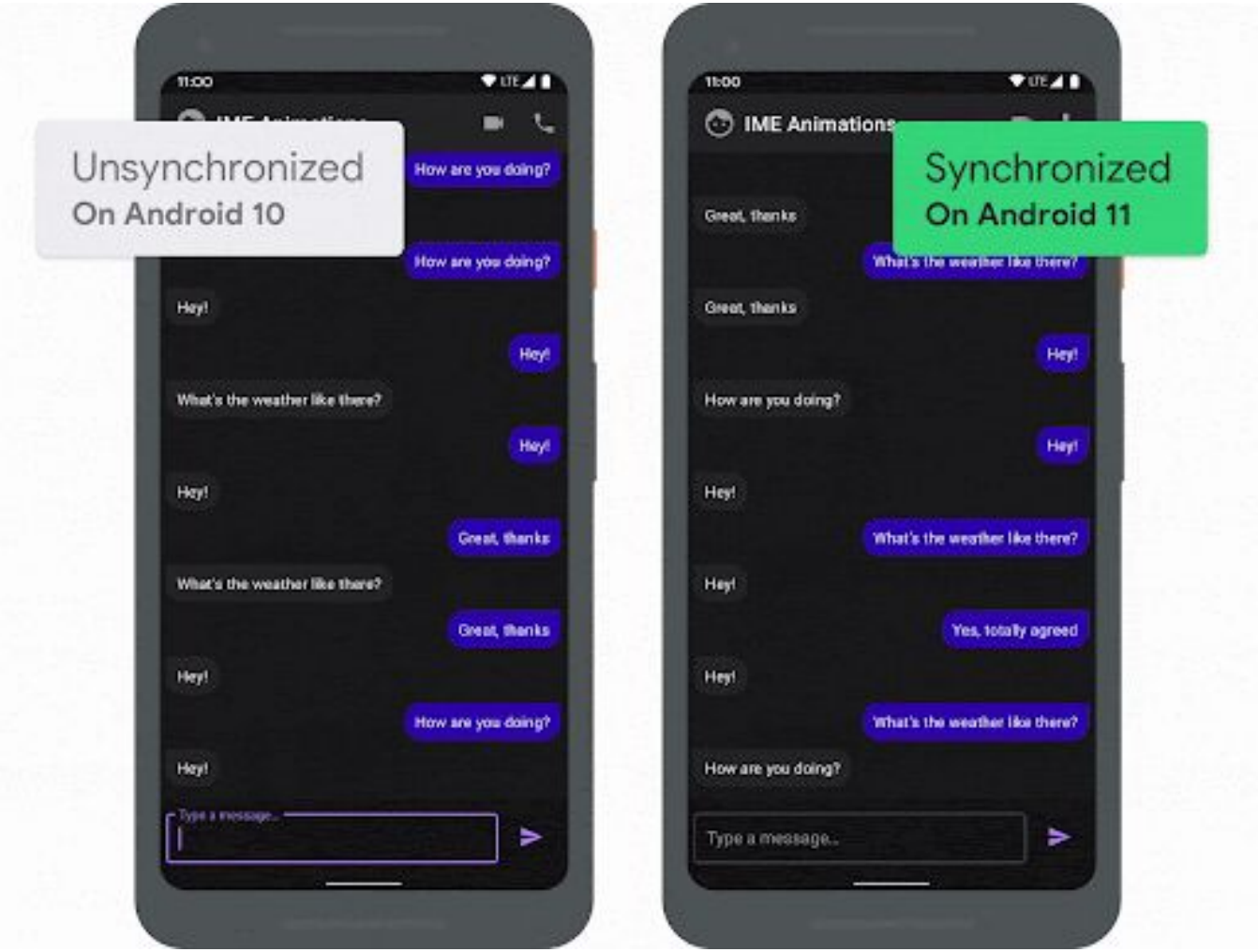
Android Insets и анимация клавиатуры

Обо мне

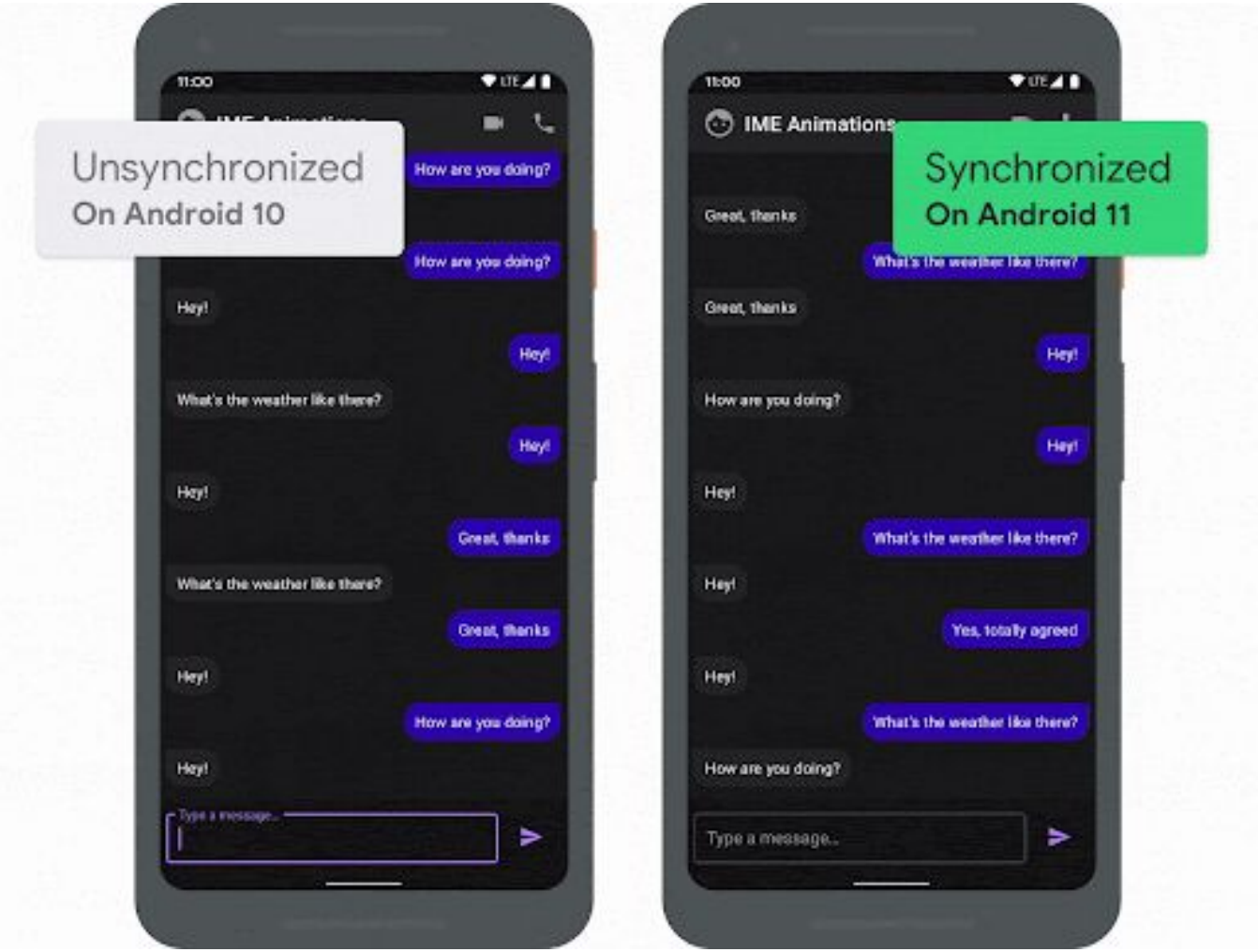
Михаил Гуревич

1. Начал разрабатывать под Android с 2011
2. Работал как в аусорс, так и в продуктах
3. Прошел путь от Junior до TeamLead
4. Сейчас Senior Android Engineer в компании, которая помогает барберам и клиентам найти друг друга - GetSquire.

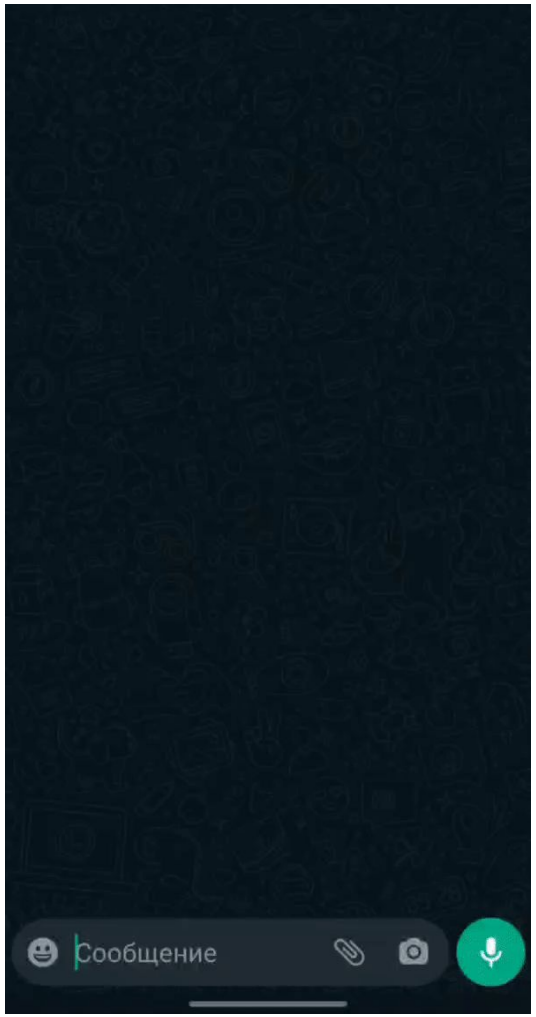
Фича начиная с Android 11

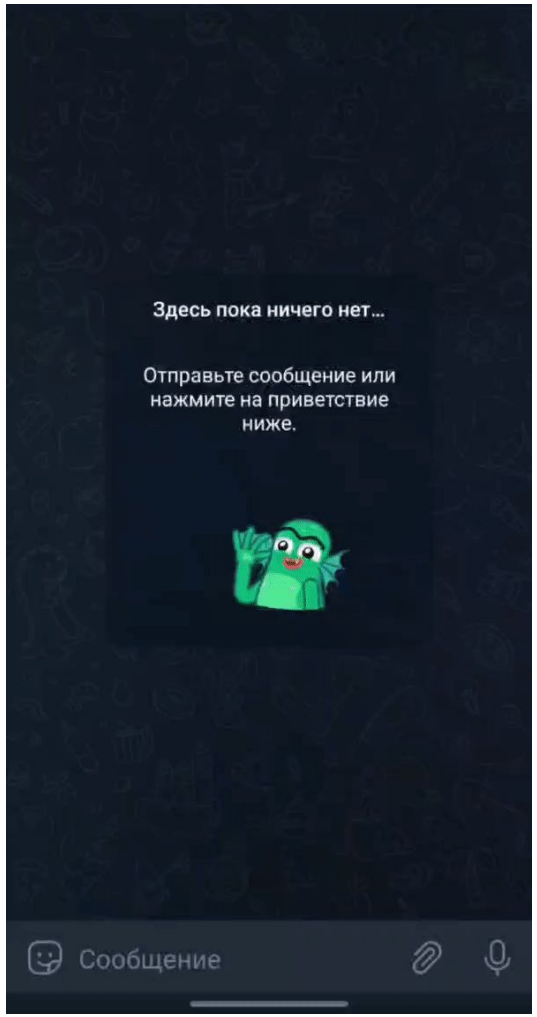


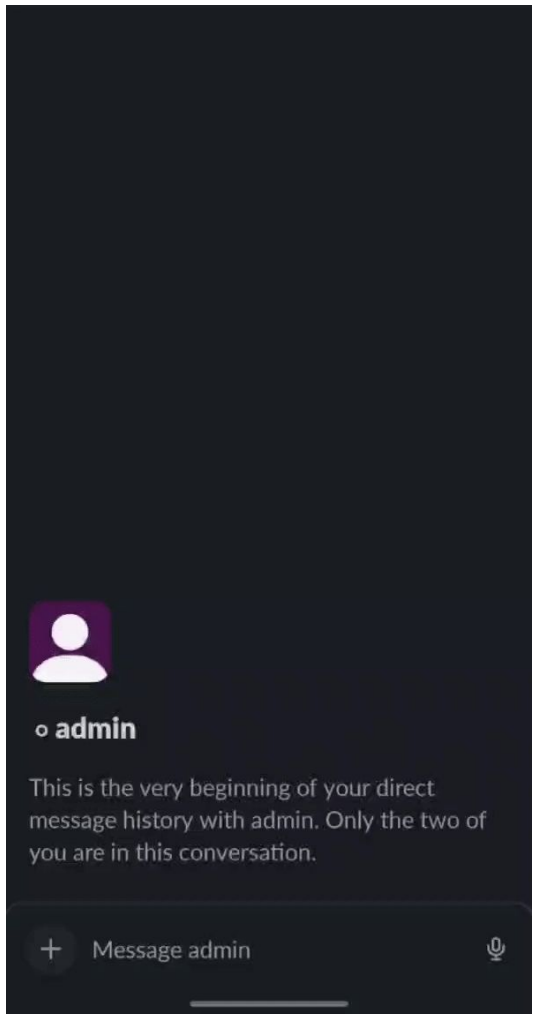
Фича начиная с Android ~~11~~ 5 (Compat)

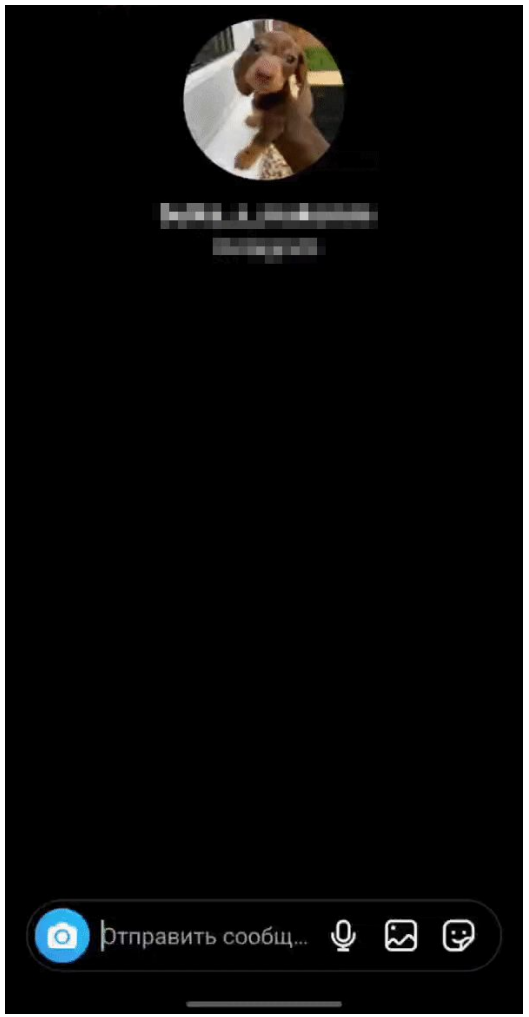


Примеры приложений









План доклада

План доклада

1. Узнаем какую проблему решают Insets

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры

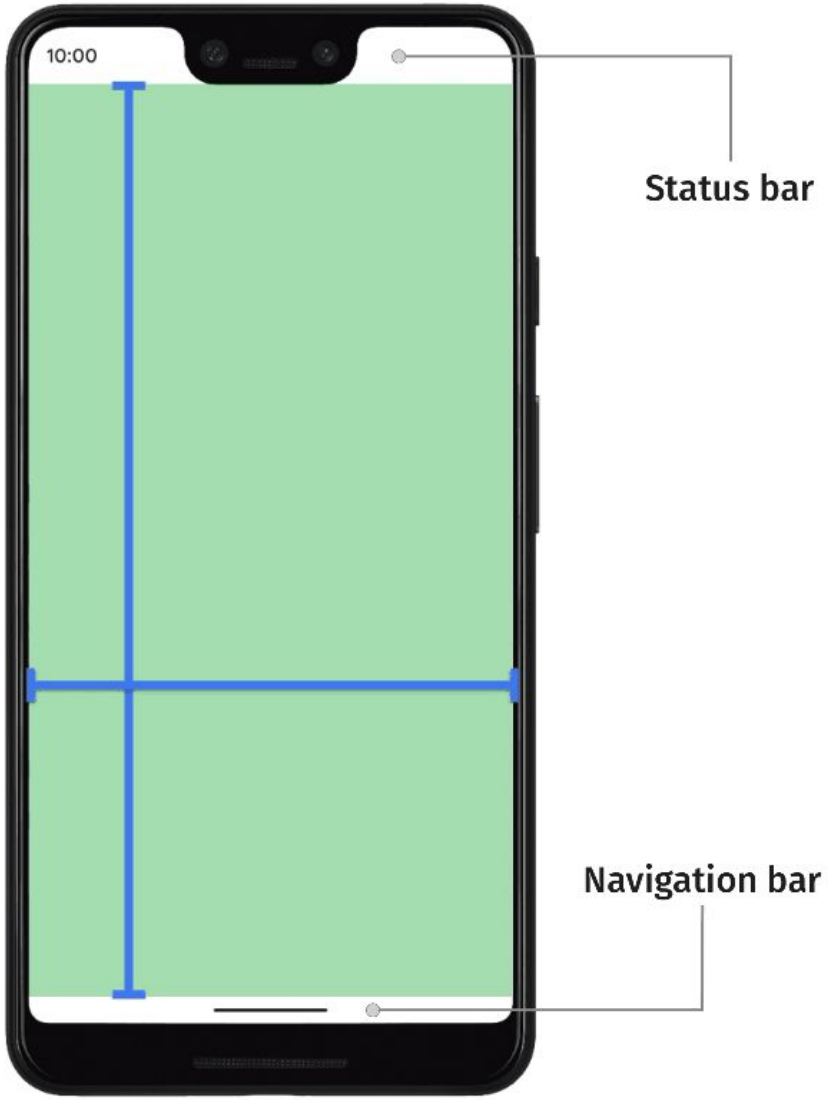
План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

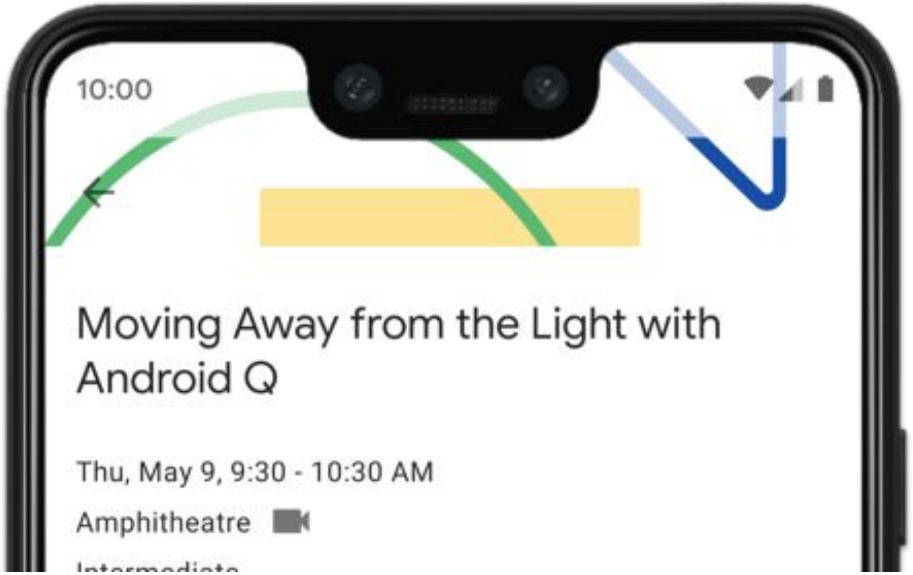
План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

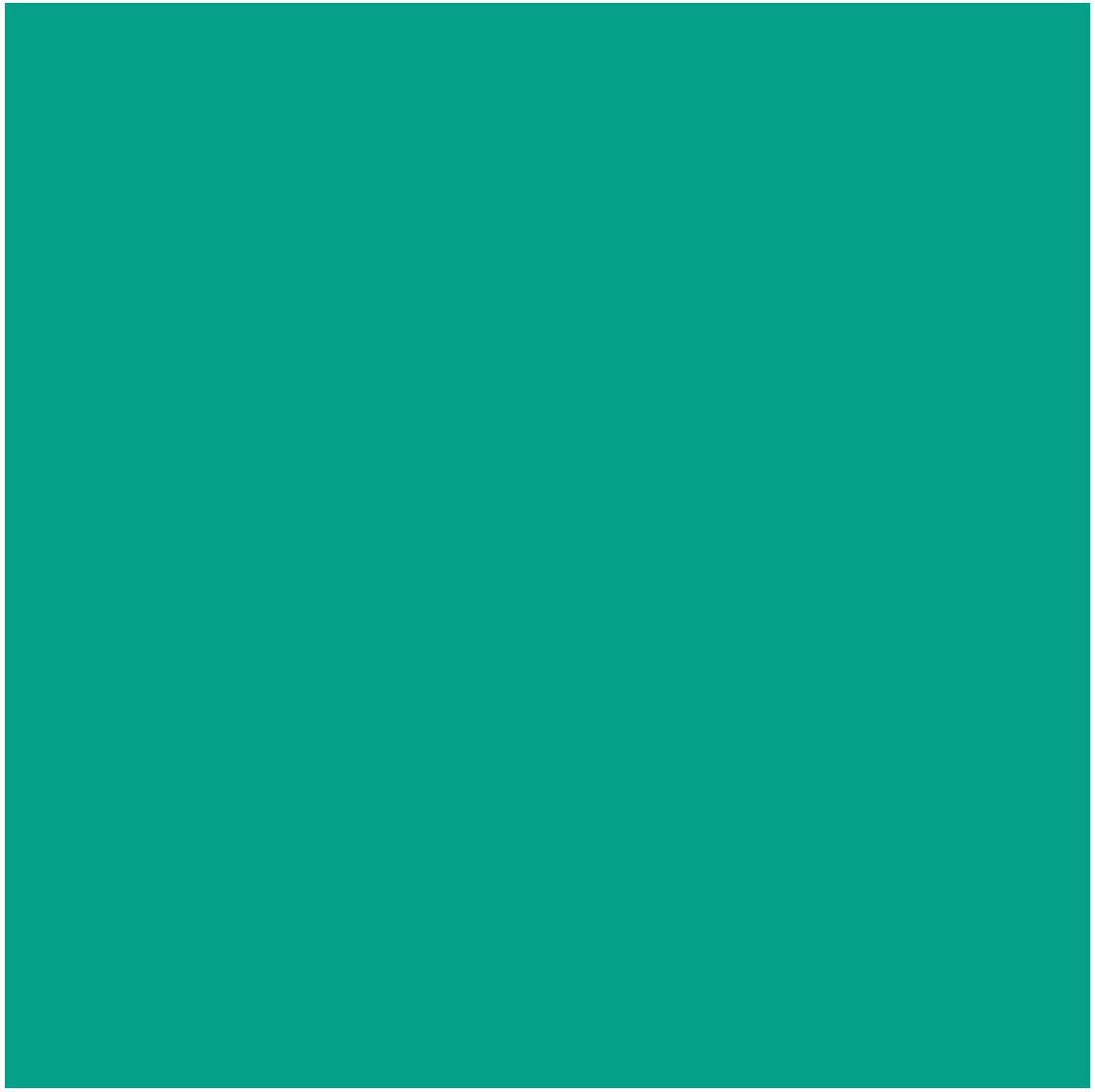
Edge-to-Edge



Edge-to-Edge


10:00
Moving Away from the Light with Android Q
Thu, May 9, 9:30 - 10:30 AM
Amphitheatre

Gesture Navigation



Insets

Это api, которое позволяет узнать:

1. О текущем наложении системного UI
2. О возможном наложении системного UI

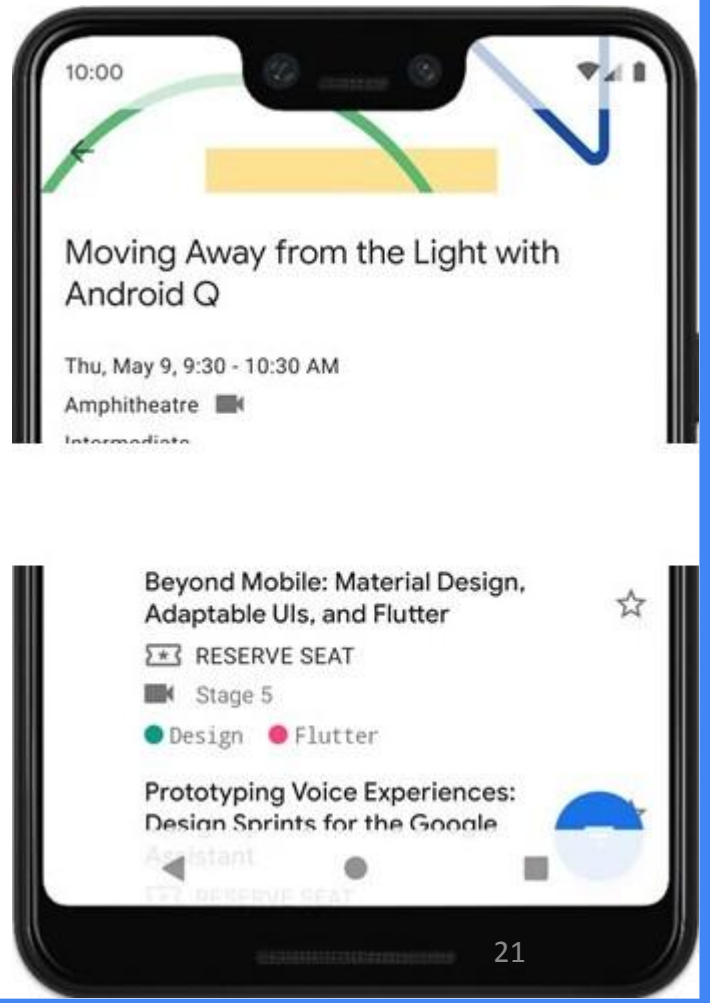
План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

Основные виды Insets

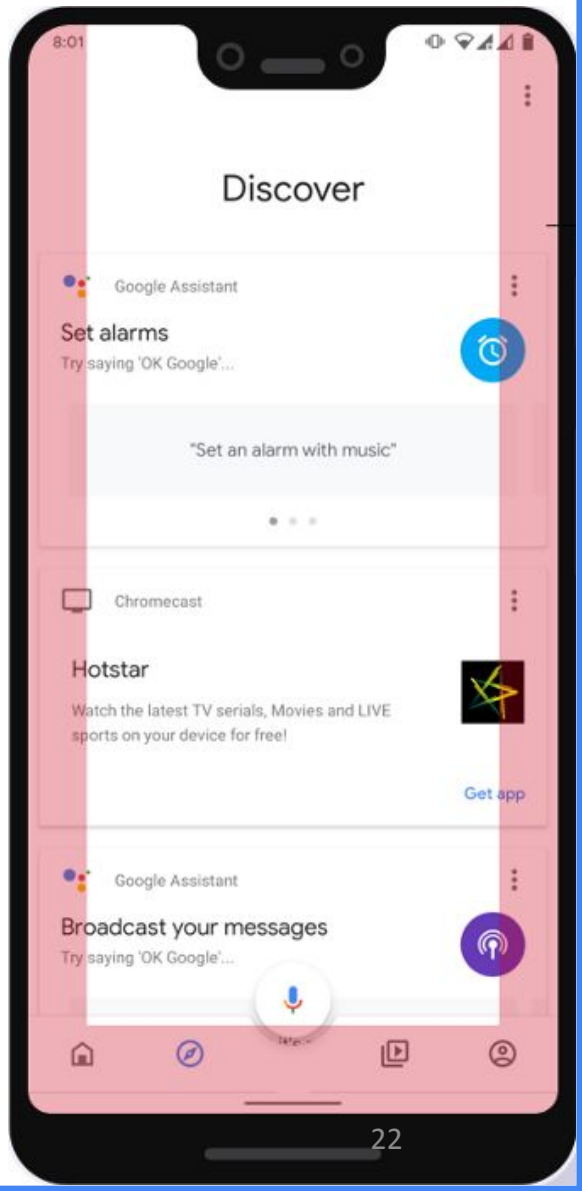
Основные виды Insets

- SystemBars
  - объединяет statusBar, navigationBar и captionBar



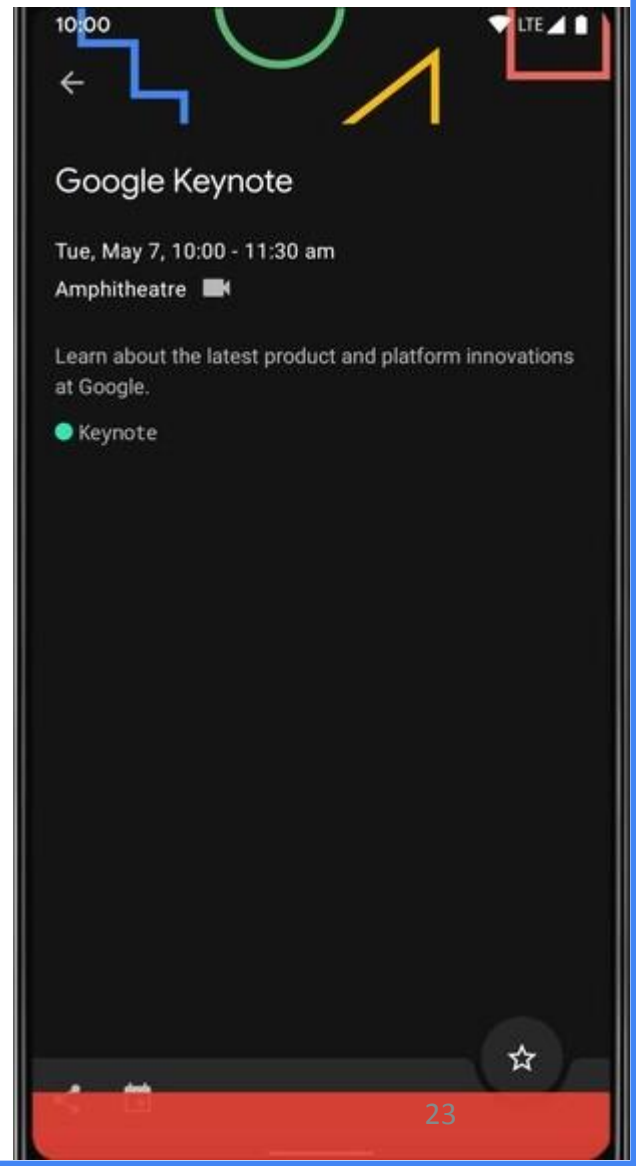
Основные виды Insets

- SystemBars
- SystemGesures
  - указывает какие области зарезервированы за системной навигацией



Основные виды Insets

- SystemBars
- SystemGesures
- MandatorySystemGestures
  - подмножество SystemGesures которые невозможно переопределить

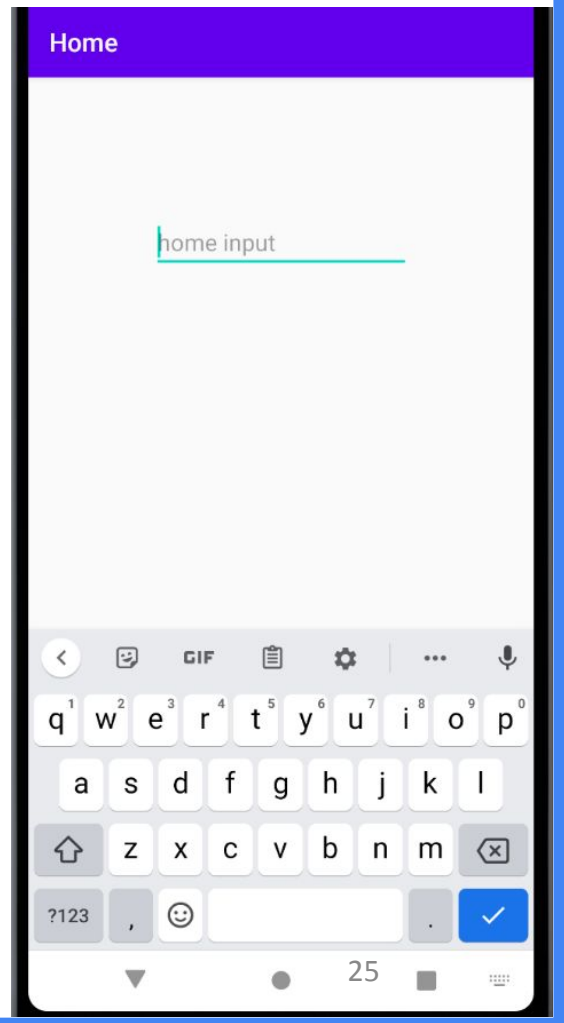


Основные виды Insets

- SystemBars
- SystemGesures
- MandatorySystemGestures
- DisplayCutout
  - вернет информацию о display cutout, если они есть

Основные виды Insets

- SystemBars
- SystemGesures
- MandatorySystemGestures
- DisplayCutout
- Ime
  - вся информация о клавиатуре включая размер и видимость в данный момент



План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

Распространение по иерархии (api < 30)

Распространение по иерархии (api < 30)

- Система шлет данные корневой View

Распространение по иерархии (api < 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
  - Может вернуть измененные данные об Insets
  - Может вернуть пустые Insets - consumed

Распространение по иерархии (api < 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:

Распространение по иерархии (api < 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:
  - Обрабатывает Insets сама как View

```
@Override
public WindowInsets dispatchApplyWindowInsets(WindowInsets insets){
    insets = super.dispatchApplyWindowInsets(insets);
    if (insets.isConsumed()) {
        return insets;
    }
    final int count = getChildCount();
    for (int i = 0; i < count; i++) {
        insets = getChildAt(i).dispatchApplyWindowInsets(insets);
        if (insets.isConsumed()) {
            break;
        }
    }
    return insets;
}
```

Распространение по иерархии (api < 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:
  - Обрабатывает Insets сама как View
  - Если они все обработаны - завершаем работу

```
@Override
public WindowInsets dispatchApplyWindowInsets(WindowInsets insets){
    insets = super.dispatchApplyWindowInsets(insets);
    if (insets.isConsumed()) {
        return insets;
    }
    final int count = getChildCount();
    for (int i = 0; i < count; i++) {
        insets = getChildAt(i).dispatchApplyWindowInsets(insets);
        if (insets.isConsumed()) {
            break;
        }
    }
    return insets;
}
```

Распространение по иерархии (api < 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:
  - Обрабатывает Insets сама - как View
  - Если они все обработаны - завершаем работу
  - Шлет их ChildView в порядке добавления в иерархию

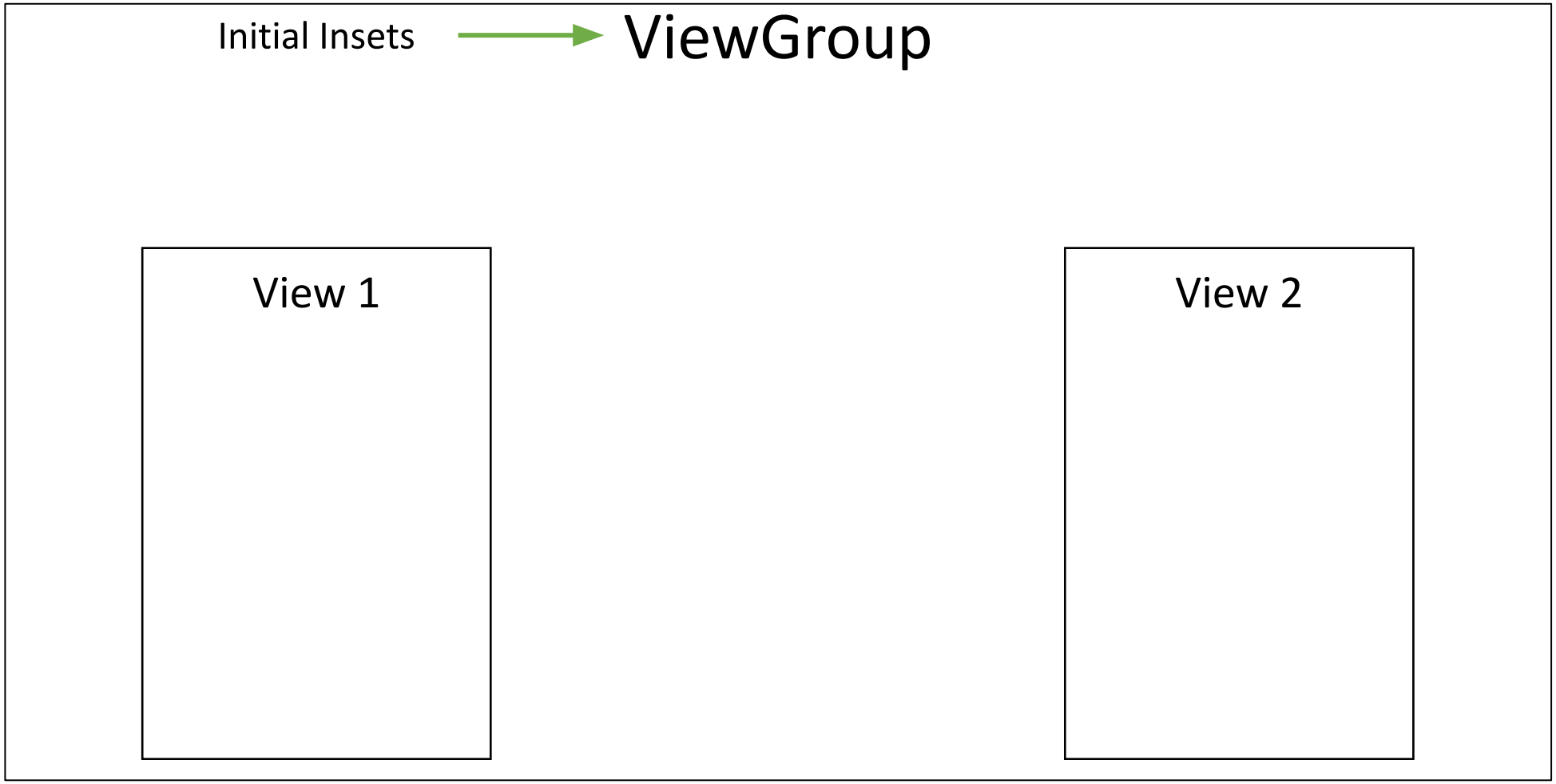
```
@Override
public WindowInsets dispatchApplyWindowInsets(WindowInsets insets){
    insets = super.dispatchApplyWindowInsets(insets);
    if (insets.isConsumed()) {
        return insets;
    }
    final int count = getChildCount();
    for (int i = 0; i < count; i++) {
        insets = getChildAt(i).dispatchApplyWindowInsets(insets);
        if (insets.isConsumed()) {
            break;
        }
    }
    return insets;
}
```

Распространение по иерархии (api < 30)

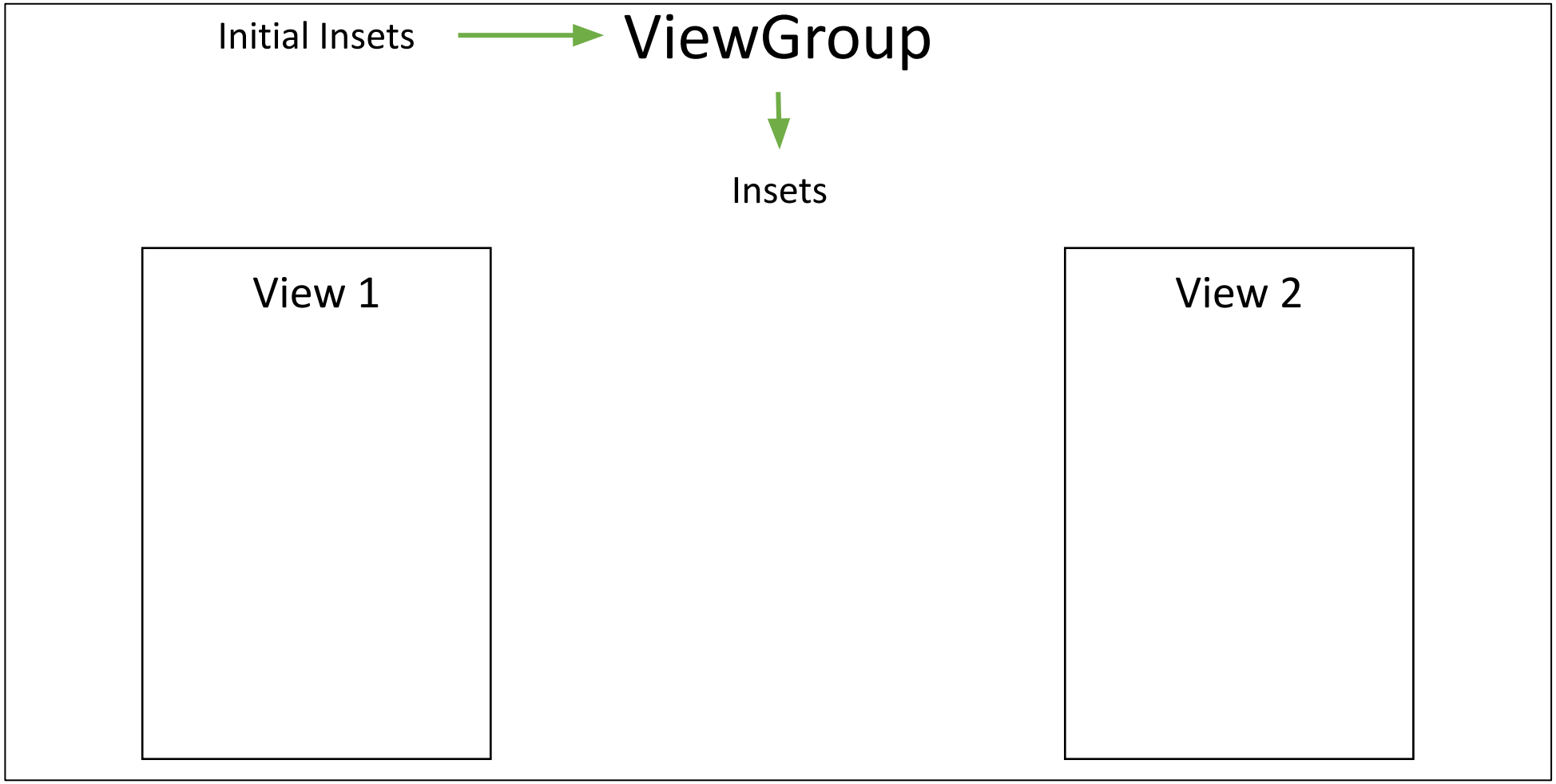
- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:
  - Обрабатывает Insets сама
    - как View
  - Если они все обработаны - завершаем работу
  - Шлет их ChildView в порядке добавления в иерархию
  - Если ChildView обработала Insets, то они не уйдут другим View ниже по иерархии

```
@Override
public WindowInsets dispatchApplyWindowInsets(WindowInsets insets){
    insets = super.dispatchApplyWindowInsets(insets);
    if (insets.isConsumed()) {
        return insets;
    }
    final int count = getChildCount();
    for (int i = 0; i < count; i++) {
        insets = getChildAt(i).dispatchApplyWindowInsets(insets);
        if (insets.isConsumed()) {
            break;
        }
    }
    return insets;
}
```

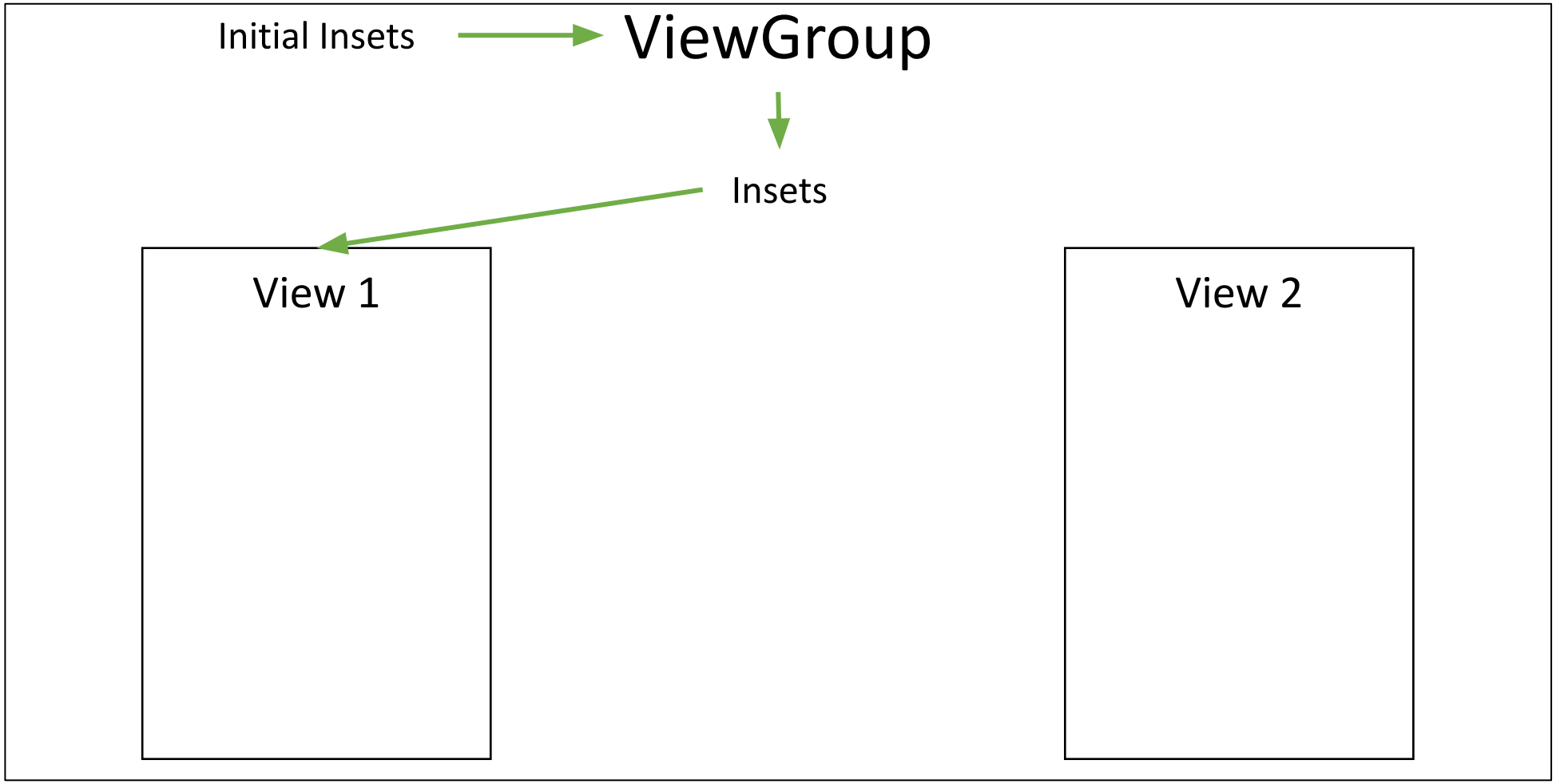
Распространение по иерархии (api < 30)



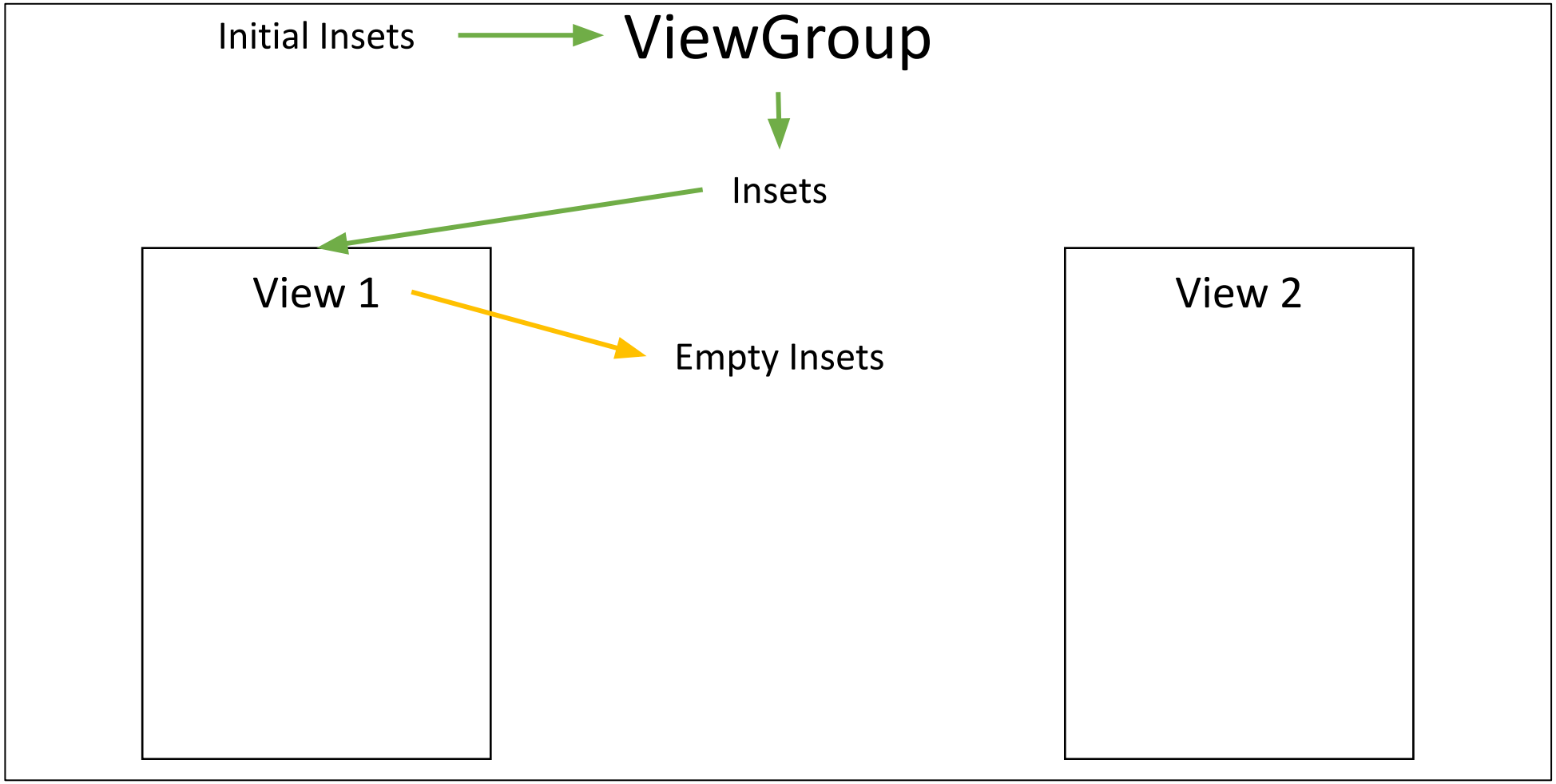
Распространение по иерархии (api < 30)



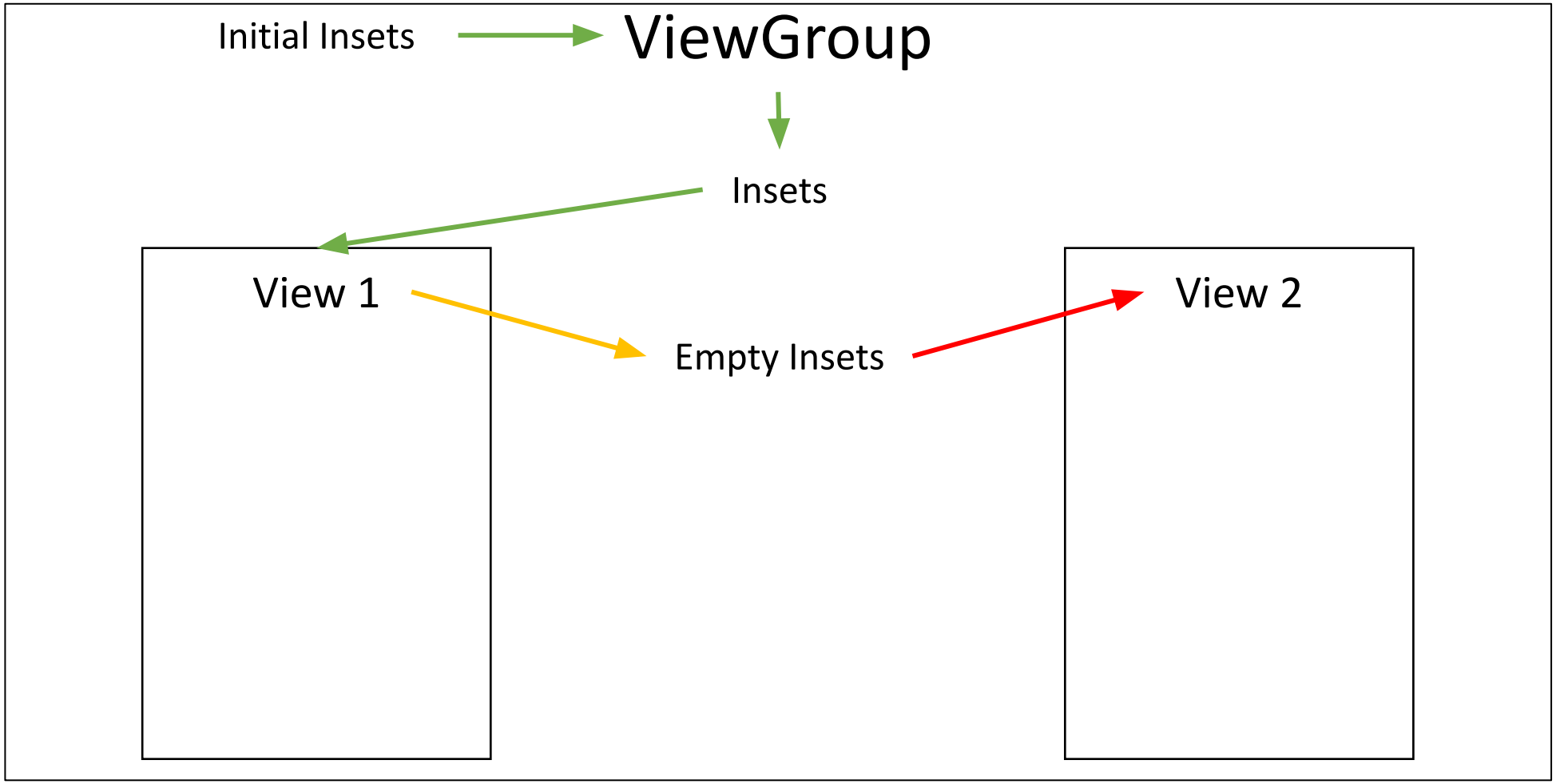
Распространение по иерархии (api < 30)



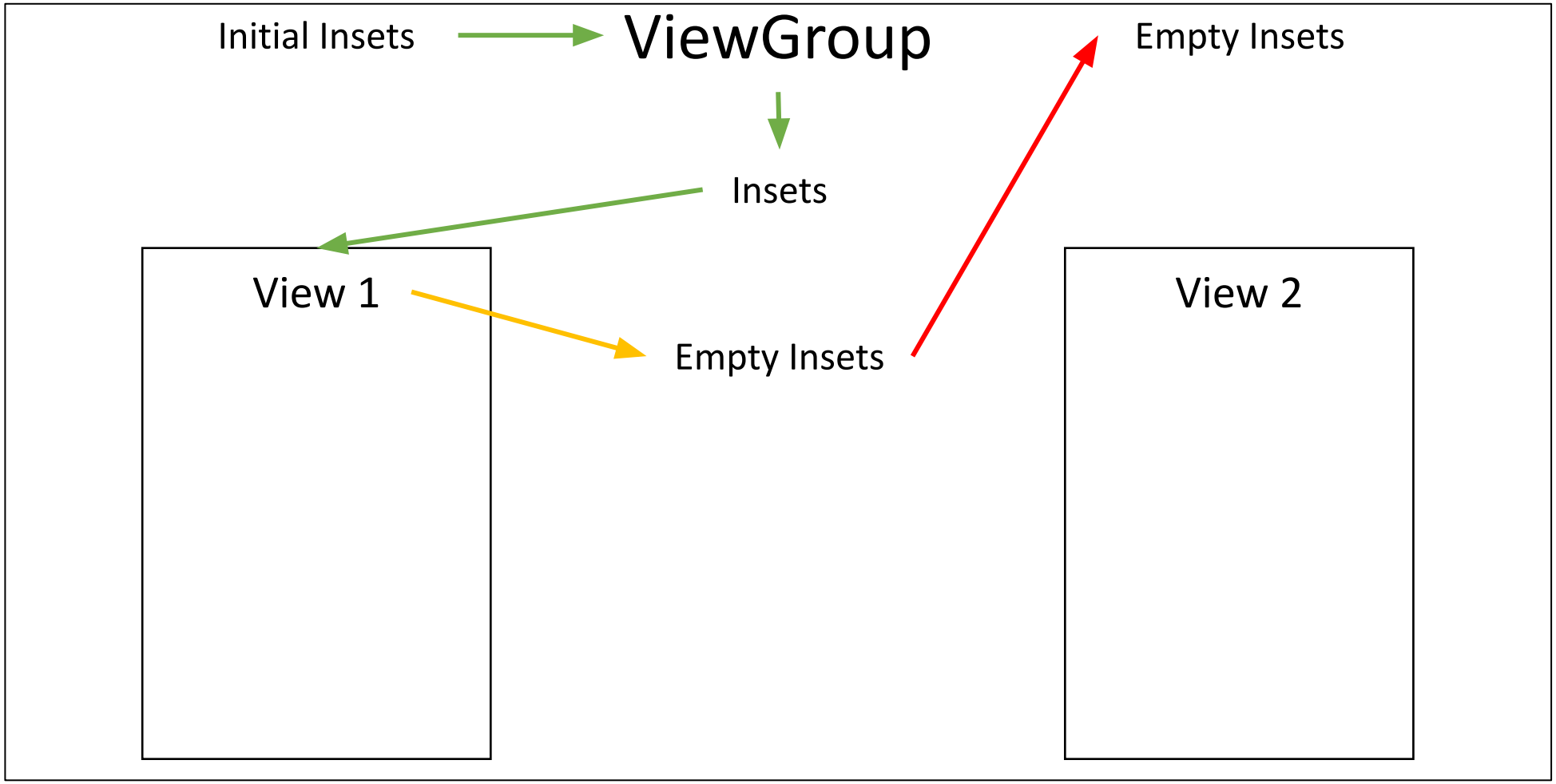
Распространение по иерархии (api < 30)



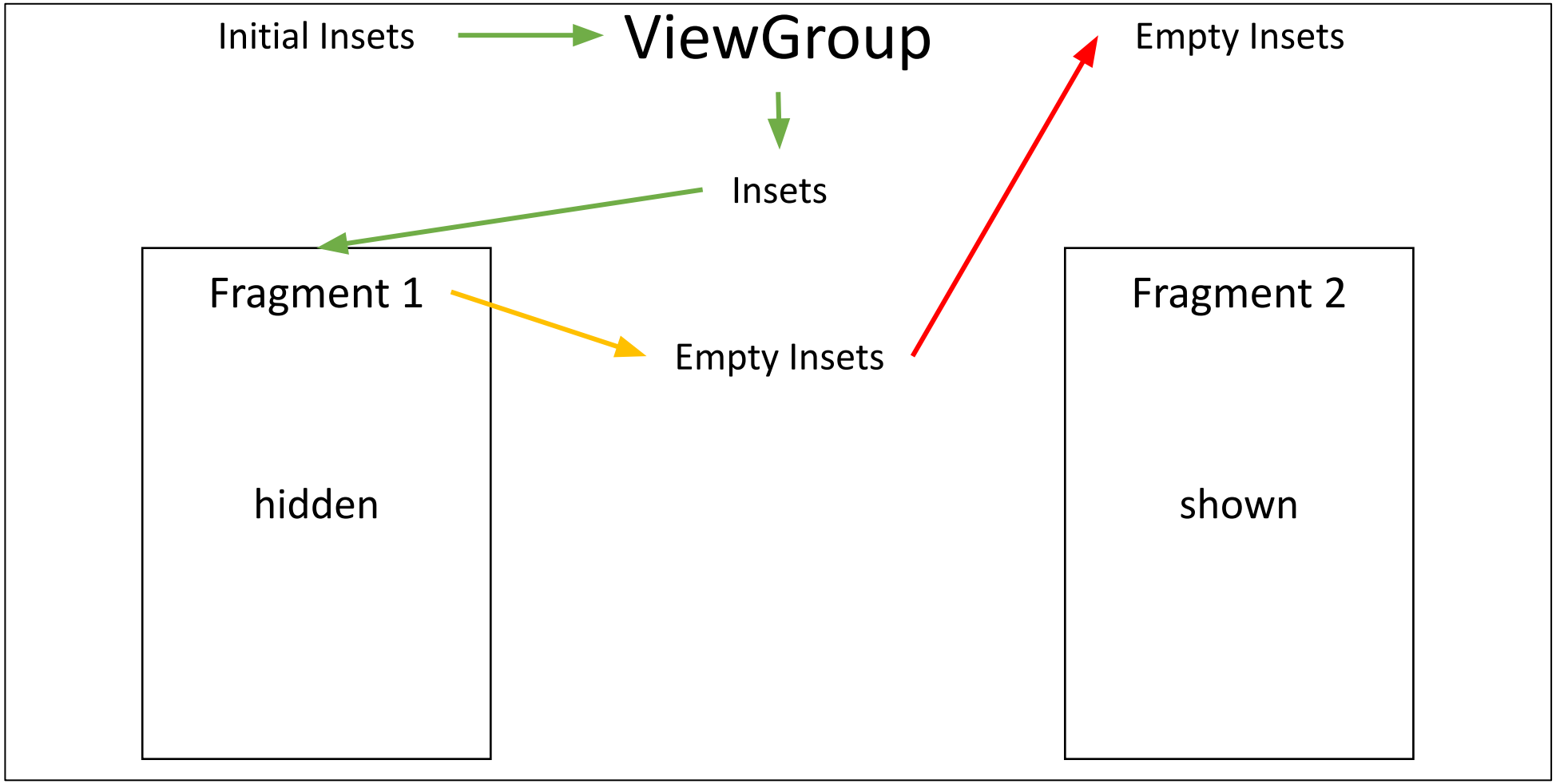
Распространение по иерархии (api < 30)



Распространение по иерархии (api < 30)



Распространение по иерархии (api < 30)



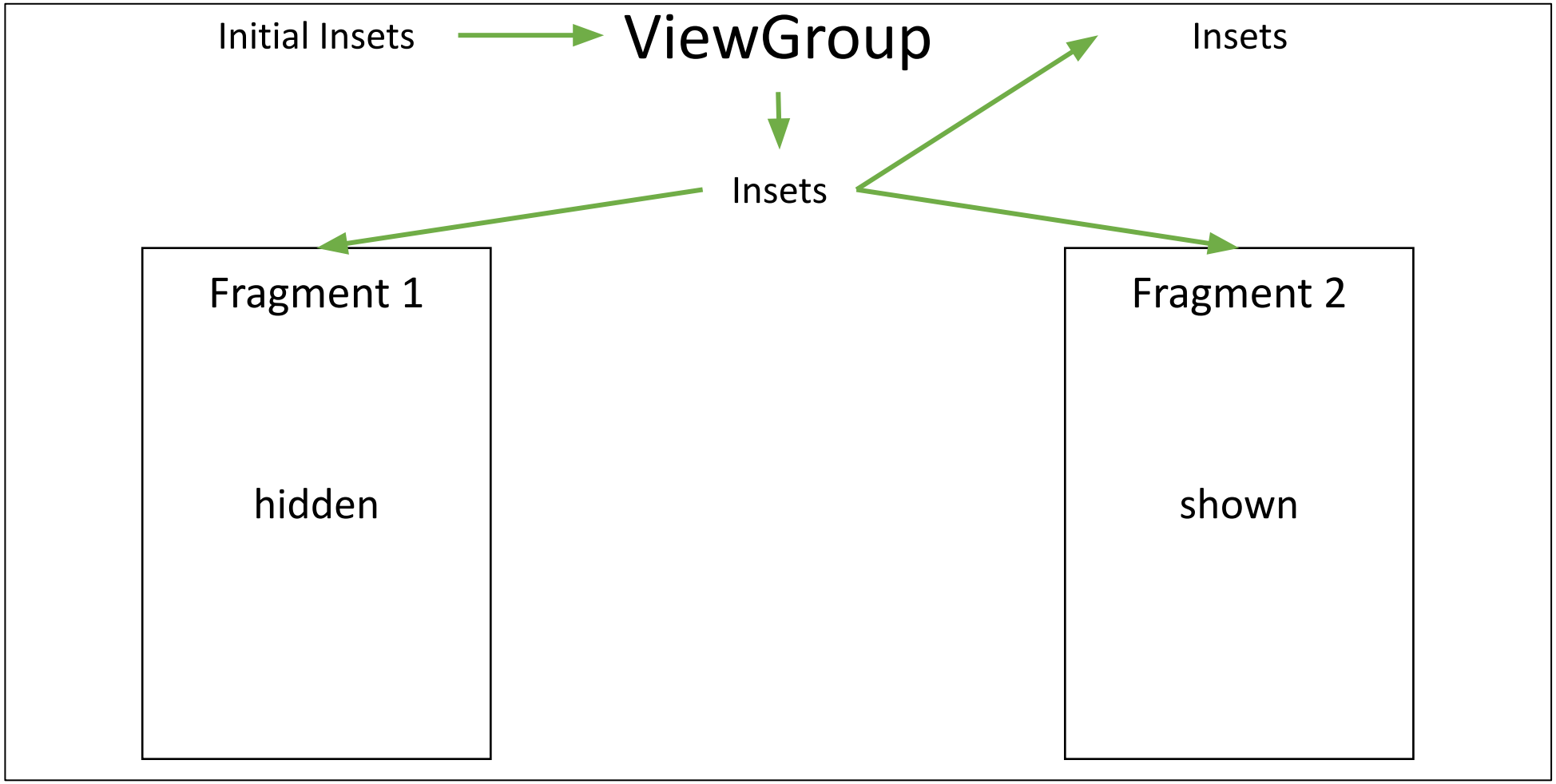
https://habr.com/ru/company/oleg-bunin/blog/488196/ - Константин Цховребов

Распространение по иерархии (api >= 30)

- Система шлет данные корневой View
- View обрабатывает и возвращает Insets
- ViewGroup:
  - Обрабатывает Insets сама
    - как View
  - Если они все обработаны - завершаем работу
  - Шлет их ChildView в порядке добавления в иерархию
  - ~~Если ChildView обработала Insets, то они не уйдут другим View ниже по иерархии~~

```
@Override
public WindowInsets dispatchApplyWindowInsets(WindowInsets insets){
    insets = super.dispatchApplyWindowInsets(insets);
    if (insets.isConsumed()) {
        return insets;
    }
    final int count = getChildCount();
    for (int i = 0; i < count; i++) {
        insets = getChildAt(i).dispatchApplyWindowInsets(insets);
        if (insets.isConsumed()) {
            break;
        }
    }
    return insets;
}
```

Распространение по иерархии (api >= 30)



Распространение по иерархии: итоги

Распространение по иерархии: итоги

- Есть ключевые отличия распространении insets в платформенных ViewGroup до Android 11 (api 30) и после
  - Об этом часто забывают, тк многие доклады сделаны до api 30

Распространение по иерархии: итоги

- Есть ключевые отличия распространении insets в платформенных ViewGroup до Android 11 (api 30) и после
  - Об этом часто забывают, тк многие доклады сделаны до api 30
- Тестируйте приложение на разных api
  - как минимум на одном до Android 11 (api 30) и после

Распространение по иерархии: итоги

- Есть ключевые отличия распространении insets в платформенных ViewGroup до Android 11 (api 30) и после
  - Об этом часто забывают, тк многие доклады сделаны до api 30
- Тестируйте приложение на разных api
  - как минимум на одном до Android 11 (api 30) и после
- Используйте compat или library классы
  - ConstraintLayout, FragmentContainerView и другие

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

Анимируем клавиатуру: теория

```
val callback : WindowInsetsAnimationCompat.Callback = …
/// WindowInsetsAnimationCompat == WIAC
/// WindowInsetsCompat == WIC
```

```
ViewCompat.setWindowInsetsAnimationCallback(view, callback)
```

Анимируем клавиатуру: теория

```
val callback : WIAC.Callback = …
```

```
public Callback(@DispatchMode int dispatchMode) {
    mDispatchMode = dispatchMode;
}
```

```
ViewCompat.setWindowInsetsAnimationCallback(view, callback)
```

Анимируем клавиатуру: теория

```
val callback : WIAC.Callback = …

@IntDef(value = {
        DISPATCH_MODE_STOP,
        DISPATCH_MODE_CONTINUE_ON_SUBTREE
})
public @interface DispatchMode {}

ViewCompat.setWindowInsetsAnimationCallback(view, callback)
```

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {
    fun onPrepare(animation: WindowInsetsAnimationCompat)

    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat

    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC

    fun onEnd(animation: WIAC)

}
```

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией

```
}
```

https://www.youtube.com/watch?v=acC7SR1EXsI - Android Developers от 24 авг. 2020
https://developer.android.com/develop/ui/views/layout/sw-keyboard

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    private lateinit var viewPositionInfo: PositionInfo

    fun onPrepare(animation: WindowInsetsAnimationCompat){
        viewPositionInfo = saveCurrentViewPosition(v)
    }

}
```

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией
- разработчик должен запомнить расположение View ДО начала анимации

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- ДО вызова этого метода вызывается dispatchApplyWindowInsets с обновленной информацией
  - Таким образом вызываются все методы View.OnApplyWindowInsetsListener ДО вызова этого метода
- сразу после происходит layout фаза в приложении
- вызывается этот метод
- только после этого происходит draw фаза приложения

```
}
```

https://www.youtube.com/watch?v=acC7SR1EXsI - Android Developers от 24 авг. 2020
https://developer.android.com/develop/ui/views/layout/sw-keyboard

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

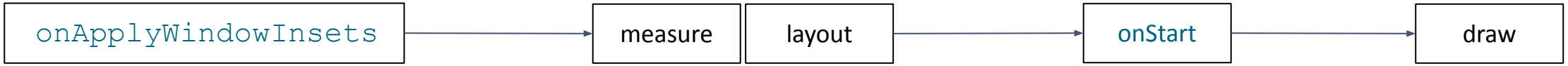
- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- получаем обновленную информацию об Insets с изменениями как после анимации
- вызывается layout фаза приложения

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- вызывается этот метод
- вызывается draw фазы приложения




```
}
```

https://www.youtube.com/watch?v=acC7SR1EXsI - Android Developers от 24 авг. 2020
https://developer.android.com/develop/ui/views/layout/sw-keyboard

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC {
        applyWindowInsets(v, windowInsets)

        return consumeWindowInsets(v, windowInsets)
    }

    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat {
        revertPreviousViewPosition(v, viewPositionInfo)

        return super.onStart(animation, bounds)
    }




}
```

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- вызывается после onPrepare и до onStart
- получаем обновленную информацию об Insets с изменениями как после анимации
- вызывается layout фаза приложения

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- вызывается этот метод
- вызывается draw фазы приложения

```
    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC
```

- вызывается каждый тик анимации, когда Insets были изменены

```
}
```

https://www.youtube.com/watch?v=acC7SR1EXsI - Android Developers от 24 авг. 2020
https://developer.android.com/develop/ui/views/layout/sw-keyboard

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC {
        return applyInsetsDuringAnimation(insets)
    }









}
```

Анимируем клавиатуру: подход Google

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- получаем обновленную информацию об Insets
- вызывается после onPrepare и до onStart

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- вызывается этот метод
- вызывается draw фазы приложения

```
    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC
```

- вызывается каждый тик анимации, когда Insets были изменены

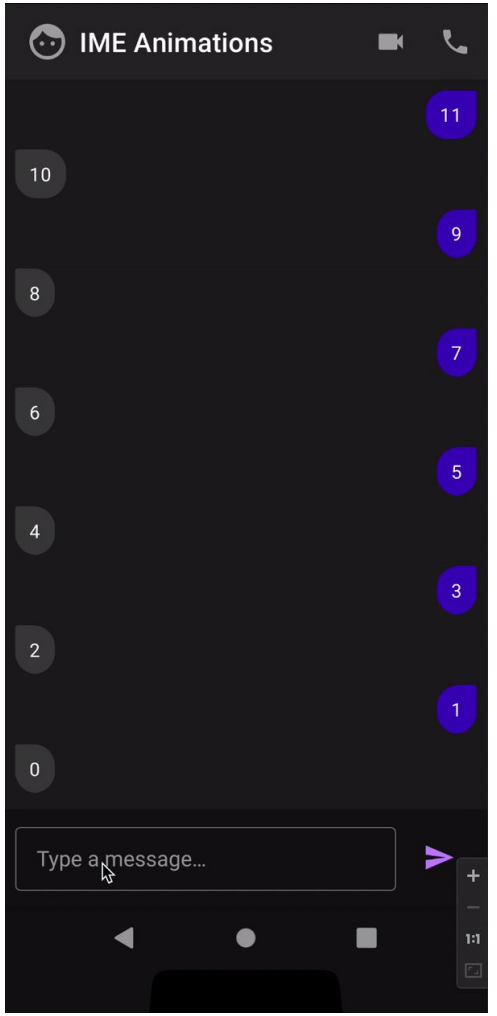
```
    fun onEnd(animation: WIAC)
```

- вызывается после окончания анимации

```
}
```

https://www.youtube.com/watch?v=acC7SR1EXsI - Android Developers от 24 авг. 2020
https://developer.android.com/develop/ui/views/layout/sw-keyboard

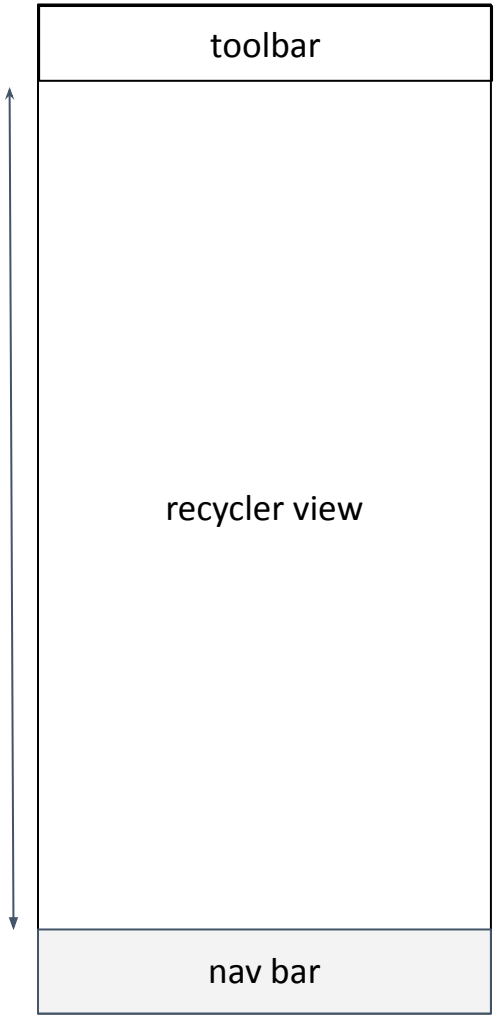
Анимируем клавиатуру: подход Google

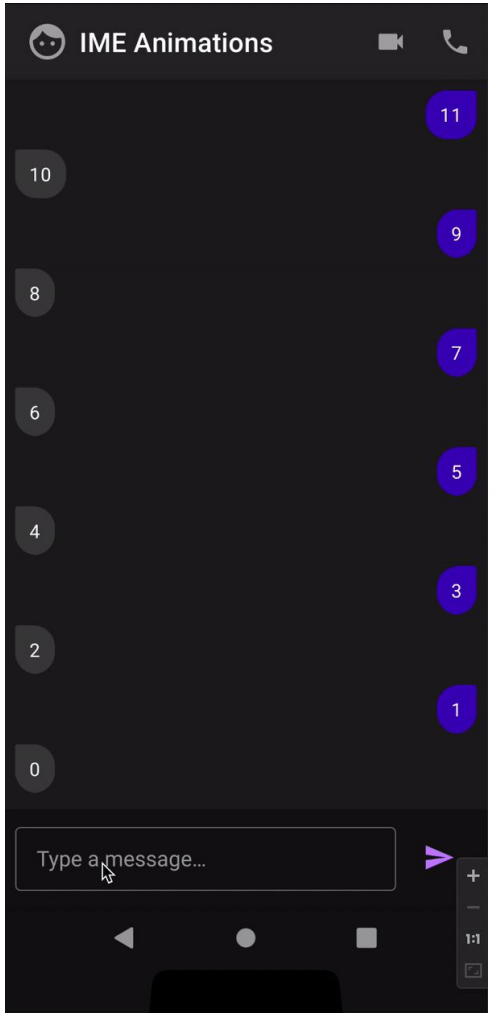


План доклада

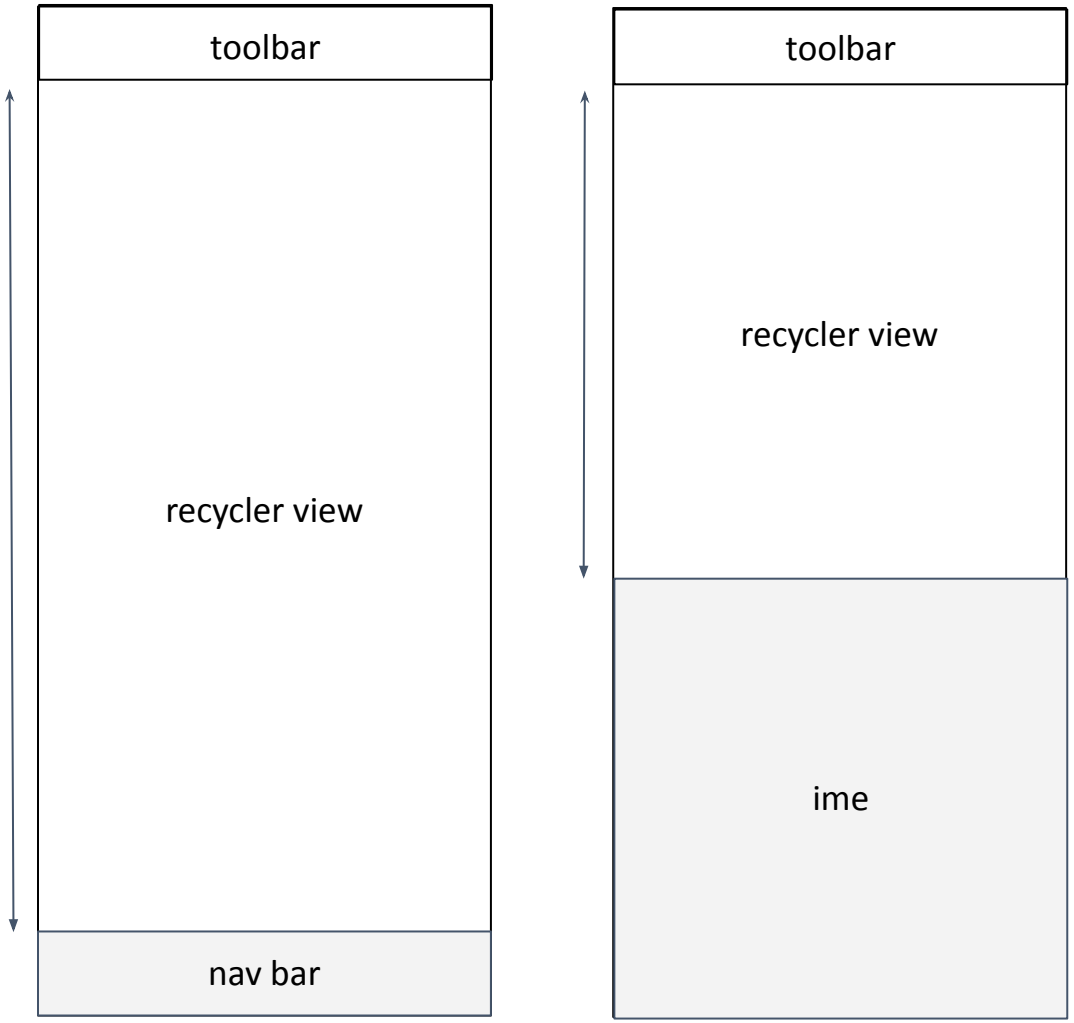
1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

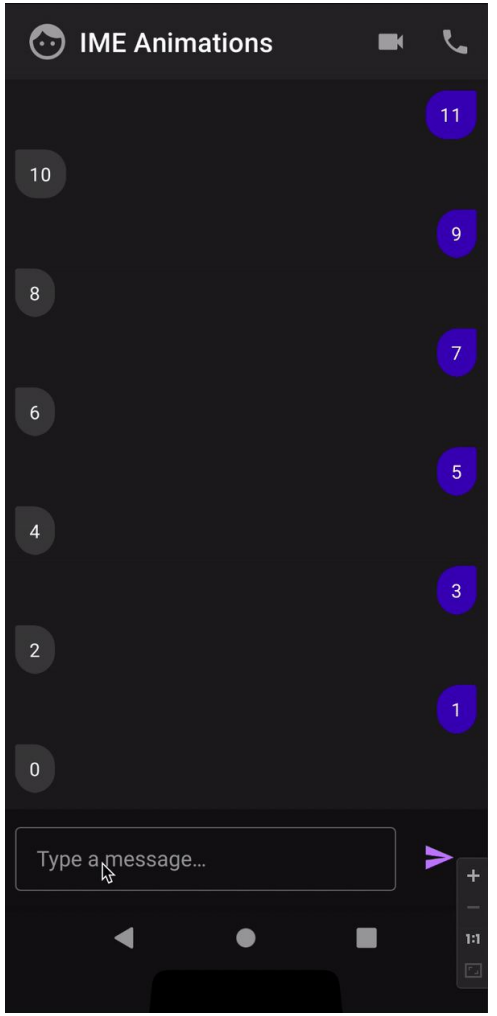
Анимируем клавиатуру: подход Google



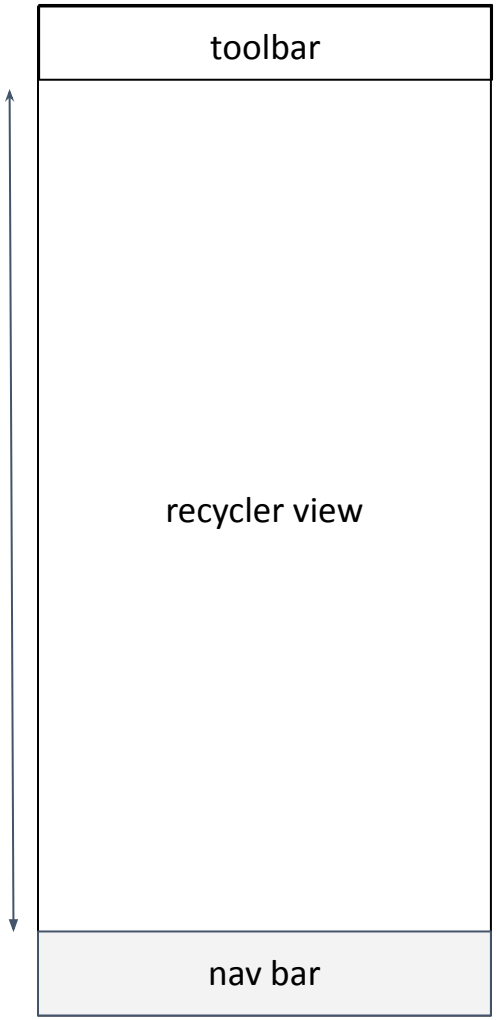


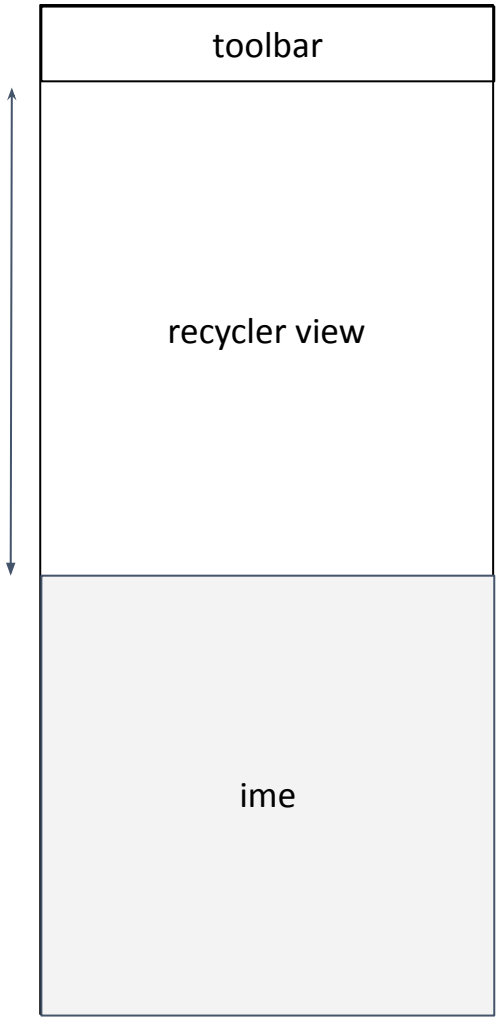
Анимируем клавиатуру: подход Google

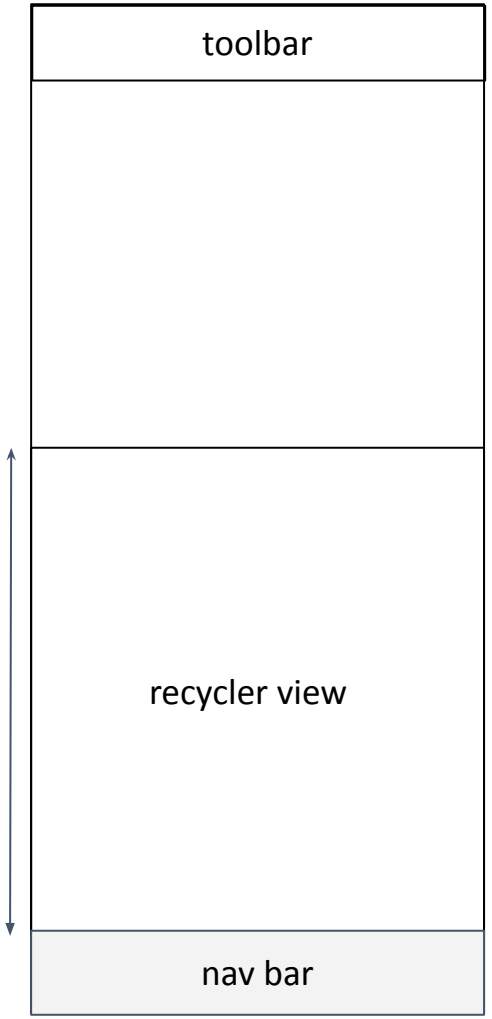


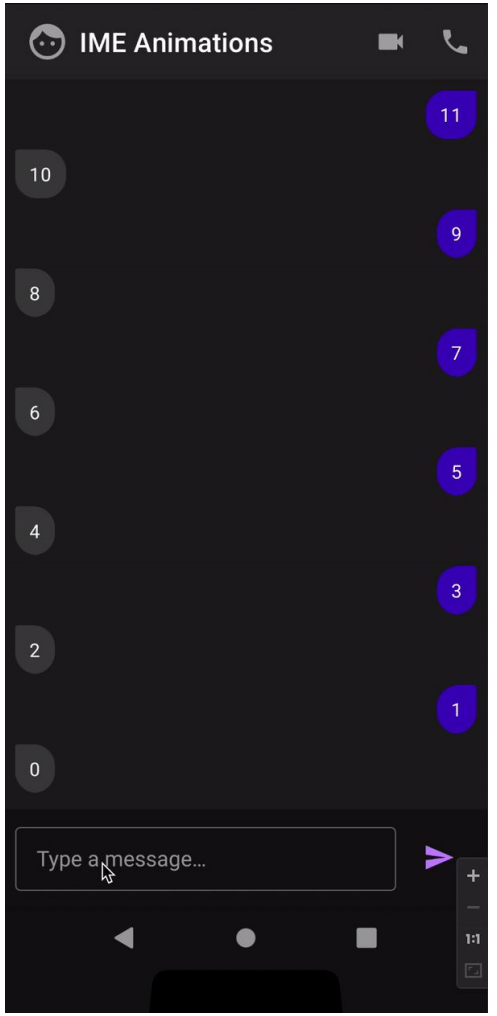


Анимируем клавиатуру: подход Google

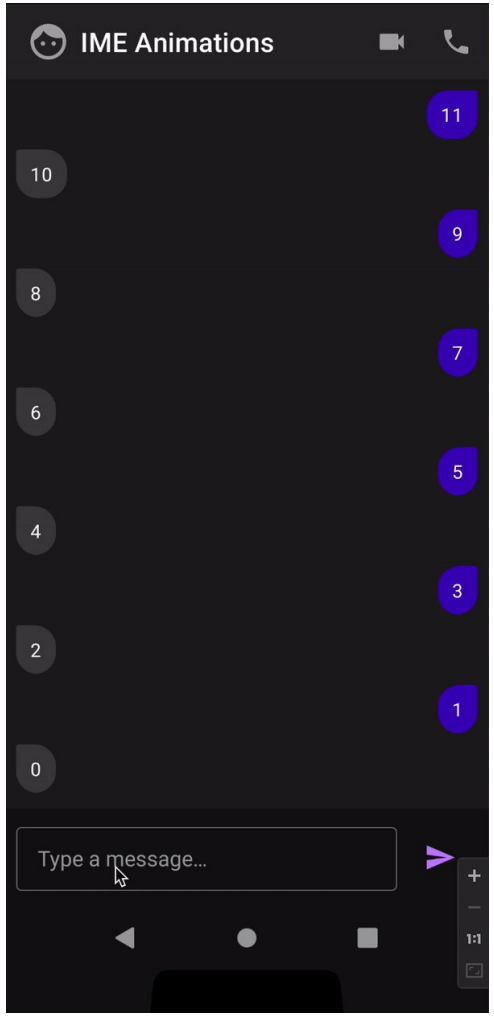


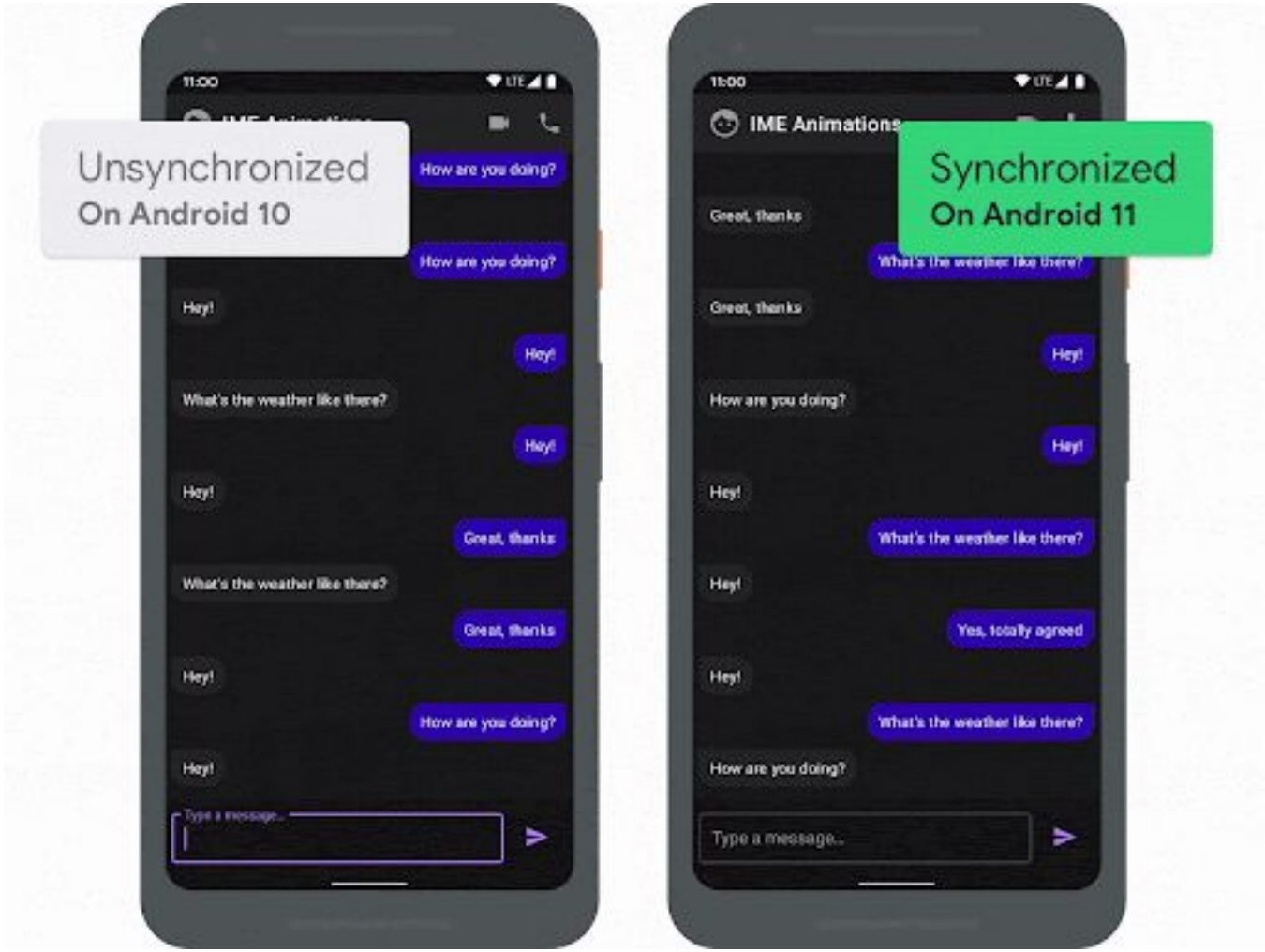






А как же семпл Google? Он же работает!





https://github.com/android/user-interface-samples/tree/main/WindowInsetsAnimation - last updated 23.08.2022

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- вызывается ДО начала анимации
- вызывается ДО любых изменений в Insets, связанных с этой анимацией
- проверяем что это анимация клавиатуры
- нужно запомнить изначальные Insets ДО анимации

```
}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    private var animationTypeMask: Int = -1
    private lateinit var windowInsetsBeforeAnimation: WIC

    fun onPrepare(animation: WindowInsetsAnimationCompat){
        if (animationTypeMask == -1 && (animation.typeMask and WIC.Type.ime()) != 0) {
            animationTypeMask = animation.typeMask
            windowInsetsBeforeAnimation = latestWindowInsets
        }
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- проверяем что это анимация клавиатуры
- нужно запомнить изначальные Insets ДО анимации

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- вызывается после onPrepare и до onStart
- получаем обновленную информацию об Insets с изменениями как после анимации
- вызывается layout фаза приложения
- применяем обновленные Inset, только если анимация НЕ запущена
- если анимация запущена, то мы применяем только системные Insets (имитируем отсутствие клавиатуры)
- запоминаем Insets

```
}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    private lateinit var latestWindowInsets: WIC

    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC {
        latestWindowInsets = windowInsets
        if (animationTypeMask == -1) applyWindowInsets(v, windowInsets)
        else applyWindowInsets(v, windowInsets.removeImeInsets())

        return consumeWindowInsets(v, windowInsets)
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- проверяем что это анимация клавиатуры
- нужно запомнить изначальные Insets ДО анимации

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- применяем обновленные Inset, только если анимация НЕ запущена
- если анимация запущена, то мы применяем только системные Insets (имитируем отсутствие клавиатуры)
- запоминаем Insets

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- вызывается этот метод
- вызывается draw фазы приложения
- восстанавливаем состояние ДО

```
}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat {
        if (animationTypeMask == animation.typeMask) {
            applyInsetsDuringAnimation(windowInsetsBeforeAnimation)
        }
        return super.onStart(animation, bounds)
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onPrepare(animation: WindowInsetsAnimationCompat)
```

- проверяем что это анимация клавиатуры
- нужно запомнить изначальные Insets ДО анимации

```
    fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC
```

- применяем обновленные Inset, только если анимация НЕ запущена
- если анимация запущена, то мы применяем только системные Insets (имитируем отсутствие клавиатуры)
- запоминаем Insets

```
    fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat
```

- восстанавливаем состояние ДО

```
    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC
```

- анимируем переход из одного состояния в другое исходя из пришедших данных в `insets`
  - важно не использовать методы вызывающие `requestLayout`

```
}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC {
        if (runningAnimations.any { it.typeMask == animationTypeMask }) {
            return applyInsetsDuringAnimation(insets)
        }
        return insets
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {
```

`fun onPrepare(animation: WindowInsetsAnimationCompat)`

- проверяем что это анимация клавиатуры
- нужно запомнить изначальные Insets ДО анимации

`fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC`

- применяем обновленные Inset, только если анимация НЕ запущена
- если анимация запущена, то мы применяем только системные Insets (имитируем отсутствие клавиатуры)
- запоминаем Insets

`fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat`

- восстанавливаем состояние ДО

`fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC`

- анимируем переход из одного состояния в другое исходя из пришедших данных в `insets`

`fun onEnd(animation: WIAC)`

- сбрасываем внутреннее состояние
- применяем Insets, которые мы запомнили ранее

```
}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onEnd(animation: WIAC) {
        if (animationTypeMask == animation.typeMask) {
            animationTypeMask = -1

            onImeEnd()
            onApplyWindowInsets(view, latestWindowInsets)
        }
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    fun onEnd(animation: WIAC) {
        if (animationTypeMask == animation.typeMask) {
            animationTypeMask = -1

            onImeEnd()
            onApplyWindowInsets(view, latestWindowInsets)
        }
    }









}
```

Анимируем клавиатуру: новый подход

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {
     private var animationTypeMask: Int = -1
     private lateinit var windowInsetsBeforeAnimation: WIC
     private lateinit var latestWindowInsets: WIC
     fun onPrepare(animation: WindowInsetsAnimationCompat){
          if (animationTypeMask == -1 && animation.typeMask and WIC.Type.ime() != 0) {
               animationTypeMask = animation.typeMask
               windowInsetsBeforeAnimation = latestWindowInsets
          }
     }
     fun onApplyWindowInsets(v: View, windowInsets: WIC): WIC {
          latestWindowInsets = windowInsets
          if (animationTypeMask == -1) applyWindowInsets(v, windowInsets)
          else applyWindowInsets(v, windowInsets.removeImeInsets())
          return consumeWindowInsets(v, windowInsets)
     }
     fun onStart(animation: WIAC, bounds: WIAC.BoundsCompat): WIAC.BoundsCompat {
          if (animationTypeMask == animation.typeMask) applyInsetsInAnimation(windowInsetsBeforeAnimation)
          return super.onStart(animation, bounds)
     }
     fun onProgress(insets: WindowInsetsCompat, runningAnimations: List<WIAC>): WIC =
          if (runningAnimations.any { it.typeMask == animationTypeMask }) applyInsetsInAnimation(insets)
          else insets
     fun onEnd(animation: WIAC) {
          if (animationTypeMask == animation.typeMask) {
               animationTypeMask = -1
               onImeEnd()
               onApplyWindowInsets(view, latestWindowInsets)
          }
     }
}
```

https://github.com/mig35/WindowInsetsAnimation

Анимируем клавиатуру: новый подход

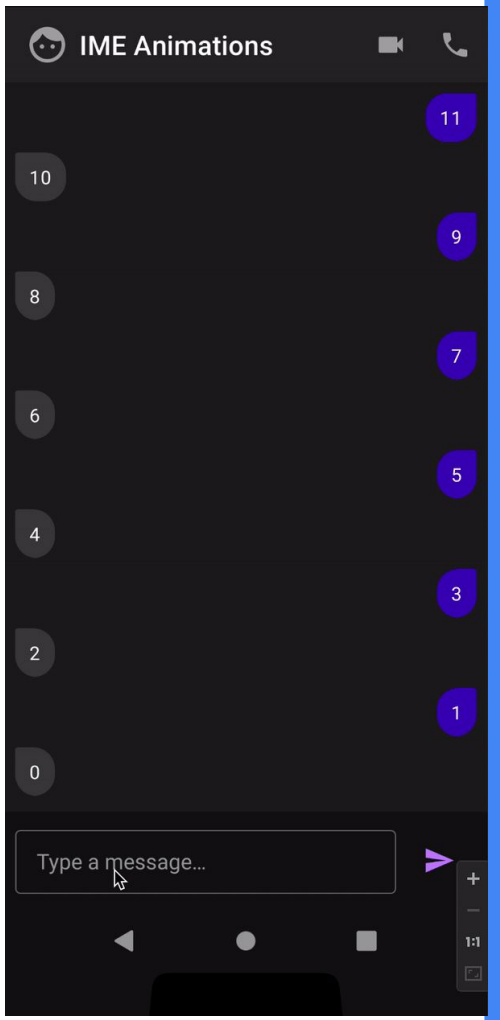
```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE) {

    private fun applyWindowInsets(v: View, windowInsets: WIC): WIC {

    }

    private fun applyInsetsInAnimation(windowInsets: WIC): WIC {

    }

    private fun onImeEnd() {

    }



}
```

План доклада

1. Узнаем какую проблему решают Insets
2. Узнаем какие вообще виды Insets бывают
3. Узнаем как Insets доставляются до каждой View
4. Узнаем как анимировать клавиатуру по докам Google
5. Узнаем почему документация врет и почему их подход не работает
6. Узнаем как верно реализовать анимацию клавиатуры
7. Рассмотрим пример реализации анимации клавиатуры

Применим наши знания

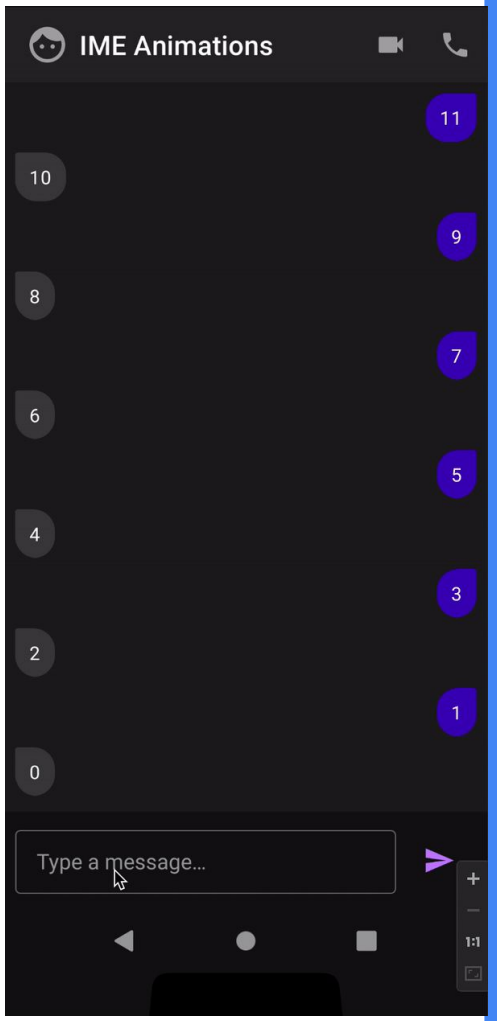
```
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 >

 <androidx.appcompat.widget.Toolbar
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   />

 <androidx.recyclerview.widget.RecyclerView
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   />

 <LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   >

   <EditText
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:layout_weight="1"
     />
   <com.google.android.material.button.MaterialButton
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     />

 </LinearLayout>

</LinearLayout>
```



Применим наши знания

View, взаимодействующие с SystemBarsInsets:

View, взаимодействующие с Ime:

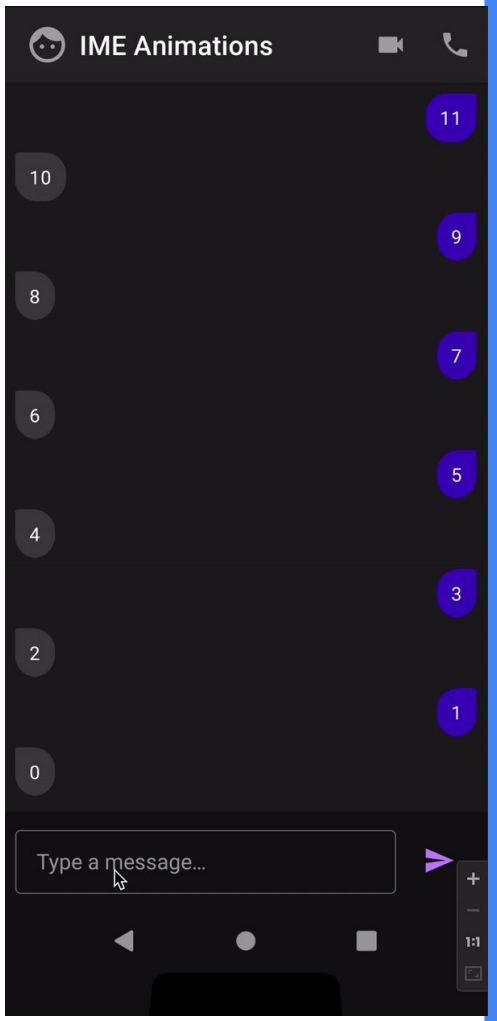


Применим наши знания

View, взаимодействующие с SystemBarsInsets:

- Toolbar
- EditTextContainer

View, взаимодействующие с Ime:



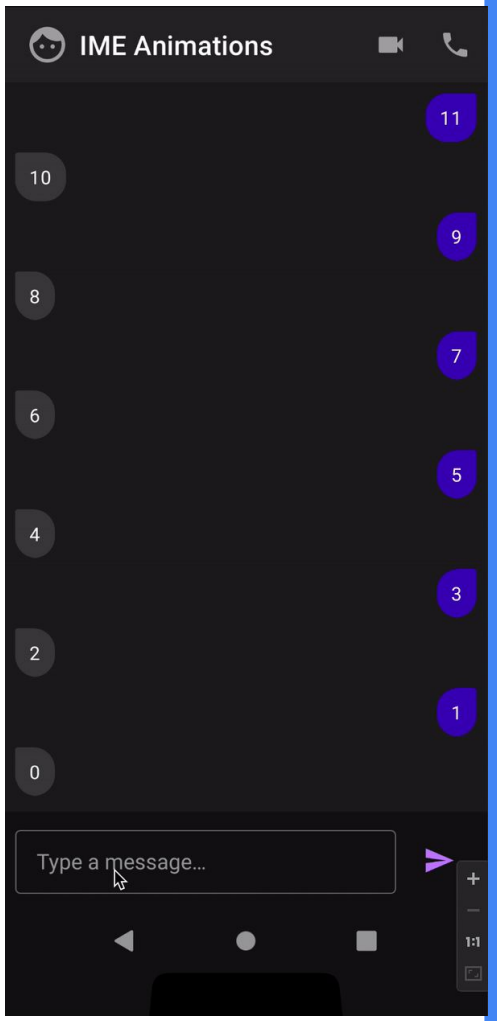
Применим наши знания

View, взаимодействующие с SystemBarsInsets:

- Toolbar
- EditTextContainer

View, взаимодействующие с Ime:

- EditTextContainer



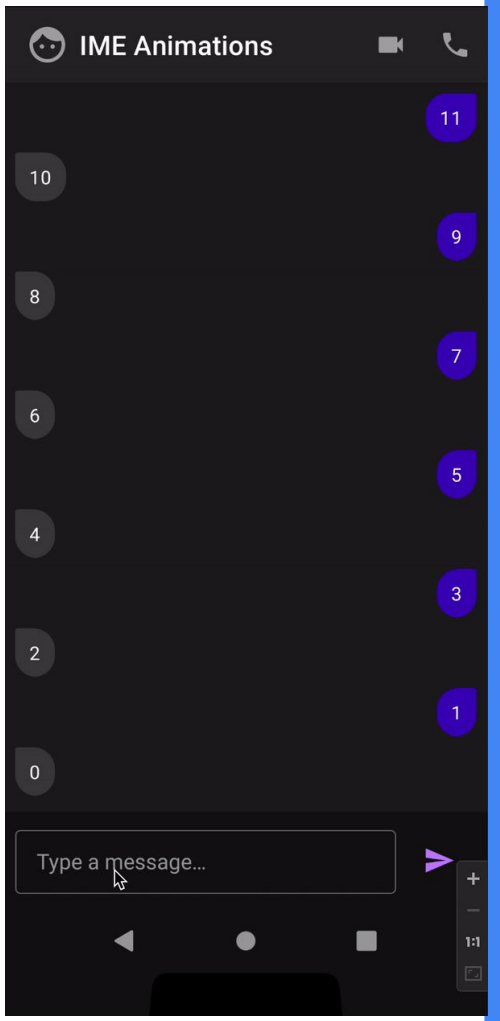
Применим наши знания

View, взаимодействующие с SystemBarsInsets:

- Toolbar
- EditTextContainer

View, взаимодействующие с Ime:

- EditTextContainer
- RecyclerView



Применим наши знания: Toolbar

```
toolbar.doOnApplyWindowInsets { view, insets, initialPadding ->
    val systemBarsInsets = insets.getInsets(WIC.Type.systemBars())

    view.updatePadding(
        top = initialPadding.top + systemBarsInsets.top
    )

    insets
}
```

Применим наши знания: EditText

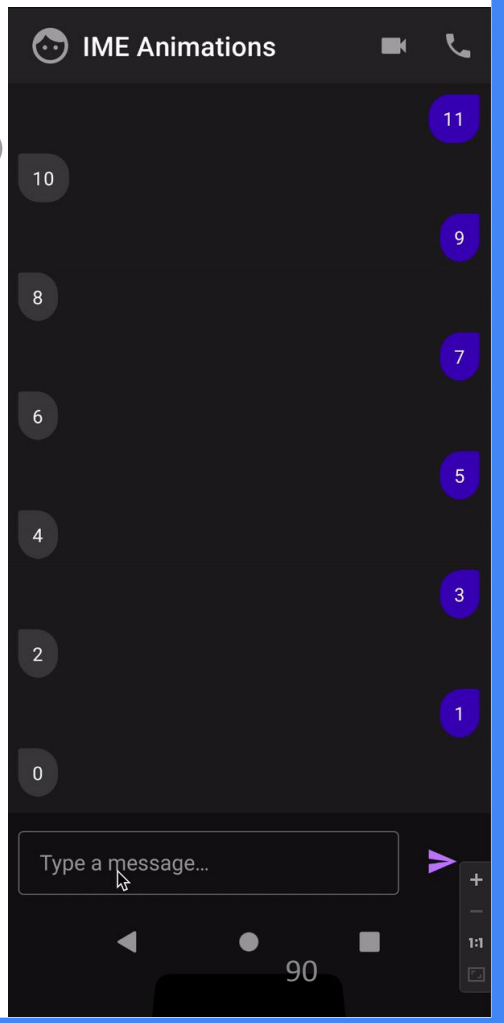
```
val callback = object :
                    WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE),
                    OnApplyWindowInsetsListener {

}



ViewCompat.setWindowInsetsAnimationCallback(editText, callback)
ViewCompat.setOnApplyWindowInsetsListener(editText, callback)
```

Применим наши знания: EditText

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE),
                        OnApplyWindowInsetsListener {

    private fun applyWindowInsets(v: View, windowInsets: WIC): WIC {
        val currentInsets = windowInsets.getInsets(
                if (windowInsets.isVisible(WIC.Type.ime())) WIC.Type.ime()
                else WIC.Type.systemBars()
            )

        view.updatePadding(bottom = initialPadding.bottom + currentInsets.bottom)
    }

    private fun applyInsetsInAnimation(windowInsets: WIC): WIC {
        val imeInset = windowInsets.getInsets(WIC.Type.ime())
        val systemBarsInset =
                    windowInsets.getInsets(WIC.Type.systemBars())

        val diff = maxOf(imeInset.bottom - systemBarsInset.bottom, 0)
        view.translationY = -diff.toFloat()
    }

    private fun onImeEnd() {
        view.translationY = 0f
    }
```

Применим наши знания: EditText

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE),
                        OnApplyWindowInsetsListener {

    private fun applyWindowInsets(v: View, windowInsets: WIC): WIC {
        val currentInsets = windowInsets.getInsets(
                if (windowInsets.isVisible(WIC.Type.ime())) WIC.Type.ime()
                else WIC.Type.systemBars()
            )

        view.updatePadding(bottom = initialPadding.bottom + currentInsets
    }

    private fun applyInsetsInAnimation(windowInsets: WIC): WIC {
        val imeInset = windowInsets.getInsets(WIC.Type.ime())
        val systemBarsInset =
                    windowInsets.getInsets(WIC.Type.systemBars())

        val diff = maxOf(imeInset.bottom - systemBarsInset.bottom, 0)
        view.translationY = -diff.toFloat()
    }

    private fun onImeEnd() {
        view.translationY = 0f
    }
```



Применим наши знания: EditText

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE),
                        OnApplyWindowInsetsListener {

    private fun applyWindowInsets(v: View, windowInsets: WIC): WIC {
        val currentInsets = windowInsets.getInsets(
                if (windowInsets.isVisible(WIC.Type.ime())) WIC.Type.ime()
                else WIC.Type.systemBars()
            )

        view.updatePadding(bottom = initialPadding.bottom + currentInsets.bottom)
    }

    private fun applyInsetsInAnimation(windowInsets: WIC): WIC {
        val imeInset = windowInsets.getInsets(WIC.Type.ime())
        val systemBarsInset =
                    windowInsets.getInsets(WIC.Type.systemBars())

        val diff = maxOf(imeInset.bottom - systemBarsInset.bottom, 0)
        view.translationY = -diff.toFloat()
    }

    private fun onImeEnd() {
        view.translationY = 0f
    }
```

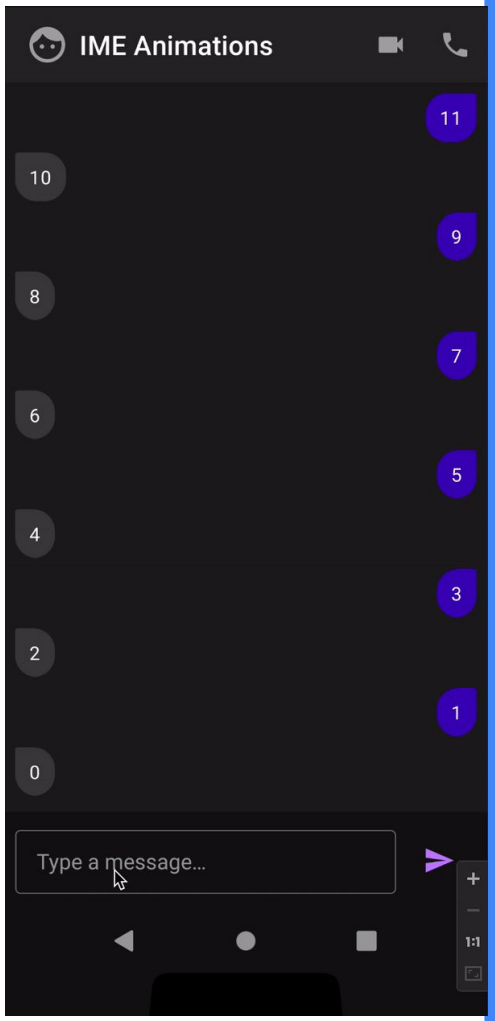
Применим наши знания: RecyclerView

```
val callback = object : WIAC.Callback(DISPATCH_MODE_CONTINUE_ON_SUBTREE),
                        OnApplyWindowInsetsListener {

    private fun applyInsetsInAnimation(windowInsets: WIC): WIC {
        val imeInset = windowInsets.getInsets(WIC.Type.ime())
        val systemBarsInset = windowInsets.getInsets(WIC.Type.systemBars())

        val diff = maxOf(imeInset.bottom - systemBarsInset.bottom, 0)
        view.translationY = -diff.toFloat()
    }

    private fun onImeEnd() {
        view.translationY = 0f
    }




}
```

Применим наши знания: результат

Каждый экран индивидуален.

И, к сожалению, невозможно сделать один универсальный метод обработки анимации клавиатуры.



https://github.com/mig35/WindowInsetsAnimation

Вместо вывода

Вместо вывода

- Insets это важно и обязательно для реализации

Вместо вывода

- Insets это важно и обязательно для реализации
- Помните про различие dispatch метода до api 30 и после

Вместо вывода

- Insets это важно и обязательно для реализации
- Помните про различие dispatch метода до api 30 и после
- Анимация клавиатуры это не просто, но значительно повышает UX приложения

Вместо вывода

- Insets это важно и обязательно для реализации
- Помните про различие dispatch метода до api 30 и после
- Анимация клавиатуры это не просто, но значительно повышает UX приложения
- Помните что могут происходить разные анимации в одно и тоже одно время

Вместо вывода

- Insets это важно и обязательно для реализации
- Помните про различие dispatch метода до api 30 и после
- Анимация клавиатуры это не просто, но значительно повышает UX приложения
- Помните что могут происходить разные анимации в одно и тоже одно время
  - Внимательно следите за параметрами расположения view: xml, кастомные анимации, анимации клавиатуры

Вместо вывода

- Insets это важно и обязательно для реализации
- Помните про различие dispatch метода до api 30 и после
- Анимация клавиатуры это не просто, но значительно повышает UX приложения
- Помните что могут происходить разные анимации в одно и тоже одно время
  - Внимательно следите за параметрами расположения view: xml, кастомные анимации, анимации клавиатуры
- Старайтесь использовать только translate метод в onProgress

Спасибо!

https://github.com/mig35
https://t.me/mmig35
mig35@mig35.com

Полезные материалы по теме

- https://developer.android.com/reference/android/view/WindowInsets
- https://developer.android.com/training/gestures/edge-to-edge
- https://blog.stylingandroid.com/android11-windowinsets-part1/
- https://medium.com/androiddevelopers/animating-your-keyboard-fb776a8fb66d
- https://medium.com/androiddevelopers/gesture-navigation-handling-visual-overlaps-4aed565c134c
- https://habr.com/ru/company/oleg-bunin/blog/488196/
- https://habr.com/ru/company/surfstudio/blog/464373/