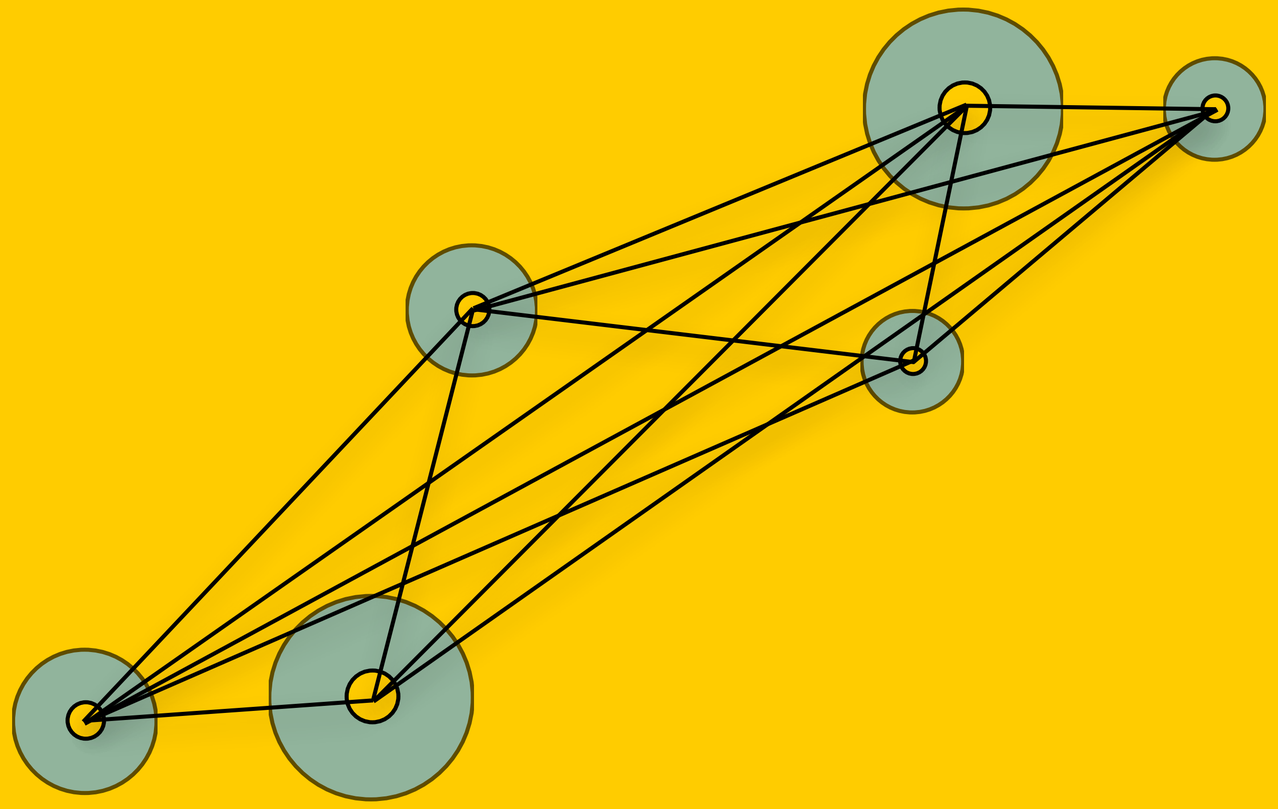
Выбор пар

› Случайные N процентов от всех пар



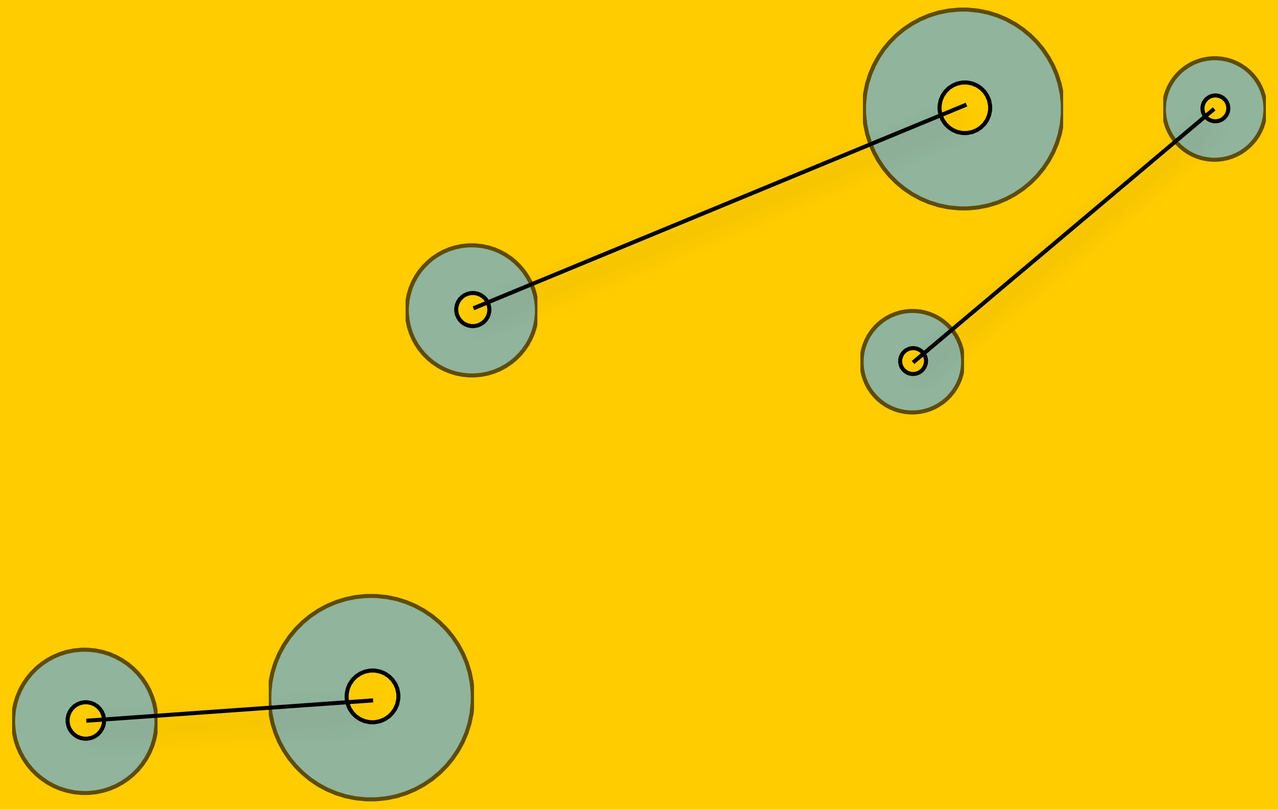
Выбор пар

› Случайные N процентов от всех пар

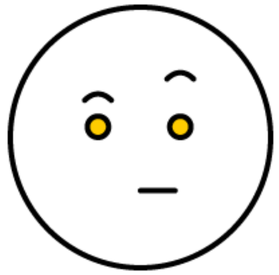


Выбор пар

- Случайные N процентов от всех пар
- Каждый эстимейт участвует только в одной паре



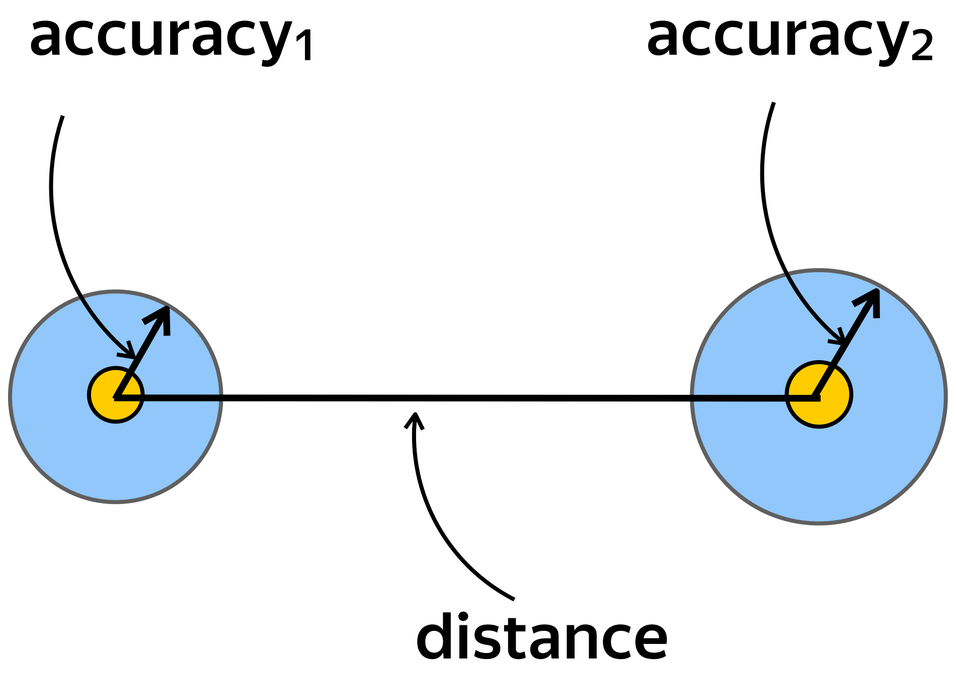
Должен быть способ определить «силу» пары…



Учет веса пар

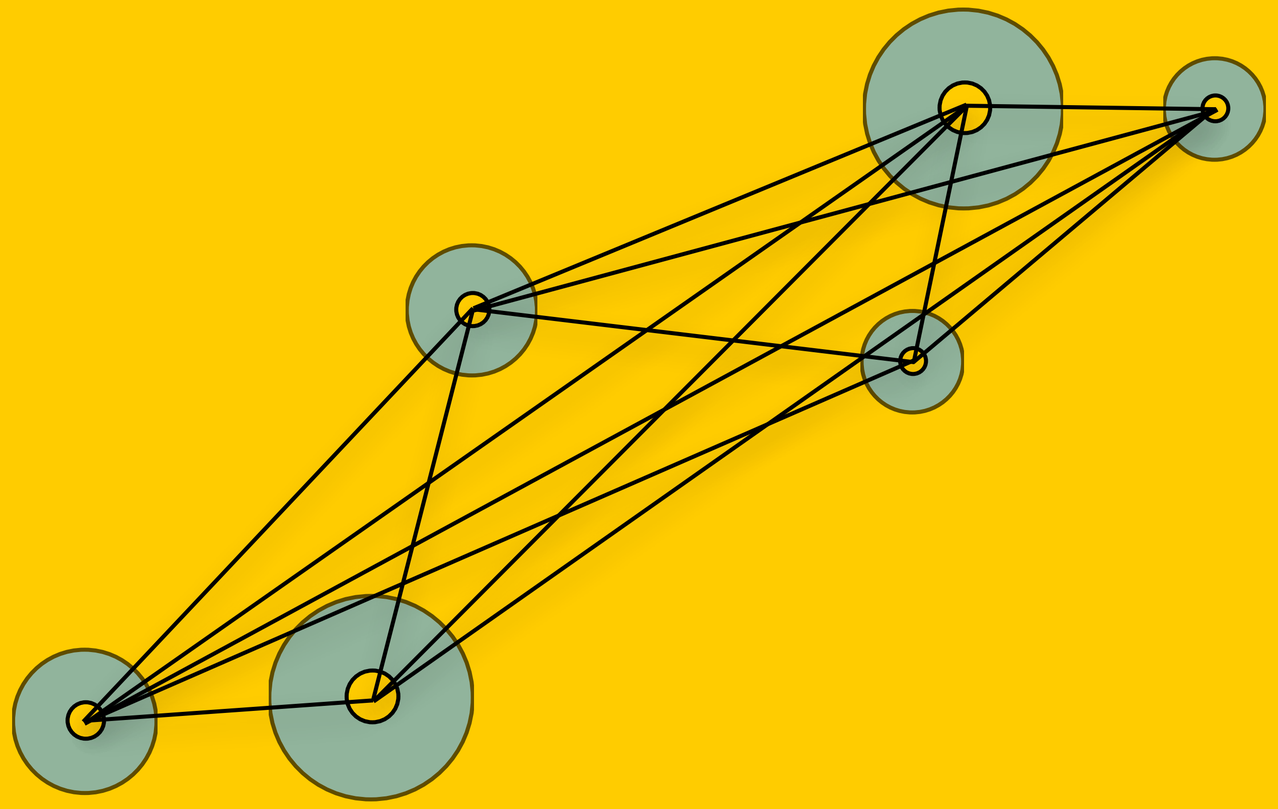
Вес пары – отношение расстояние между точками к сумме радиусов погрешности

weight = distance / (acc1 + acc2)



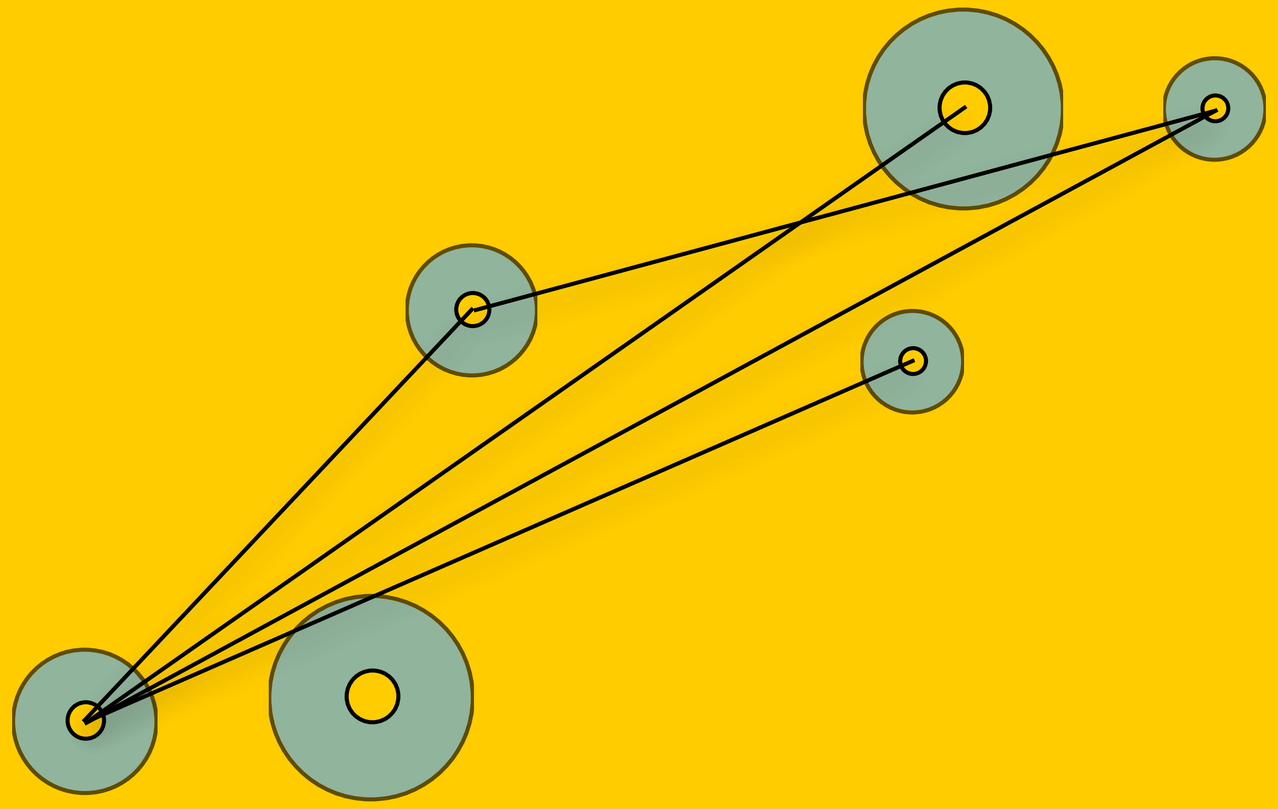
Выбор пар

- Случайные N процентов от всех пар
- Каждый эстимейт участвует только в одной паре



Выбор пар

- Случайные N процентов от всех пар
- Каждый эстимейт участвует только в одной паре
- Пары с наибольшим весом (20% лучших)



Спойлер

Выбор пар с наибольшим весом не оправдал ожиданий

Тестирование алгоритма

Способы тестирования

Способы тестирования

- Юнит тесты

Способы тестирования

Юнит тесты

- проверка определения координаты по смещению в ARKit

Способы тестирования

Юнит тесты

- проверка определения координаты по смещению в ARKit
- проверка расчета угла между двумя эстимейтами

Способы тестирования

- Юнит тесты
  - проверка определения координаты по смещению в ARKit
  - проверка расчета угла между двумя эстимейтами
- Полевые испытания

Способы тестирования

Юнит тесты

- проверка определения координаты по смещению в ARKit
- проверка расчета угла между двумя эстимейтами

Полевые испытания

- предварительное задание неправильного направления осей с последующей проверкой корректной работы алгоритма

Способы тестирования

Юнит тесты

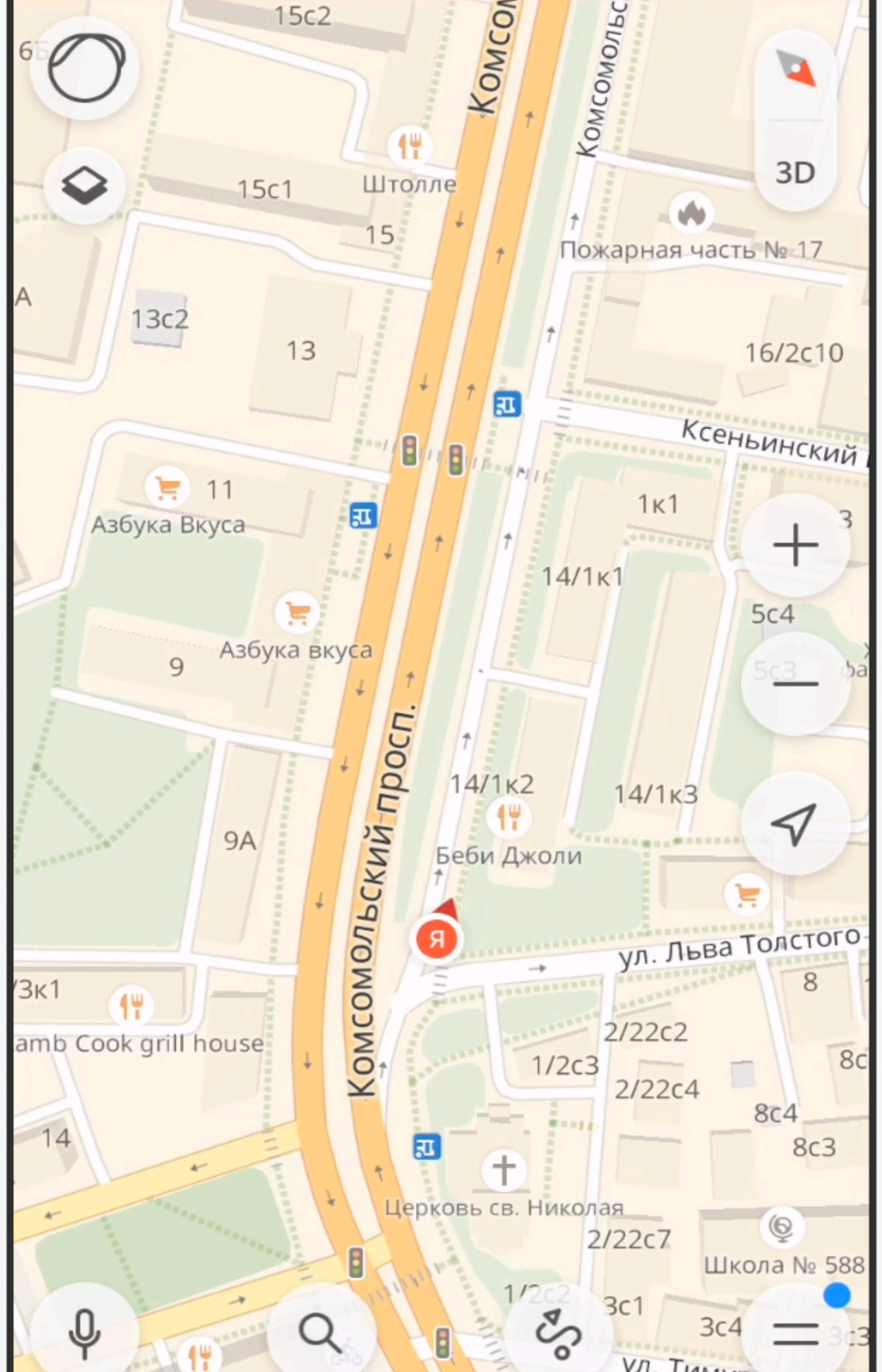
- проверка определения координаты по смещению в ARKit
- проверка расчета угла между двумя эстимейтами

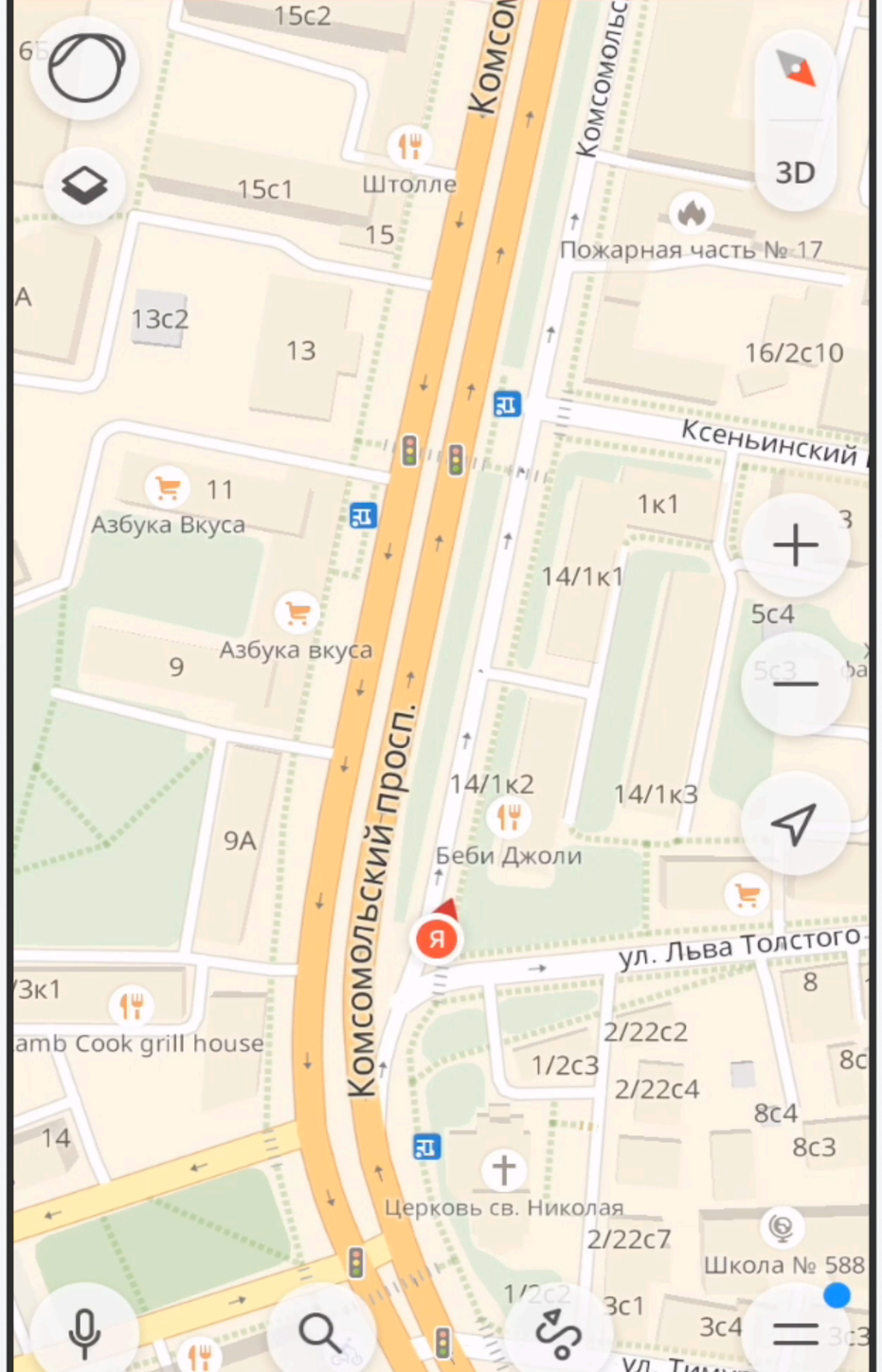
Полевые испытания

- предварительное задание неправильного направления осей с последующей проверкой корректной работы алгоритма
- сравнение треков по данным CoreLocation и конвертированных в географические к-ты положений в ARKit

Юнит тесты

```
func testPresiceTrueNorthWhenMovingFromWorkDownToTimuraFrunzeByCLAndMovingNorthByAR()
{
    let e1 = SceneLocationEstimate(
        location: CLLocation(latitude: 55.733953, longitude: 37.589836),
        position: SCNVector3Make(0, 0, 3))
    let e2 = SceneLocationEstimate(
        location: CLLocation(latitude: 55.733499, longitude: 37.590571),
        position: SCNVector3Make(0, 0, 2))

    let angle = correctionAngleByBearing(e1, e2)
    XCTAssertEqual(angle, -138.6, accuracy: 2.0)
}
```


15с2
Комсомольский просп.
Комсомольс
3D
15с1
Штолле
15
Пожарная часть № 17
13с2
13
16/2с10
Ксеньинский
11
Азбука Вкуса
1к1
14/1к1
5с4
Азбука вкуса
9
14/1к2
14/1к3
9А
Беби Джоли
ул. Льва Толстого
8
3к1
amb Cook grill house
2/22с2
1/2с3
2/22с4
8с4
8с3
14
Церковь св. Николая
2/22с7
Школа № 588
3с1
3с4


15с2
Комсомольский просп.
Комсомольс
3D
15с1
Штолле
15
Пожарная часть № 17
13с2
13
16/2с10
Ксеньинский
11
Азбука Вкуса
1к1
14/1к1
5с4
Азбука вкуса
9
14/1к2
14/1к3
9А
Беби Джоли
ул. Льва Толстого
8
3к1
amb Cook grill house
2/22с2
1/2с3
2/22с4
8с4
8с3
14
Церковь св. Николая
2/22с7
Школа № 588
3с1
3с4

Адрес или объект
Найти
Пробки
Слои
Парк культуры
Школа № 1253
Комсомольский просп.
Крымский пр-д
ул. Тимура Фрунзе
Большой Чудов пер.
ул. Льва Толстого
Фрунзенская наб.
Институт Конфуция МГЛУ
Папа Джонс
Азбука Вкуса
Акрополь
Альфа-Банк
БЦ Мамонтов
Prime
Два кренделя
Яндекс
Полуев-Дент
ЦНИИС и ЧЛХ
ЦНИИ стоматологии и челюстно-лицевой хирургии
Инвитро
NailsProfi
БЦ Морозов
Starbucks
Лошадь Клоун
ЕвроКлиник
Среди своих
Школа № 588
Интеробразование
Бистро
Fashion color
Fin-palki.ru
Почта № 119021
ДПС № 5
Открыть в Яндекс.Картах
30 м
© Яндекс Условия использования
— CoreLocation
— AR

В результате

В результате

- Отрисовываем только видимый участок маршрута

В результате

- Отрисовываем только видимый участок маршрута
- Используем лучший эстимейт для уменьшения погрешности определения позиции объектов на сцене

В результате

- Отрисовываем только видимый участок маршрута
- Используем лучший эстимейт для уменьшения погрешности определения позиции объектов на сцене
- Корректируем направление осей

Дальнейшие улучшения

Дальнейшие улучшения

- Использовать геометрию зданий для показа только видимой части маршрута

Дальнейшие улучшения

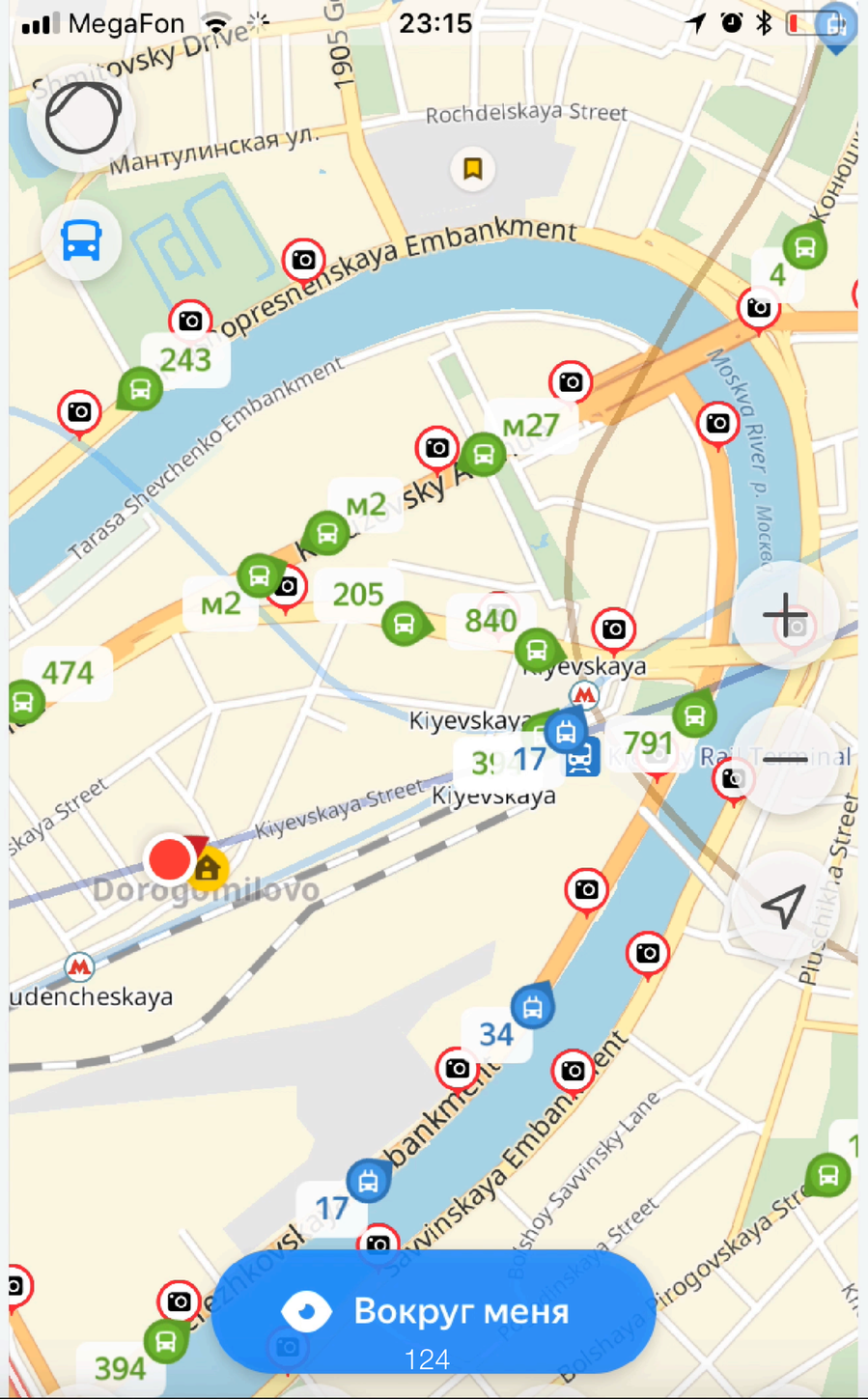
- Использовать геометрию зданий для показа только видимой части маршрута
- Улучшение алгоритма корректировки системы к-т ARKit

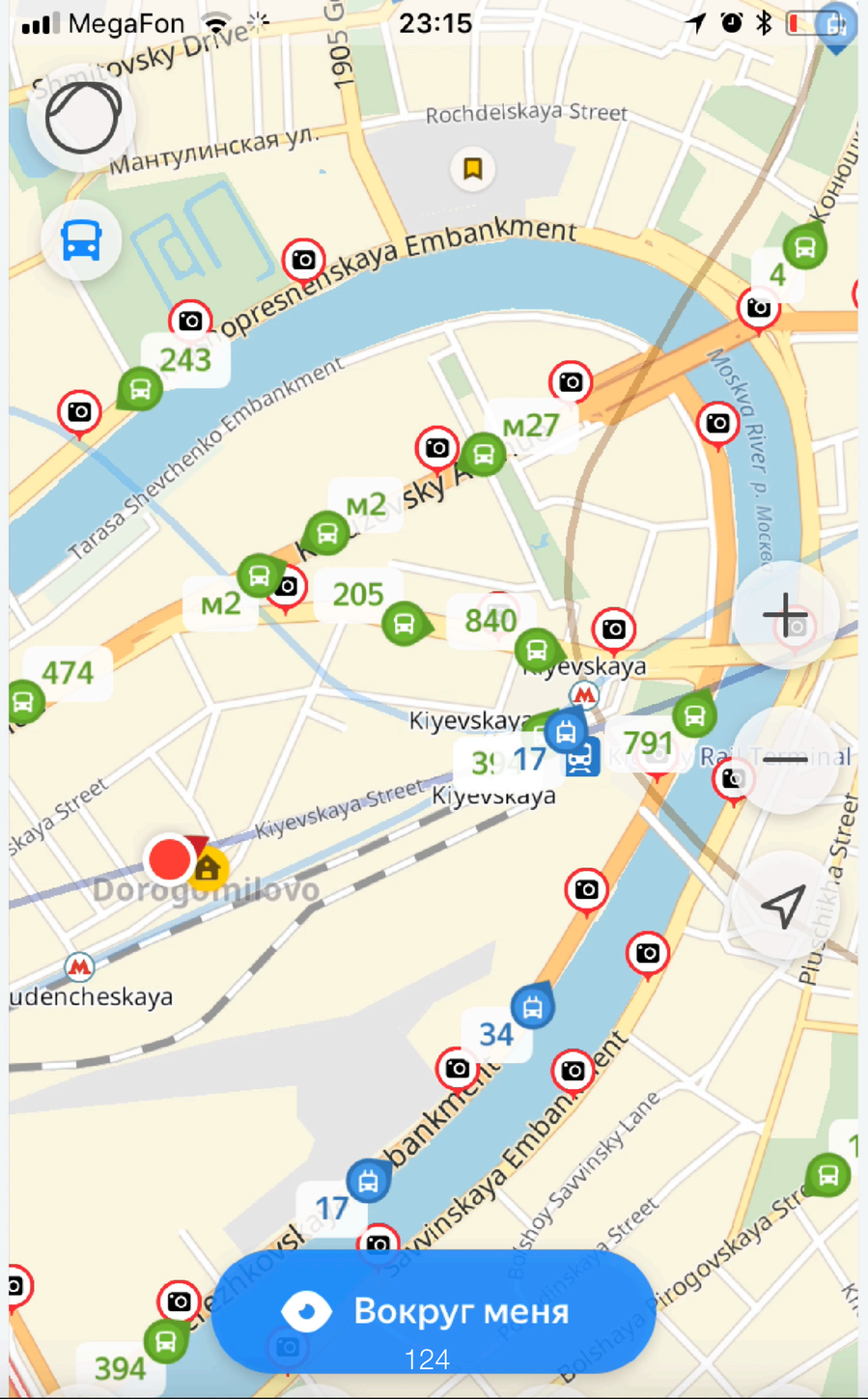
Дальнейшие улучшения

- Использовать геометрию зданий для показа только видимой части маршрута
- Улучшение алгоритма корректировки системы к-т ARKit
- etc...

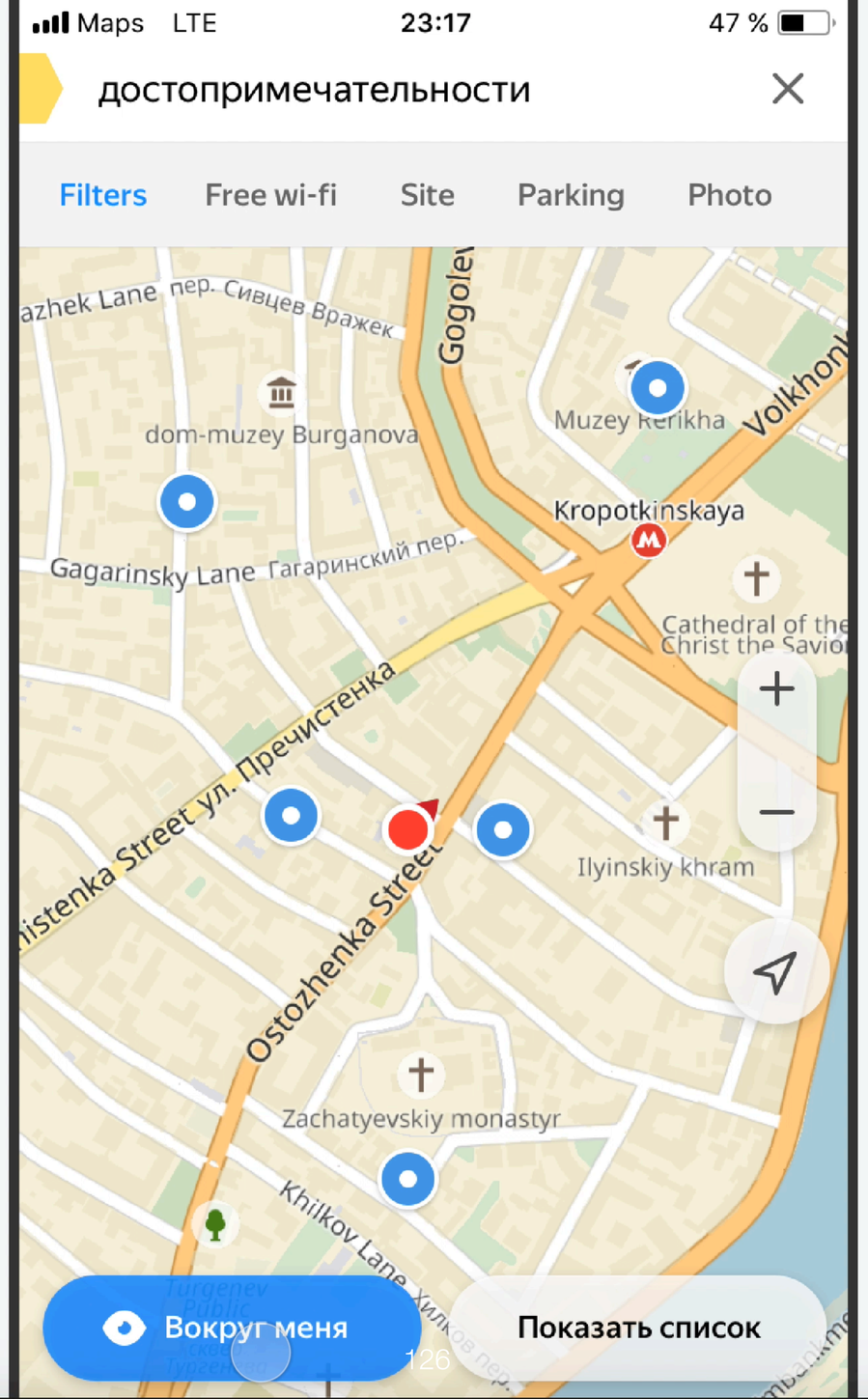
В планах

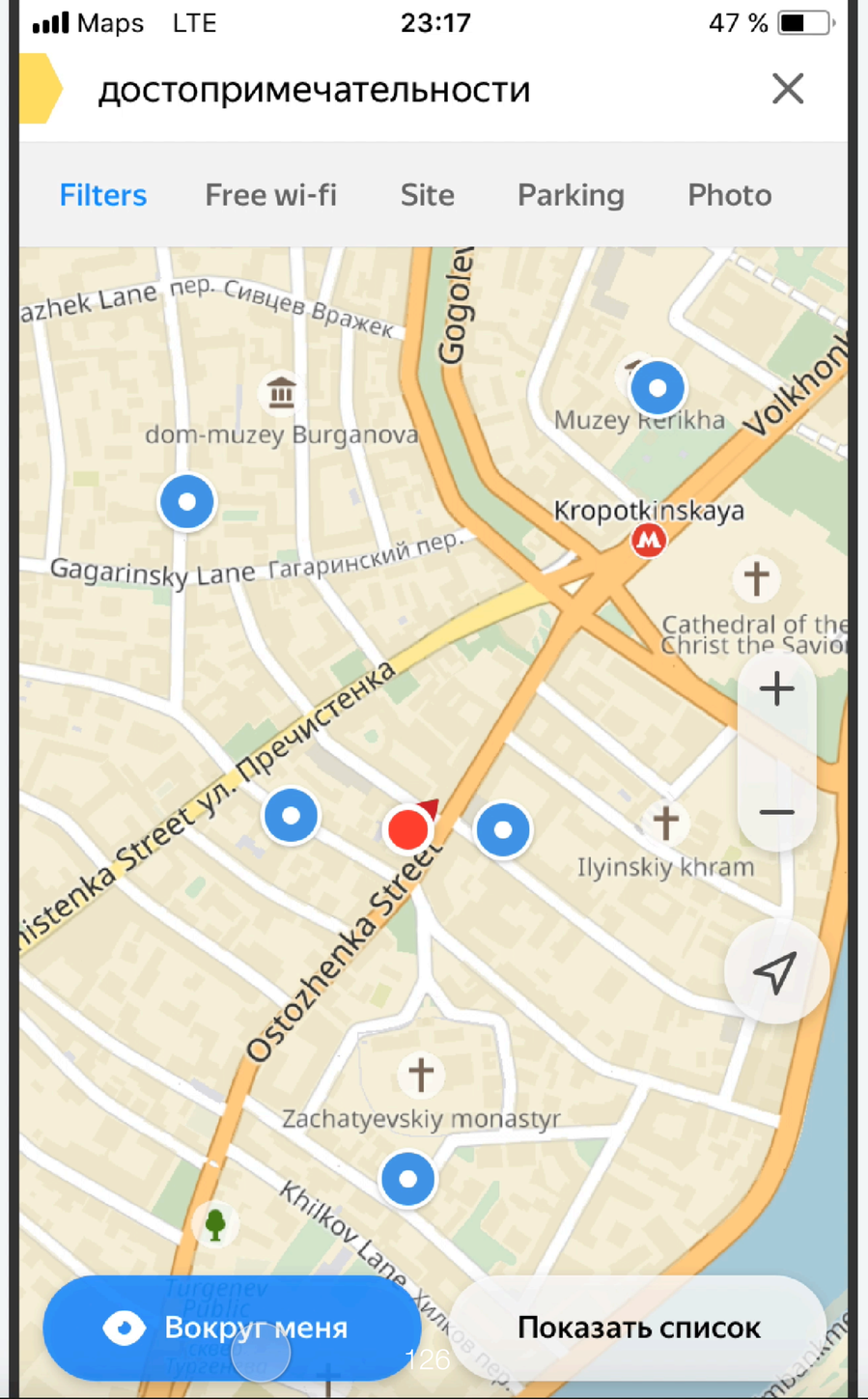
Улица Петра Алексеева
Petra Alekseyeva Street


MegaFon
23:15
Rochdelskaya Street
Мантулинская ул.
Kopresnenskaya Embankment
Tarasa Shevchenko Embankment
Moskva River
243
4
M27
M2
205
840
474
791
17
34
394
Kiyevskaya
Kiyevskaya Street
Dorogomilovo
Savvinskaya Embankment
Bolshoy Savvinsky Lane
Bolshaya Pirogovskaya Street
Вокруг меня
124


MegaFon
23:15
Rochdelskaya Street
Мантулинская ул.
Kopresnenskaya Embankment
Tarasa Shevchenko Embankment
Moskva River
243
4
M27
M2
205
840
474
791
17
34
394
Kiyevskaya
Kiyevskaya Street
Dorogomilovo
Savvinskaya Embankment
Bolshoy Savvinsky Lane
Bolshaya Pirogovskaya Street
Вокруг меня
124

СЕРГЕЮ ЕСЕНИНУ


Maps LTE 23:17 47 %
достопримечательности
Filters
Free wi-fi
Site
Parking
Photo
azhek Lane пер. Сивцев Вражек
Gogolev
Volkhonk
dom-muzey Burganova
Muzey Serikha
Kropotkinskaya
Gagarinsky Lane Гагаринский пер.
Cathedral of the Christ the Savior
istenka Street ул. Пречистенка
Ostozhenka Street
Ilyinskiy khram
Zachatyevskiy monastyr
Khilkov Lane
Вокруг меня
Показать список


Maps LTE 23:17 47 %
достопримечательности
Filters
Free wi-fi
Site
Parking
Photo
azhek Lane пер. Сивцев Вражек
Gogolev
Volkhonk
dom-muzey Burganova
Muzey Serikha
Kropotkinskaya
Gagarinsky Lane Гагаринский пер.
Cathedral of the Christ the Savior
istenka Street ул. Пречистенка
Ostozhenka Street
Ilyinskiy khram
Zachatyevskiy monastyr
Khilkov Lane
Вокруг меня
Показать список

Статистика

Использование AR в картах

Использование AR в картах

11% пользователей использовали пешеходное ведение с AR за период с момента запуска

Использование AR в картах

- 11% пользователей использовали пешеходное ведение с AR за период с момента запуска
- 31% пользователей, из тех кто переходил в пешеходное ведение использовали AR хотя бы раз

Частота использования за последний месяц

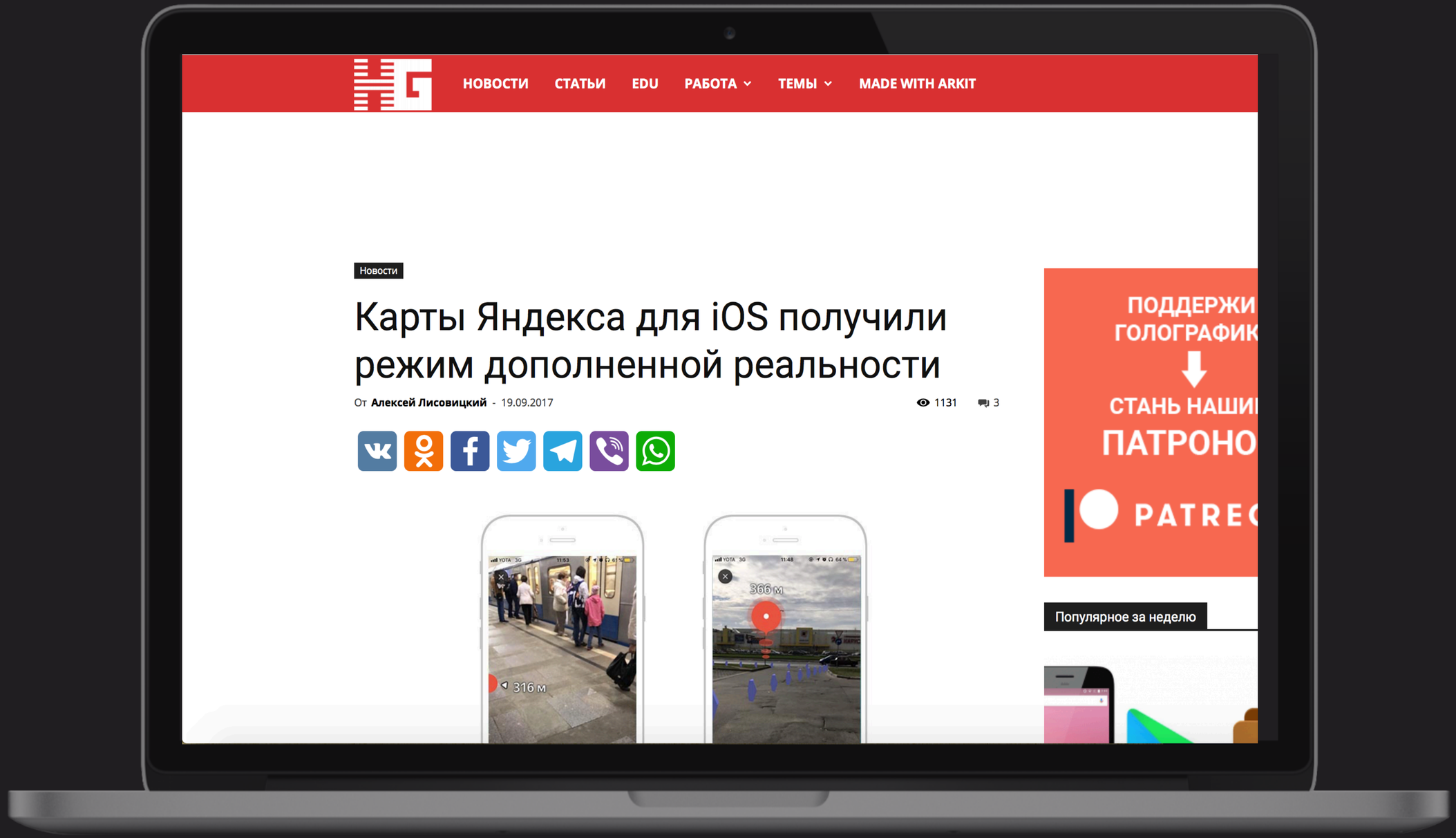
Сколько раз использовали AR в пешеходной маршрутизации	Доля пользователей
1	14,8 %
2	4,1 %
3	1,8 %
4	0,8 %
5	0,4 %

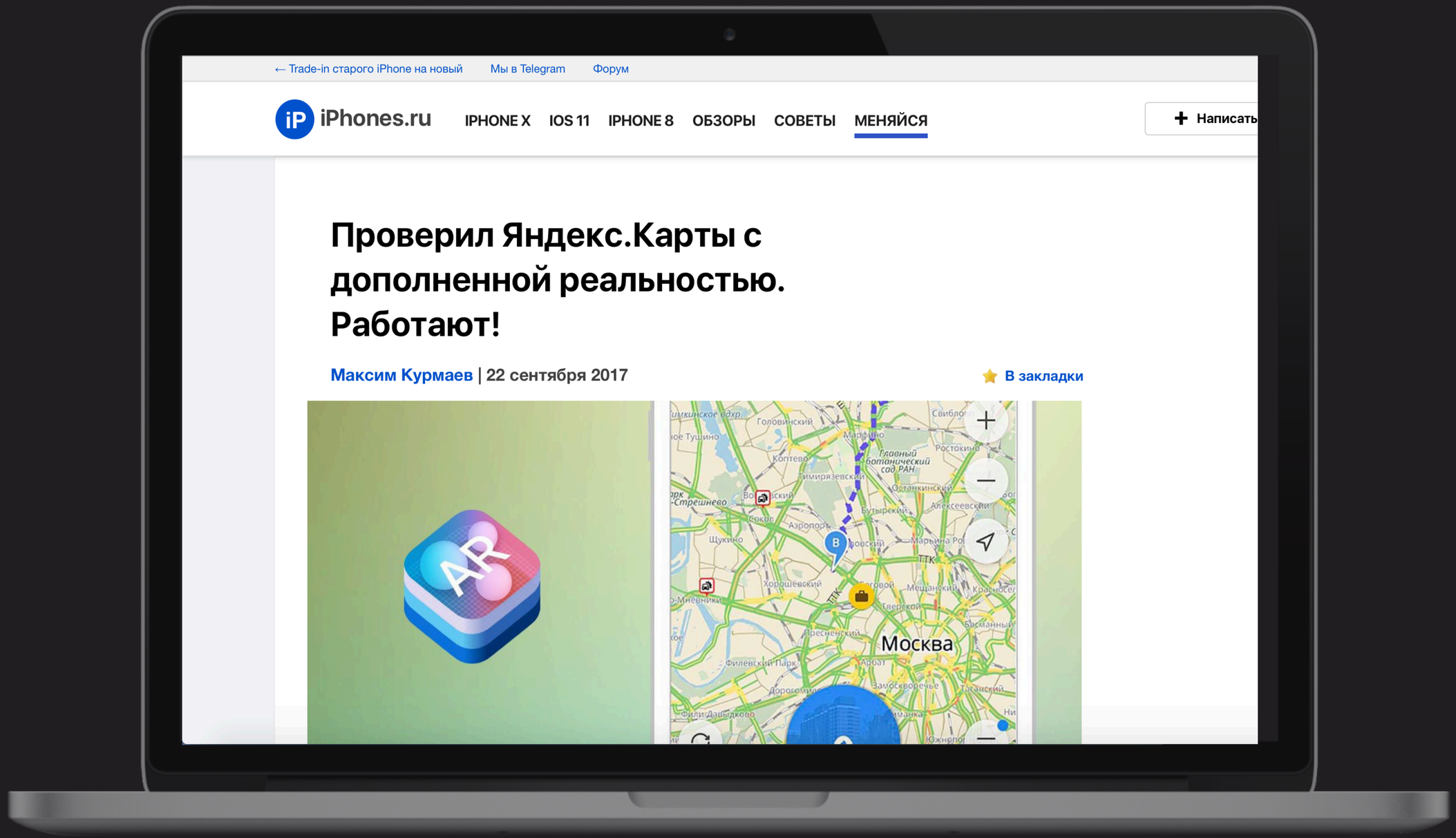
Постоянное использование

Среди пользователей, которые переходили в пешеходное ведение больше 5 раз

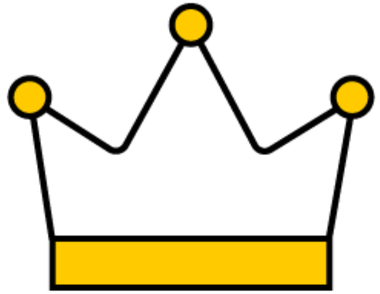
- 50% переходили в AR в 1 маршруте из 10
- 25% - в 3 маршрутах из 10
- 9% - в 5 маршрутах из 10
- 1% - в 9 маршрутах из 10

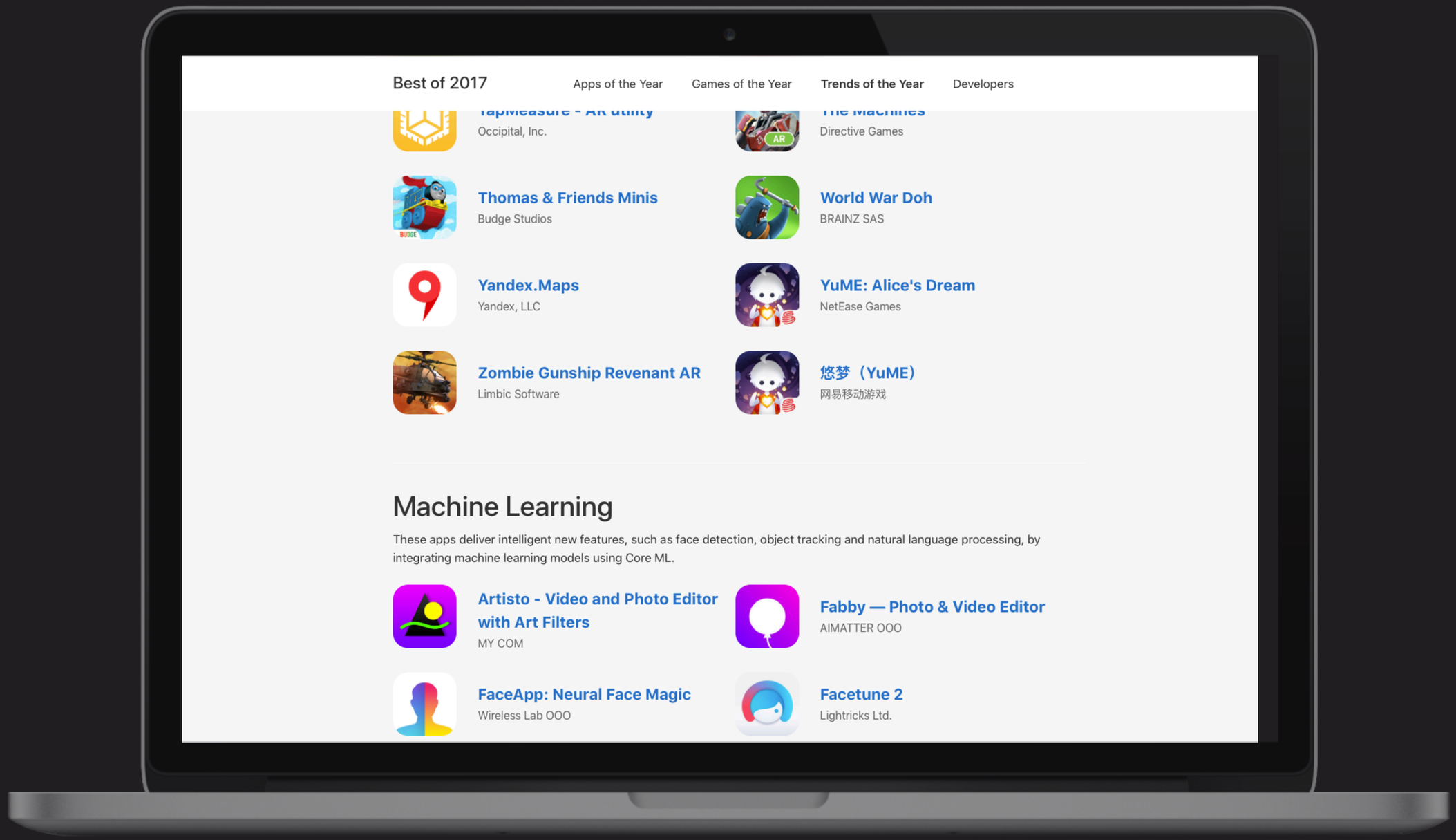
ХАЙПАНЕМ НЕМНОЖЕЧКО?


НОВОСТИ
СТАТЬИ
EDU
РАБОТА
ТЕМЫ
MADE WITH ARKIT
Новости
Карты Яндекса для iOS получили режим дополненной реальности
От Алексей Лисовицкий - 19.09.2017
1131
3
316 м
366 м
ПОДДЕРЖИ
ГОЛОГРАФИК
СТАНЬ НАШИ
ПАТРОНО
PATREO
Популярное за неделю


← Trade-in старого iPhone на новый
Мы в Telegram
Форум
iPhones.ru
IPHONE X
IOS 11
IPHONE 8
ОБЗОРЫ
СОВЕТЫ
МЕНЯЙСЯ
+ Написать
Проверил Яндекс.Карты с дополненной реальностью. Работают!
Максим Курмаев | 22 сентября 2017
В закладки
AR
Москва

Попали в список лучших приложений с AR в AppStore




Best of 2017
Apps of the Year
Games of the Year
Trends of the Year
Developers
Occipital, Inc.
Directive Games
Thomas & Friends Minis
Budge Studios
World War Doh
BRAINZ SAS
Yandex.Maps
Yandex, LLC
YuME: Alice's Dream
NetEase Games
Zombie Gunship Revenant AR
Limbic Software
悠梦（YuME）
网易移动游戏
Machine Learning
These apps deliver intelligent new features, such as face detection, object tracking and natural language processing, by integrating machine learning models using Core ML.
Artisto - Video and Photo Editor with Art Filters
MY COM
Fabby — Photo & Video Editor
AIMATTER OOO
FaceApp: Neural Face Magic
Wireless Lab OOO
Facetune 2
Lightricks Ltd.

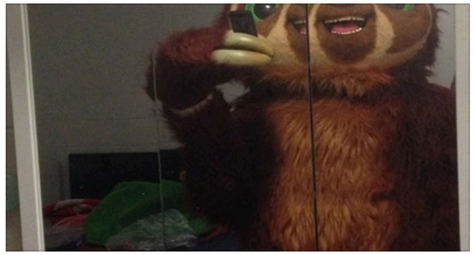
Блог команды разработки
Яндекс.Карт для **iOS**

КОМАНДА ВСЕ СТАТЬИ | ПРИЛОЖЕНИЕ

Особенности работы метода UITableView.scrollToRow

Метод scrollToRow(at:at:animated:) у UITableView при использовании в обычных условиях ведет себя предсказуемо и ожидаемо, подскролливая...

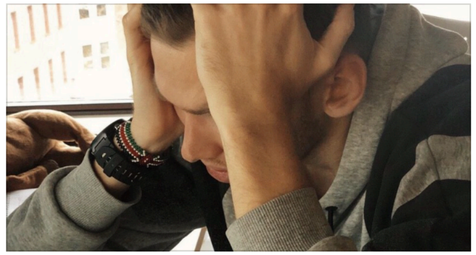
Артем Кириллов
Apr 12



Зеркало для ленивых

Всем нам приходилось сталкиваться с задачей: как определить, проинициализировано ли свойство класса, объявленное лениво.

Искандер Ядгаров
Apr 5



Расширение вложенного типа в другом файле

Эта короткая заметка откроет новую рубрику "Забавный баг". Вот один из тех, с которым мы столкнулись в языке Swift.

Артем Кириллов
Mar 30

Полезное и интересное

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

› сэмплы на github

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

- сэмплы на github
- формулы конвертации

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

- сэмплы на github
- формулы конвертации
- ARKit+CoreLocation проект и туториалы

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

- сэмплы на github
- формулы конвертации
- ARKit+CoreLocation проект и туториалы
- много интересного про ARKit

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

- сэмплы на github
- формулы конвертации
- ARKit+CoreLocation проект и туториалы
- много интересного про ARKit
- другие тоже корректируют направление оси

trimonovds@yandex-team.ru

t.me/trimonovds

Полезное и интересное

- сэмплы на github
- формулы конвертации
- ARKit+CoreLocation проект и туториалы
- много интересного про ARKit
- другие тоже корректируют направление оси
- история AR в одной картинке

trimonovds@yandex-team.ru

t.me/trimonovds