



Фёдор Благодирь
Flutter Tech Lead



Да кто такой ЭТОТ ваш 2D-скролл?



Фёдор Благодёр

- 4+ года коммерческой мобильной разработки
- Tech-lead видеоредактора Yappy
- Сведущий сообщества Oh, my Flutter 🚀
- Заблокирован в репозитории Flutter 😊

План

Введение

- Да кто такой этот ваш 2D-скролл
- Типичный кейс 2D-скролла
- Как реализовывался 2D-скролл «в среднем»

План

Введение

- Да кто такой этот ваш 2D-скролл
- Типичный кейс 2D-скролла
- Как реализовывался 2D-скролл «в среднем»

Современный подход

- `TwoDimensionalScrollView`
- Решения «из коробки»,
- Почему они могут не подойти
- Кейс из видеоредактора Yappy

План

Введение

- Да кто такой этот ваш 2D-скролл
- Типичный кейс 2D-скролла
- Как реализовывался 2D-скролл «в среднем»

Современный подход

- `TwoDimensionalScrollView`
- Решения «из коробки»
- Почему они могут не подойти
- Кейс из видеоредактора Yappy

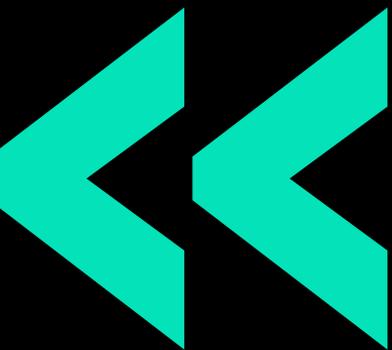
Собственная реализация

- Что требуется
- Влияние на производительность

Введение

Да кто такой этот ваш 2D-скролл

Определение



2D-скролл — прокрутка области экрана в горизонтальном и вертикальном направлениях

Row 1: Column 1	Row 1: Column 2	Row 1: Column 3	Row 1: Column 4	Row 1: Column 5
Row 2: Column 1	Row 2: Column 2	Row 2: Column 3	Row 2: Column 4	Row 2: Column 5
Row 3: Column 1	Row 3: Column 2	Row 3: Column 3	Row 3: Column 4	Row 3: Column 5

Типичный кейс

DataTable виджет

<i>Name</i>	<i>Age</i>	<i>Role</i>
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager

Как реализовывался 2D-скролл в «среднем»

Построение DataTable с 2D-скроллом

```
@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    scrollDirection: Axis.horizontal,
    child: SingleChildScrollView(
      scrollDirection: Axis.vertical,
      child: DataTable(
        columns: /*columns*/,
        rows: /*rows*/,
      ),
    ),
  );
}
```

Как реализовывался 2D-скролл в «среднем»

Старая документация DataTable

Displaying data in a table is **expensive**, because to lay out the table all the data must be **measured twice**, once to negotiate the dimensions to use for each column, and once to actually lay out the table given the results of the negotiation.

For this reason, if you have a lot of data (say, more than a dozen rows with a dozen columns, though the precise limits depend on the target device), it is suggested that you use a **PaginatedDataTable** which automatically splits the data into multiple pages.

Performance considerations when wrapping DataTable with SingleChildScrollView

Wrapping a DataTable with SingleChildScrollView is **expensive** as SingleChildScrollView **mounts and paints the entire DataTable even when only some rows are visible**. If scrolling in one direction is necessary, then consider using a CustomScrollView, otherwise use PaginatedDataTable to split the data into smaller pages.

← Data table

Name	Age	Role
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager
Charlie	50	Director
Alice	30	Software engineer
Bob	40	Manager

← Paginated data table

Episode ↑	Title
S1E01	Strange New Worlds
S1E02	Children of the Comet
S1E03	Ghosts of Illyria
S1E04	Memento Mori
S1E05	Spock Amok
S1E06	Lift Us Where Suffering Canno
S1E07	The Serene Squall
S1E08	The Elysian Kingdom
S1E09	All Those Who Wander
S2E10	A Quality of Mercy

1-10 of 20 < >



Как реализовывался 2D-скролл в «среднем»

Новая документация DataTable

Performance considerations

Columns are sized automatically based on the table's contents. It's **expensive** to display **large amounts of data with this widget, since it must be measured twice**: once to negotiate each column's dimensions, and again when the table is laid out.

A **SingleChildScrollView** mounts and paints the entire child, even when only some is visible. For a table that effectively handles large amounts of data, here are some options to consider:

- **TableView, a widget from the two_dimensional_scrollables package.**
- **PaginatedDataTable**, which automatically splits the data into multiple pages.
- **CustomScrollView**, for greater control over scrolling effects.



Современный ПОДХОД

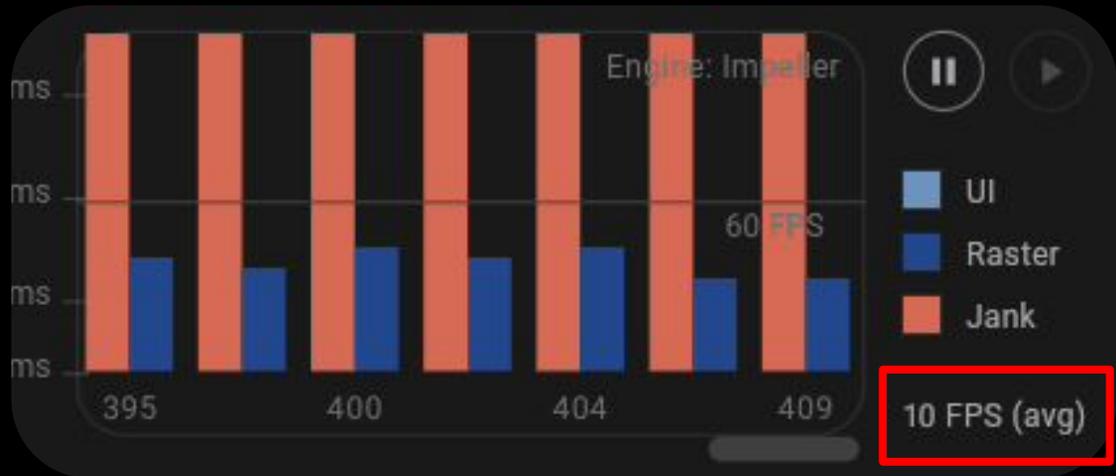
Что такое TwoDimensionalScrollView

Базовый класс

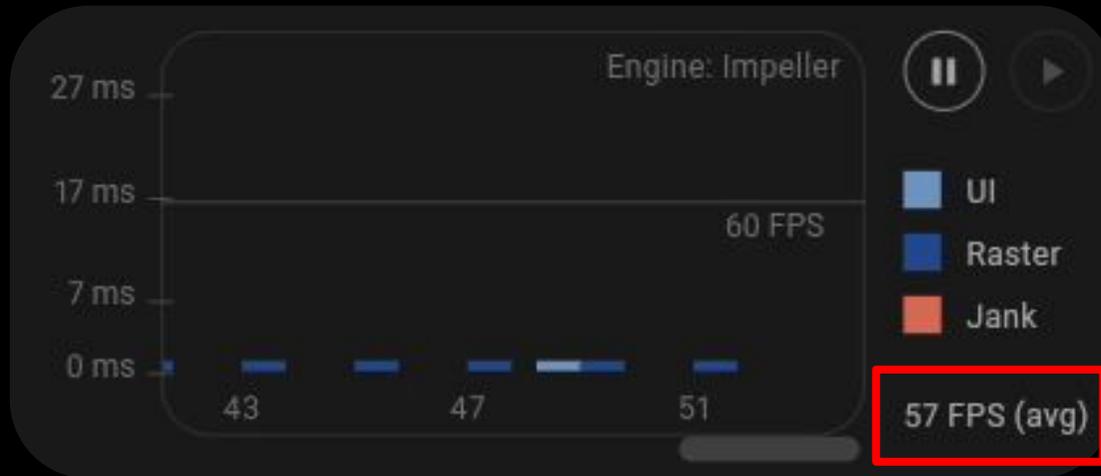
```
abstract class TwoDimensionalScrollView extends StatelessWidget {  
  const TwoDimensionalScrollView({  
    super.key,  
    this.primary,  
    this.mainAxis = Axis.vertical,  
    this.verticalDetails = const ScrollableDetails.vertical(),  
    this.horizontalDetails = const ScrollableDetails.horizontal(),  
    required this.delegate,  
    this.cacheExtent,  
    this.diagonalDragBehavior = DiagonalDragBehavior.none,  
    this.dragStartBehavior = DragStartBehavior.start,  
    this.keyboardDismissBehavior,  
    this.clipBehavior = Clip.hardEdge,  
    this.hitTestBehavior = HitTestBehavior.opaque,  
  });  
}
```



DataTable



TableView.builder



0:0	0:1	0:2	0:3	0:4	0:5	0:6	0:7	0:8	0:9
1:0	1:1	1:2	1:3	1:4	1:5	1:6	1:7	1:8	1:9
2:0	2:1	2:2	2:3	2:4	2:5	2:6	2:7	2:8	2:9
3:0	3:1	3:2	3:3	3:4	3:5	3:6	3:7	3:8	3:9
4:0	4:1	4:2	4:3	4:4	4:5	4:6	4:7	4:8	4:9
5:0	5:1	5:2	5:3	5:4	5:5	5:6	5:7	5:8	5:9
6:0	6:1	6:2	6:3	6:4	6:5	6:6	6:7	6:8	6:9
7:0	7:1	7:2	7:3	7:4	7:5	7:6	7:7	7:8	7:9
8:0	8:1	8:2	8:3	8:4	8:5	8:6	8:7	8:8	8:9
9:0	9:1	9:2	9:3	9:4	9:5	9:6	9:7	9:8	9:9

DataTable +
SingleChildScrollView

100

виджета в build()

0:0	0:1	0:2	0:3	0:4	0:5	0:6	0:7	0:8	0:9
1:0	1:1	1:2	1:3	1:4	1:5	1:6	1:7	1:8	1:9
2:0	2:1	2:2	2:3	2:4	2:5	2:6	2:7	2:8	2:9
3:0	3:1	3:2	3:3	3:4	3:5	3:6	3:7	3:8	3:9
4:0	4:1	4:2	4:3	4:4	4:5	4:6	4:7	4:8	4:9
5:0	5:1	5:2	5:3	5:4	5:5	5:6	5:7	5:8	5:9
6:0	6:1	6:2	6:3	6:4	6:5	6:6	6:7	6:8	6:9
7:0	7:1	7:2	7:3	7:4	7:5	7:6	7:7	7:8	7:9
8:0	8:1	8:2	8:3	8:4	8:5	8:6	8:7	8:8	8:9
9:0	9:1	9:2	9:3	9:4	9:5	9:6	9:7	9:8	9:9

TableView

32

виджета в build()



Решения из коробки

two_dimensional_scrollables

two_dimensional_scrollables 0.3.3

Published 3 months ago • flutter.dev Dart 3 compatible

SDK | FLUTTER | PLATFORM | ANDROID | IOS | LINUX | MACOS | WEB | WINDOWS 417

[Readme](#) | [Changelog](#) | [Example](#) | [Installing](#) | [Versions](#) | [Scores](#)

Two Dimensional Scrollables

A package that provides widgets that scroll in two dimensions, built on the two-dimensional foundation of the Flutter framework.

Features

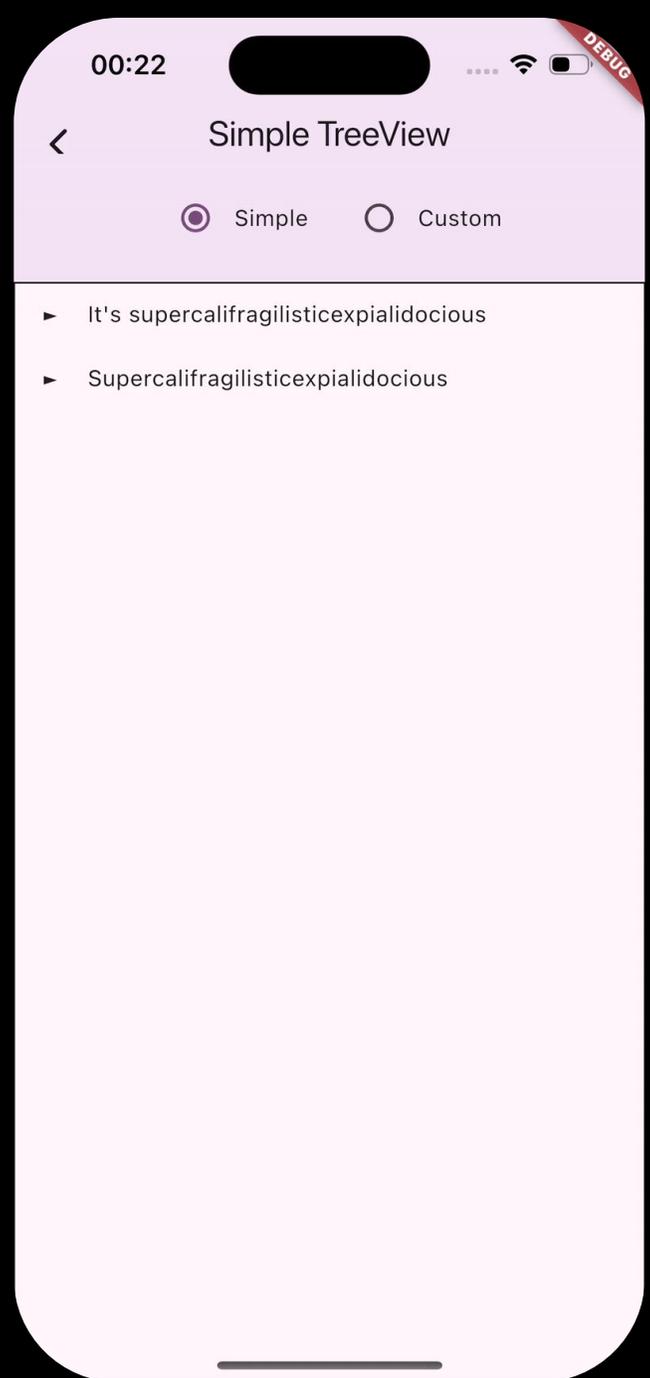
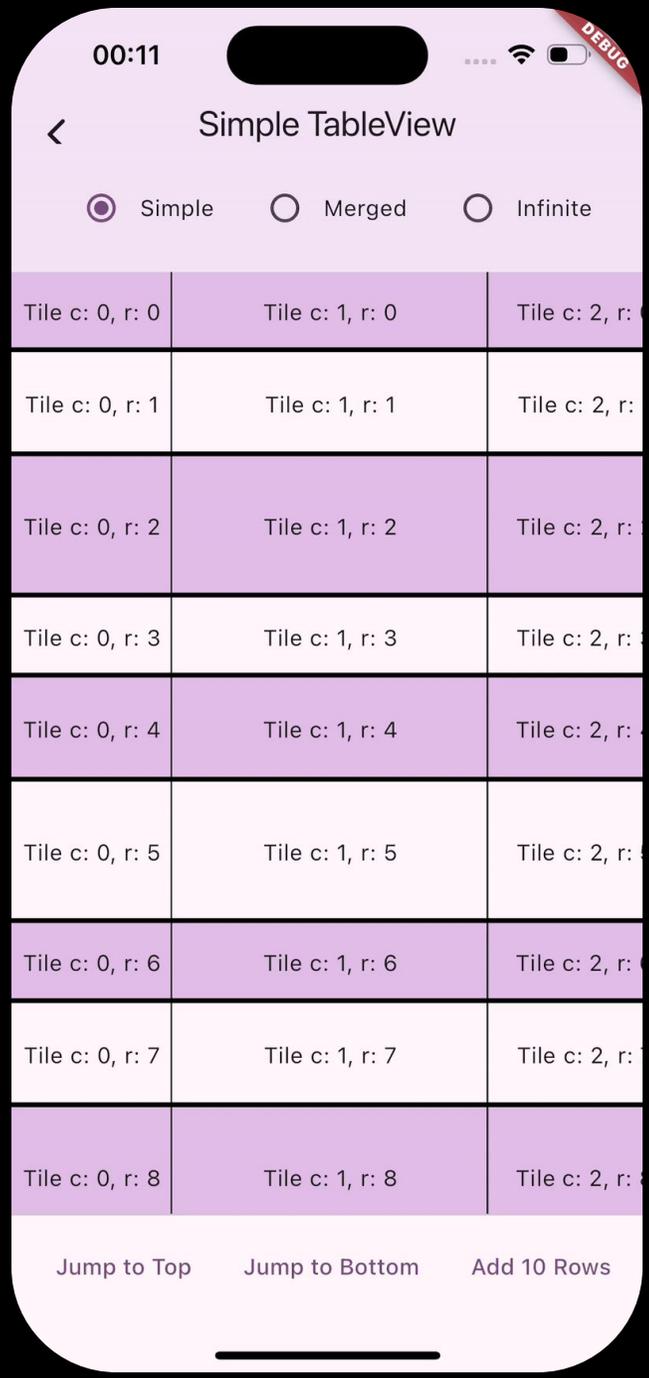
This package provides support for TableView and TreeView widgets that scroll in both the vertical and horizontal axes.

417 LIKES | 160 POINTS | 56.6k DOWNLOADS

Publisher
flutter.dev

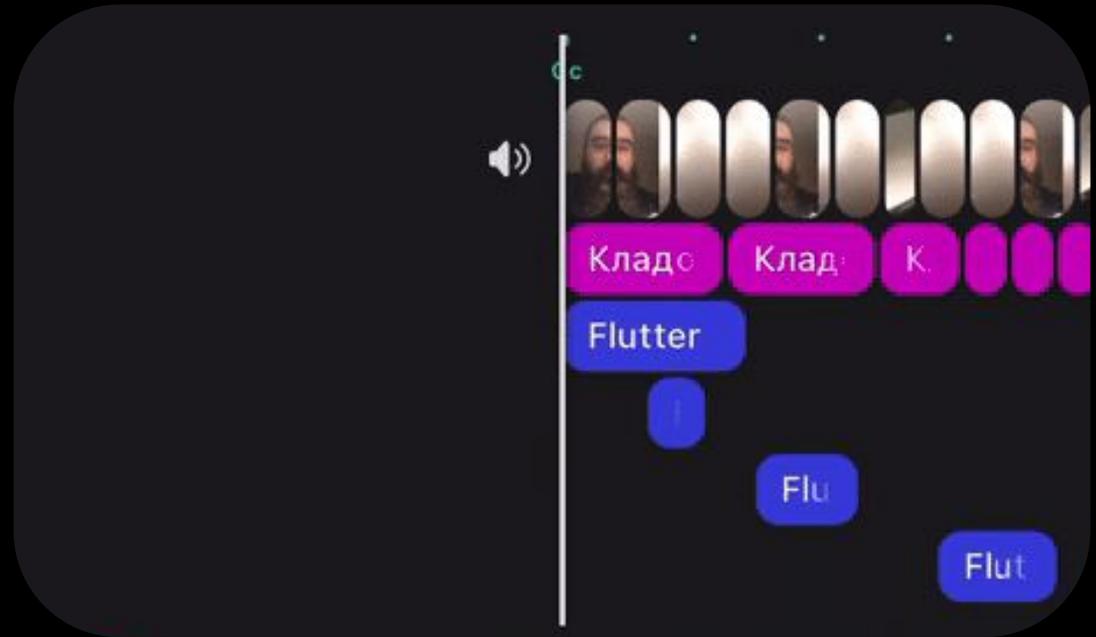
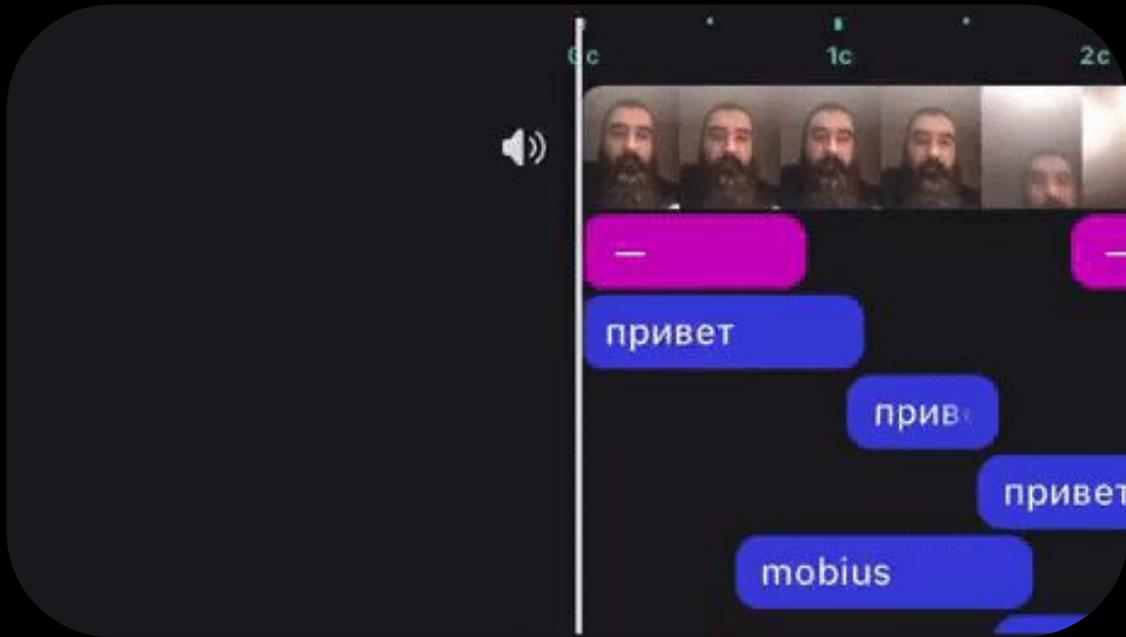
Weekly Downloads

2024.07.08 - 2025.01.20



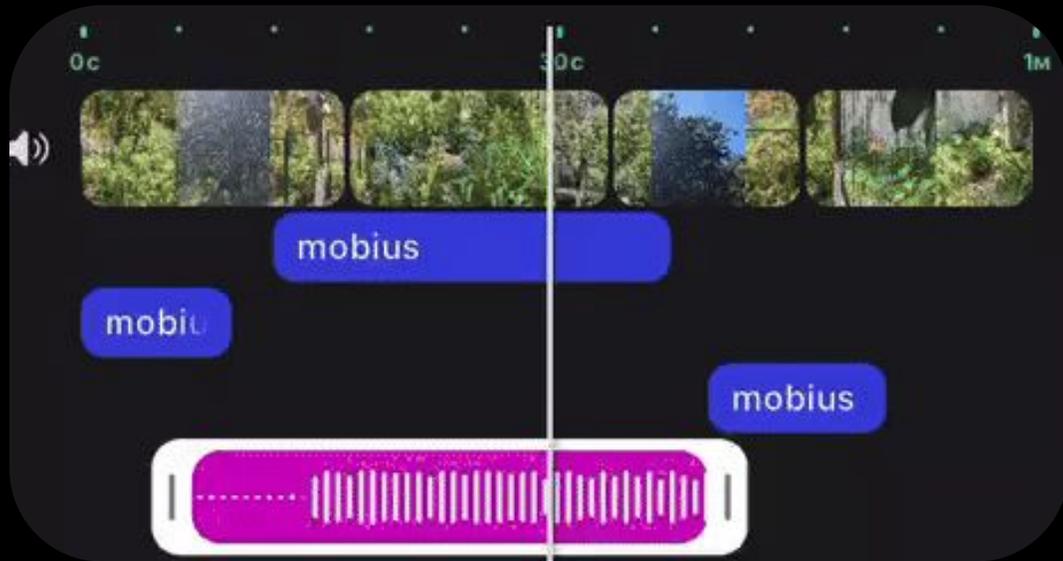
Видеоредактор Yappy

Реальное приложение



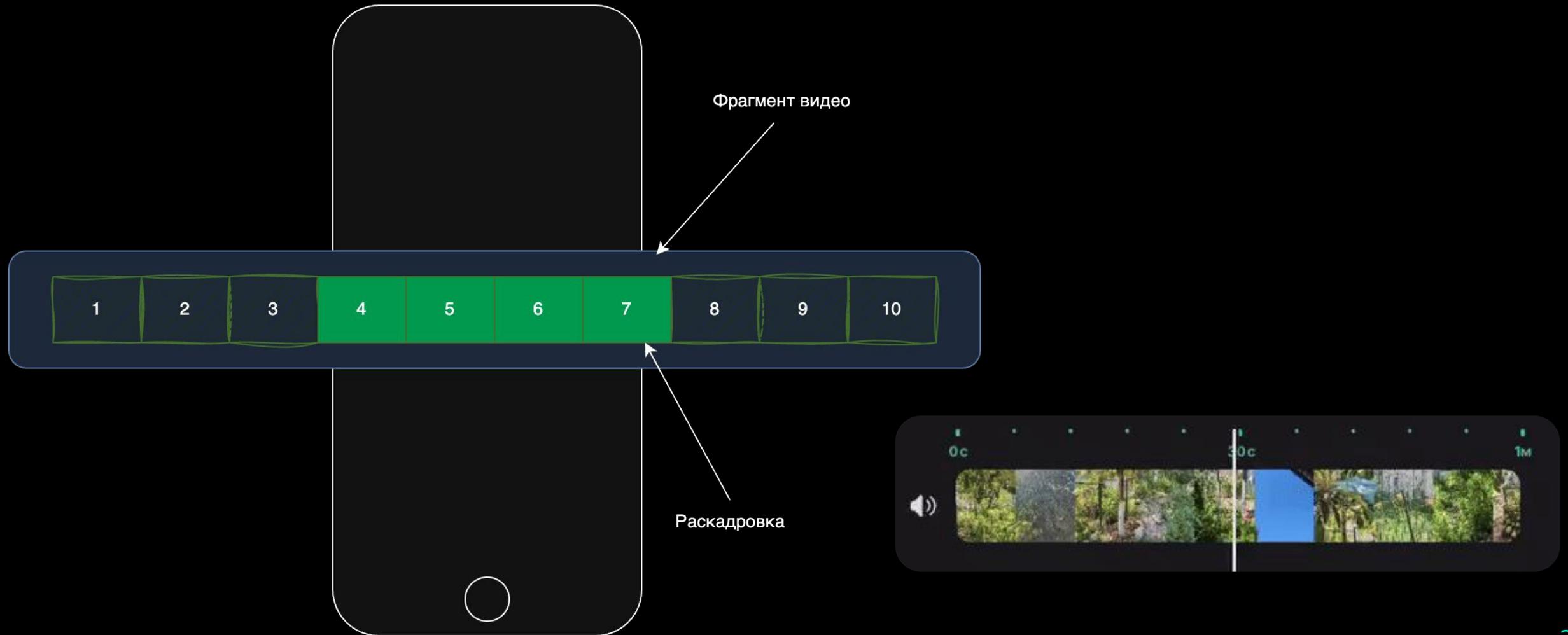
Причины собственной реализации

Динамический размер фрагментов



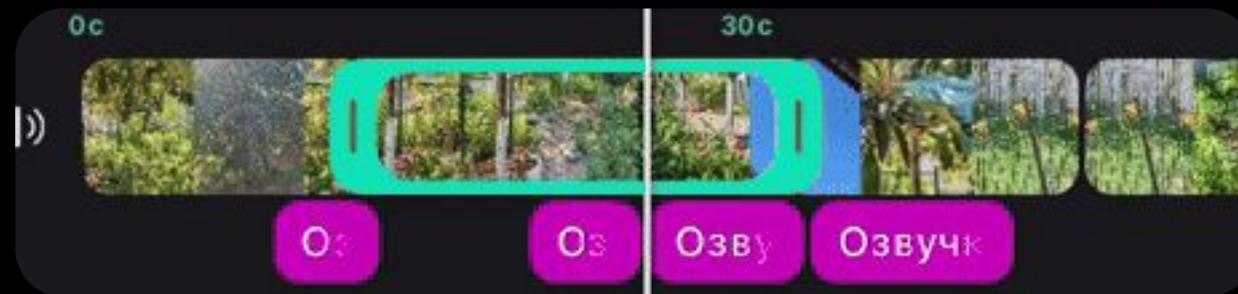
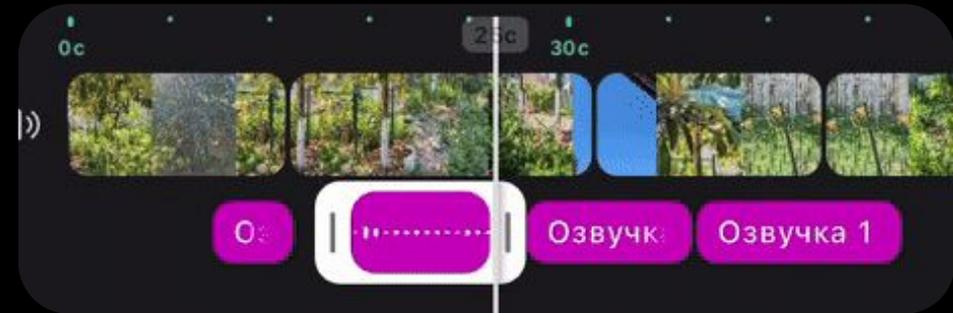
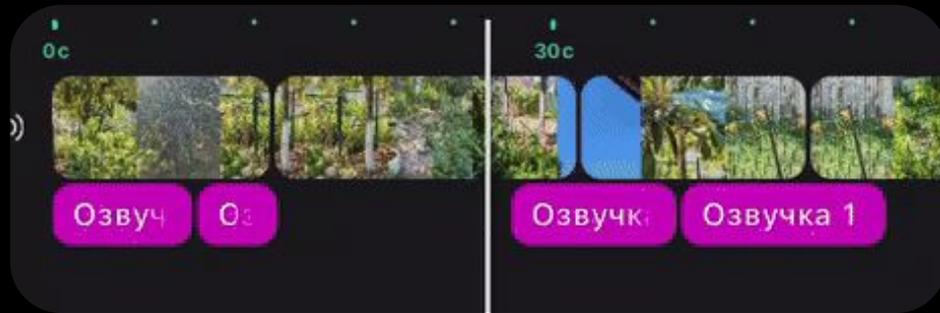
Причины собственной реализации

Списки внутри фрагментов



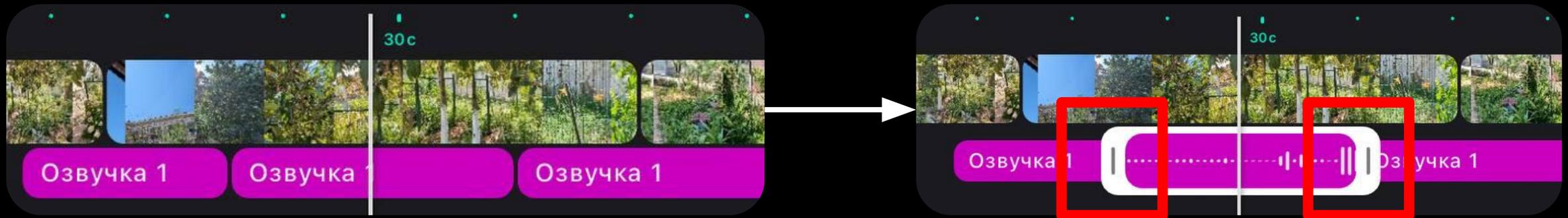
Причины собственной реализации

Взаимодействие с фрагментами



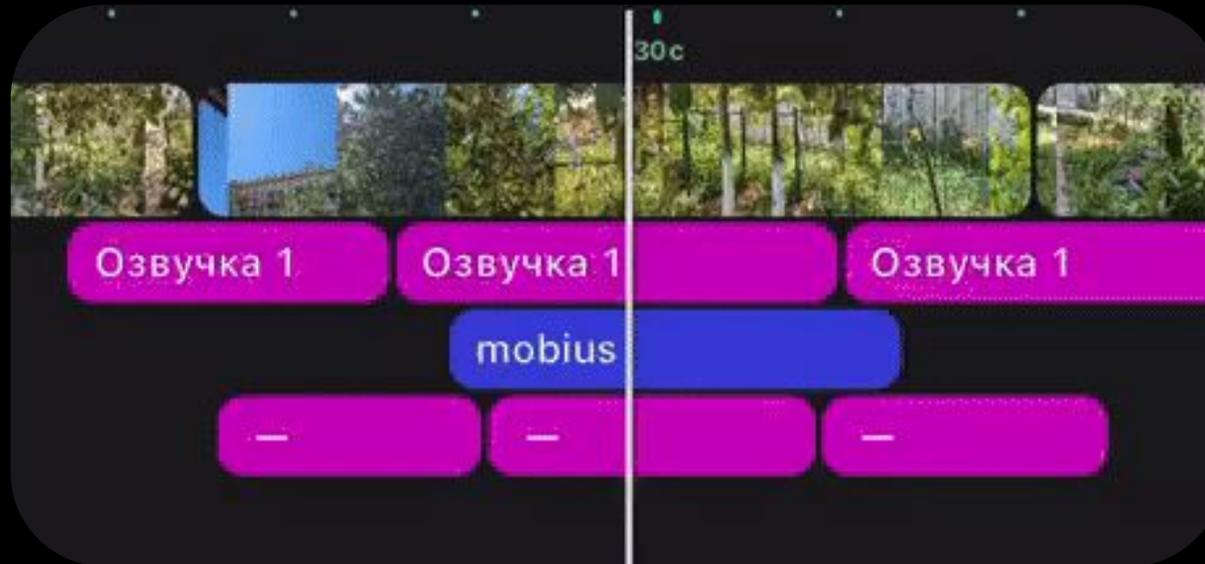
Причины собственной реализации Z index

- Stack
- Overlay
- Custom layout



Причины собственной реализации

Пятая причина — анимации





Собственная реализация

Собственная реализация

Что будем делать мы?

2:00
План

7:00
Введение

17:00
Современный под...

27:00
Собственная реализация

3

0:00
Вступительный лекция

10:00
План

15:00
Современный подход

```
RenderTwoDimensionalViewport createRenderObject(RenderContext context) {  
    throw UnimplementedError();  
}  
@override  
void render(RenderContext context)
```

днем"

7:00							17:00
Введение	Row 3, Column 4	Row 3, Column 5	Row 3, Column 6	Row 3, Column 7			Современный п

0:00 **Вступление**

 **Fedor Blagodyr**
Flutter Software Engine

Опишем модель



Собственная реализация — модель

Расписание

```
class MobiusSchedule {  
    static const capacity = Duration(hours: 4);  
    final String id;  
    final Iterable<MobiusSpeech> speeches;  
}
```

Выступление

```
class MobiusSpeech {  
    static const reservedDuration = Duration(hours: 1);  
    final String id;  
    final String title;  
    final Duration duration;  
    final Duration start;  
    final Iterable<MobiusSpeechTimestamp> timestamps;  
}
```

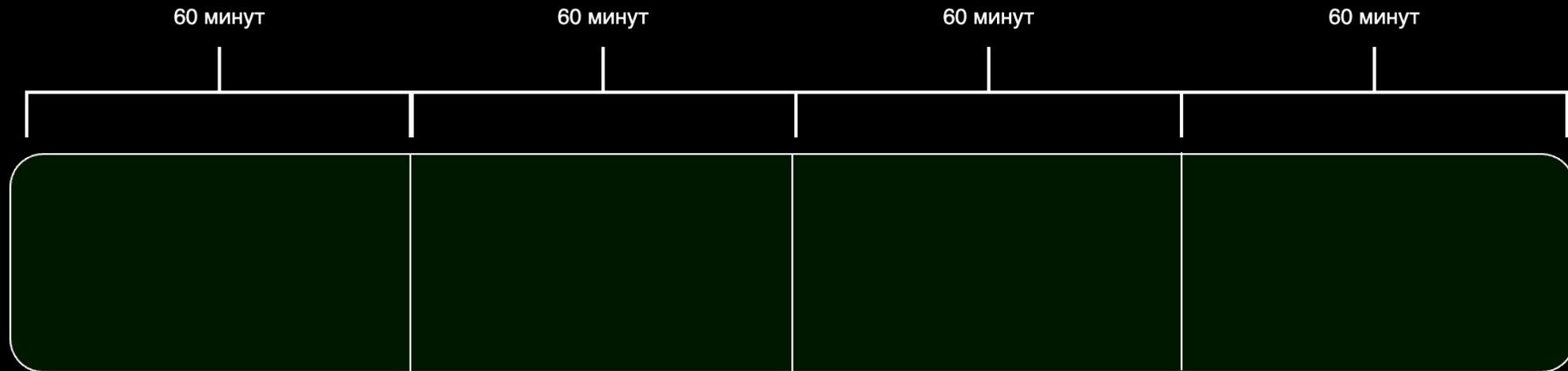
Собственная реализация — модель

Временная метка

```
class MobiusSpeechTimestamp {  
    final Duration startsAt;  
    final Duration duration;  
    final String title;  
    final String? description;  
    final String coverUrl;  
}
```

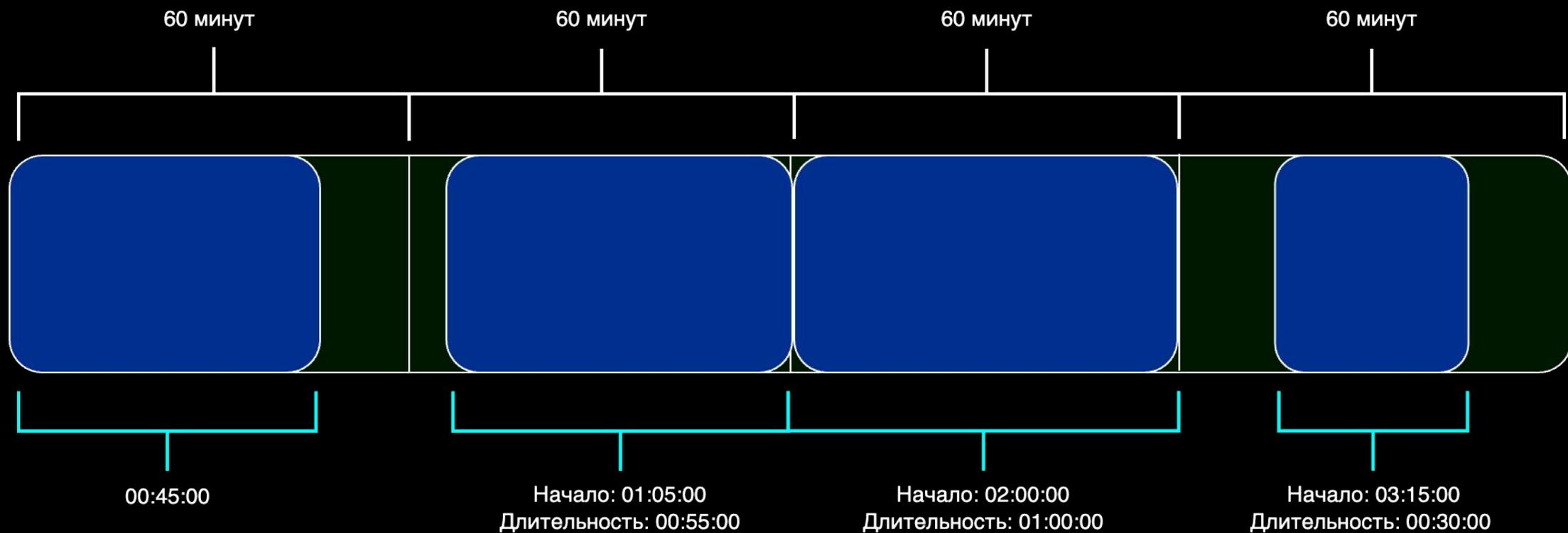
Собственная реализация — визуализация

Расписание



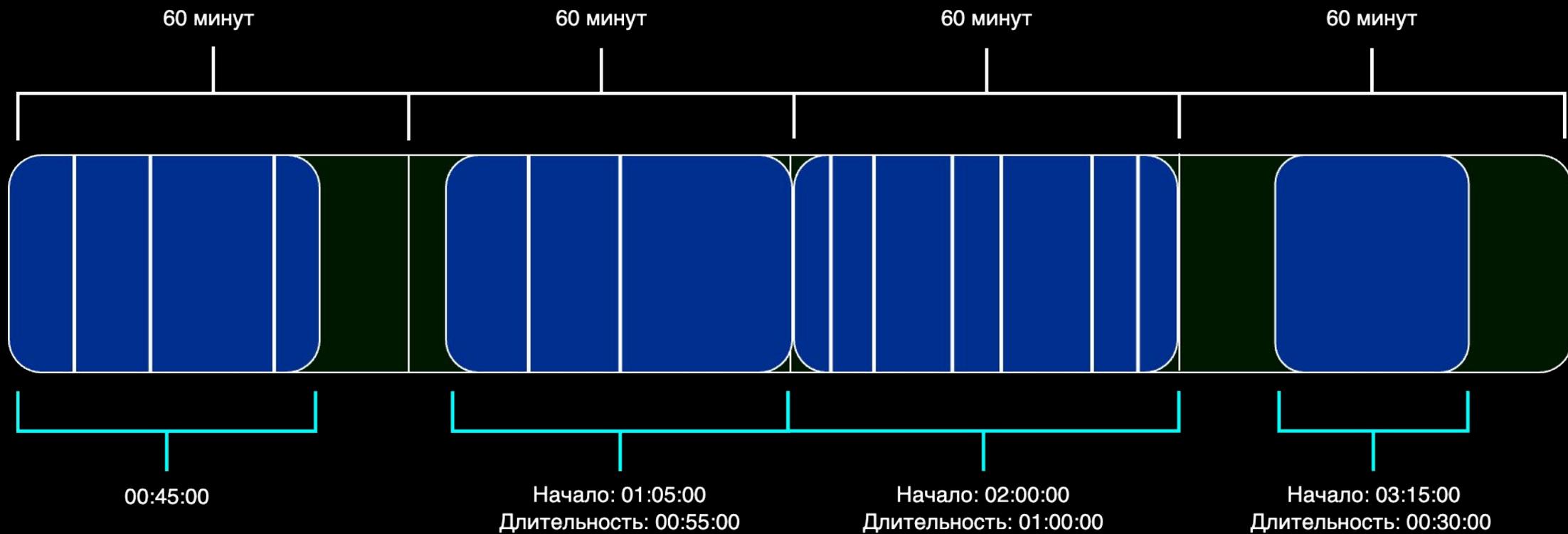
Собственная реализация — визуализация

Выступления

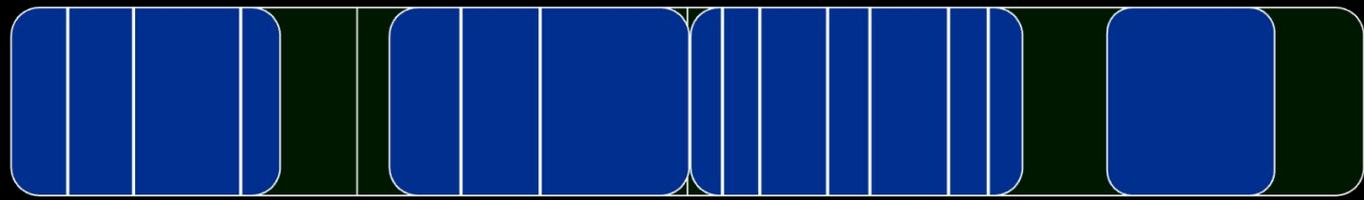


Собственная реализация — визуализация

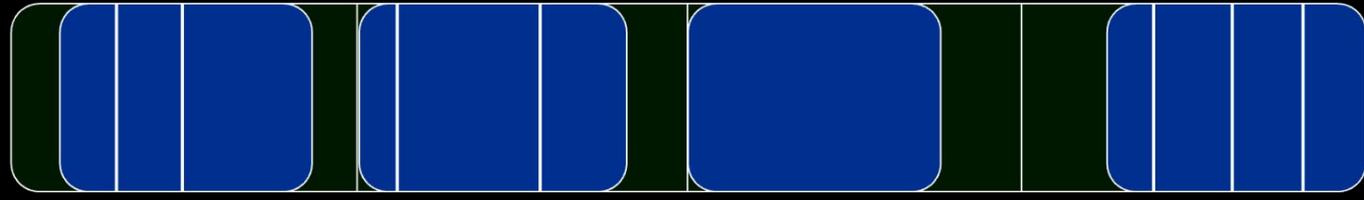
Временные метки



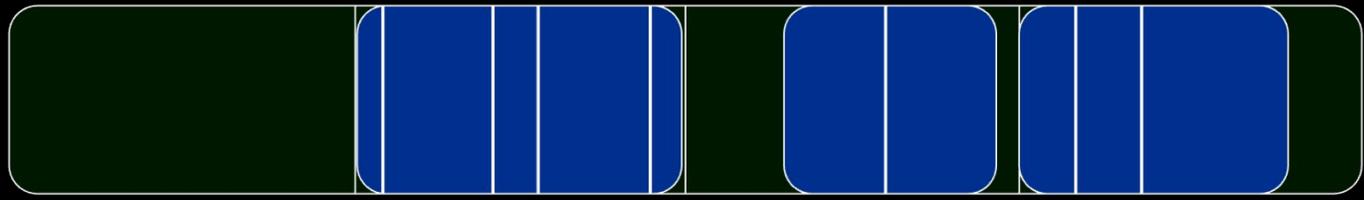
Понедельник



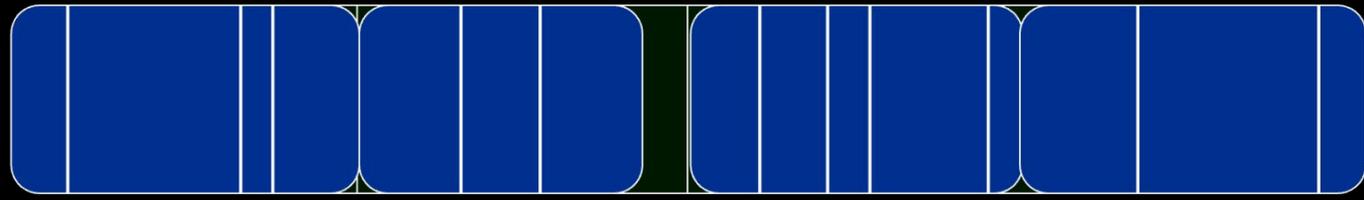
Вторник



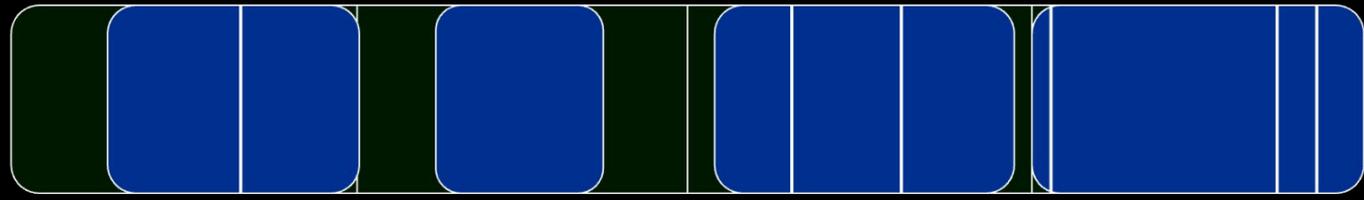
Среда

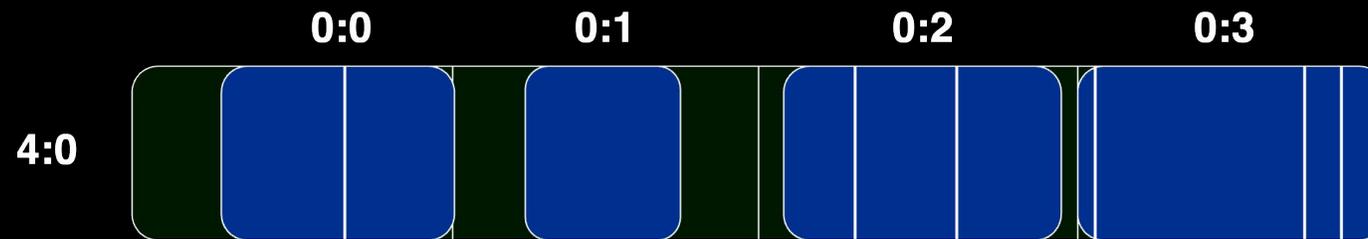
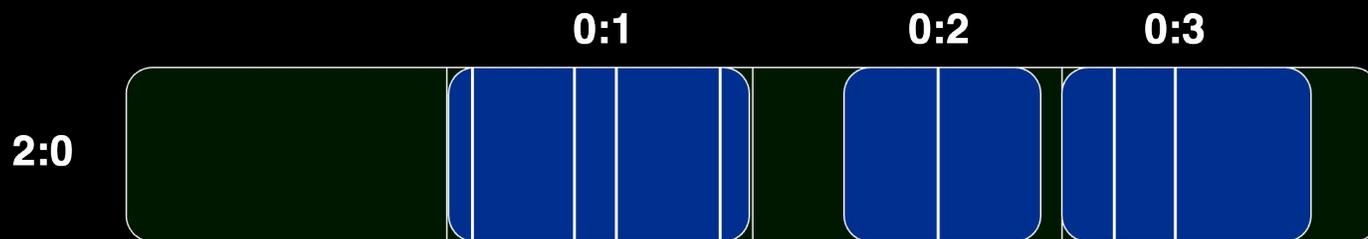
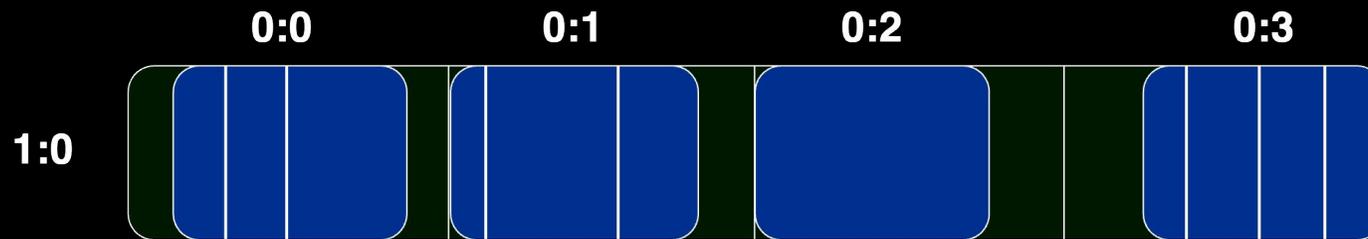
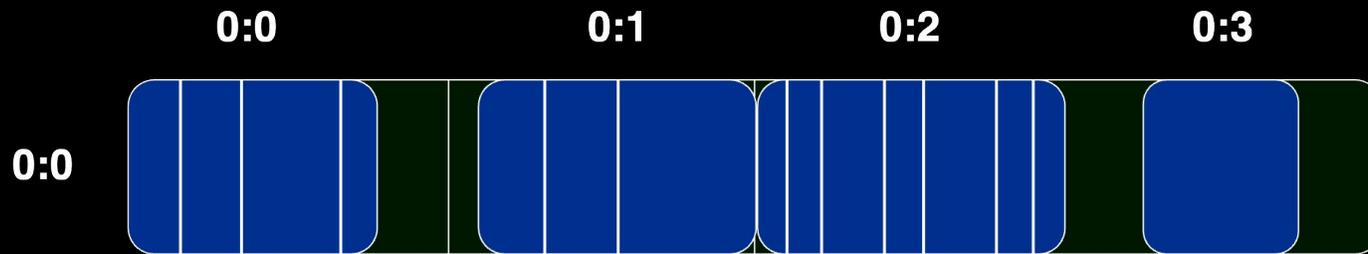


Четверг



Пятница 🕶️







Опишем UI



Данные для построения UI

Таблица

```
class MobiusScheduleViewData {  
    final Iterable<MobiusScheduleViewRow> rows;  
    final double width;  
    final int maxCellsCount;  
    final String? selectedSpeechId;  
}
```

Данные для построения UI

Строка

```
class MobiusScheduleViewRow {  
    final String scheduleId;  
    final bool hasSelectedCell;  
    final Iterable<MobiusScheduleViewCell> cells;  
    final double collapsedHeight;  
    final double expandedHeight;  
}
```

Данные для построения UI

Ячейка

```
class MobiusScheduleViewCell{  
    final double startsAt;  
    final double width;  
    final double expandedHeight;  
    final double collapsedHeight;  
    final String speechId;  
    final bool isSelected;  
}
```

Преобразование длительности в пиксели

```
const _durationFactor = 1000000;

extension DurationX on Duration {
  double toWidth(double screenWidth) =>
    inMicroseconds * screenWidth / (MobiusSchedule.capacity.inSeconds * _durationFactor);
}

extension DoubleX on double {
  Duration toDuration(double screenWidth) =>
    Duration(
      microseconds: this * MobiusSchedule.capacity.inSeconds * _durationFactor ~/ screenWidth,
    );
}
```

Таблица

```
factory MobiusScheduleViewData({
    required Iterable<MobiusSchedule> schedules,
    required double screenWidth,
    String? selectedSpeechId,
}) {
    // Вычисления
}
```

```
var width = 0.0;
final rows = <MobiusScheduleViewRow>[];
for (final schedule in schedules) {
  final row = MobiusScheduleViewRow(
    schedule: schedule,
    screenWidth: screenWidth,
    selectedSpeechId: selectedSpeechId,
  );
  rows.add(row);
  maxCellsCount = max(maxCellsCount, row.cells.length);
  width = max(
    width,
    // Каждое выступление имеет выделенную длительность 1 час
    (MobiusSpeech.reservedDuration * row.cells.length).toWidth(screenWidth),
  );
}
```

Строка

```
factory MobiusScheduleViewRow({
    required MobiusSchedule schedule,
    required double screenWidth,
    required String? selectedSpeechId,
}) {
    // Вычисления
}
```

```
var (hasSelectedCell, maxExpandedHeight, maxCollapsedHeight) = (false, 0.0, 0.0);
final cells = <MobiusScheduleViewCell>[];
for (final speech in schedule.speeches) {
    final isSelected = speech.id == selectedSpeechId;
    if (isSelected) hasSelectedCell = true;
    final cell = MobiusScheduleViewCell(
        speech: speech,
        isSelected: isSelected,
        screenWidth: screenWidth,
    );
    cells.add(cell);
    maxExpandedHeight = math.max(maxExpandedHeight, cell.expandedHeight);
    maxCollapsedHeight = math.max(maxCollapsedHeight, cell.collapsedHeight);
}
cells.sort((a, _) => a.speechId == selectedSpeechId ? 1 : 0); // Z index
```

Ячейка

```
factory MobiusScheduleViewCell({
  required MobiusSpeech speech,
  required double screenWidth,
  required bool isSelected,
}) {
  return MobiusScheduleViewCell._(
    startsAt: speech.start.toWidth(screenWidth),
    width: speech.duration.toWidth(screenWidth),
    speechId: speech.id,
    isSelected: isSelected,
    collapsedHeight: 100,
    expandedHeight: 250,
  );
}
```



`TwoDimensionalScrollView`

Widget

```
class MobiusScheduleScrollView extends TwoDimensionalScrollView {  
  final MobiusScheduleViewData data;  
  final TickerProvider vsync;  
  /*constructor*/  
  @override  
  Widget buildViewport(  
    BuildContext context,  
    ViewportOffset verticalOffset,  
    ViewportOffset horizontalOffset,  
  ) {  
    throw UnimplementedError();  
  }  
}
```

TwoDimensionalScrollView

Viewport

```
class MobiusScheduleViewport extends TwoDimensionalViewport {  
    /*constructor*/  
    @override  
    RenderTwoDimensionalViewport createRenderObject(BuildContext context) {  
        // createRenderObject  
    }  
    @override  
    void updateRenderObject(  
        BuildContext context,  
        RenderMobiusScheduleViewport renderObject,  
    ) {  
        // updateRenderObject  
    }  
}
```

`TwoDimensionalScrollView`

RenderObject

```
class RenderScheduleMobiusViewport extends RenderTwoDimensionalViewport {  
  /*constructor*/  
  
  @override  
  void layoutChildSequence () {  
    // боль  
  }  
}
```

LayoutChildSequence

```
abstract class RenderTwoDimensionalViewport extends RenderBox {  
    void layoutChildSequence();  
  
    @override  
    void performLayout() {  
        _firstChild = null;  
        _lastChild = null;  
        _activeChildrenForLayoutPass.clear();  
        _childManager._startLayout();  
  
        // Subclass lays out children.  
        layoutChildSequence();  
        // ...  
    }  
}
```

LayoutChildSequence

```
final horizontalPixels = horizontalOffset.pixels;  
final verticalPixels = verticalOffset.pixels;
```

```
final viewportWidth = viewportDimension.width + cacheExtent;  
final viewportHeight = viewportDimension.height + cacheExtent;
```

```
final rowCount = _data.rows.length;  
var allRowsHeight = 0.0;  
var totalHeight = 0.0;
```

LayoutChildSequence

```
for (var rowIndex = 0; rowIndex < rowCount; rowIndex++) {  
    final row = _data.rows.elementAt(rowIndex);  
    final rowHeight = row.hasSelectedCell ? row.expandedHeight : row.collapsedHeight;  
    final rowStarts = allRowsHeight;  
    final rowEnds = rowStarts + rowHeight;  
  
    final isPartiallyVisibleAbove = rowStarts < verticalPixels + viewportHeight;  
    final isPartiallyVisibleBelow = rowEnds > verticalPixels;  
    final isVisibleVertically = isPartiallyVisibleAbove && isPartiallyVisibleBelow;  
    totalHeight = math.max(rowEnds, totalHeight);  
  
    if (!isVisibleVertically) {  
        allRowsHeight += rowHeight;  
        continue;  
    }  
}
```

LayoutChildSequence

```
for (var cellIndex = 0; cellIndex < row.cells.length; cellIndex++) {  
    final cell = row.cells.elementAt(cellIndex);  
    final cellWidth = cell.width;  
    final cellStarts = cell.startsAt;  
    final cellEnds = cellStarts + cellWidth;  
  
    if (cellStarts > horizontalPixels + viewportWidth) continue;  
    if (cellEnds < horizontalPixels) continue;  
    //...  
}
```

LayoutChildSequence

```
//...  
final vicinity = ChildVicinity(  
    xIndex: cellIndex,  
    yIndex: rowIndex,  
);  
  
final child = buildOrObtainChildFor(vicinity) !  
    ..layout(  
        BoxConstraints(  
            maxWidth: cellWidth,  
            minHeight: cellHeight,  
            maxHeight: rowHeight,  
        ).normalize(),  
    );  
  
parentDataOf(child).layoutOffset = Offset(  
    cellStarts - horizontalPixels,  
    rowStarts - verticalPixels,  
);
```

TwoDimensionalScrollView

LayoutChildSequence

```
verticalOffset.applyContentDimensions(  
    0,  
    (totalHeight - viewportDimension.height).clamp(0, double.infinity),  
);  
  
horizontalOffset.applyContentDimensions(  
    0,  
    (data.width - viewportDimension.width).clamp(0, double.infinity),  
);
```

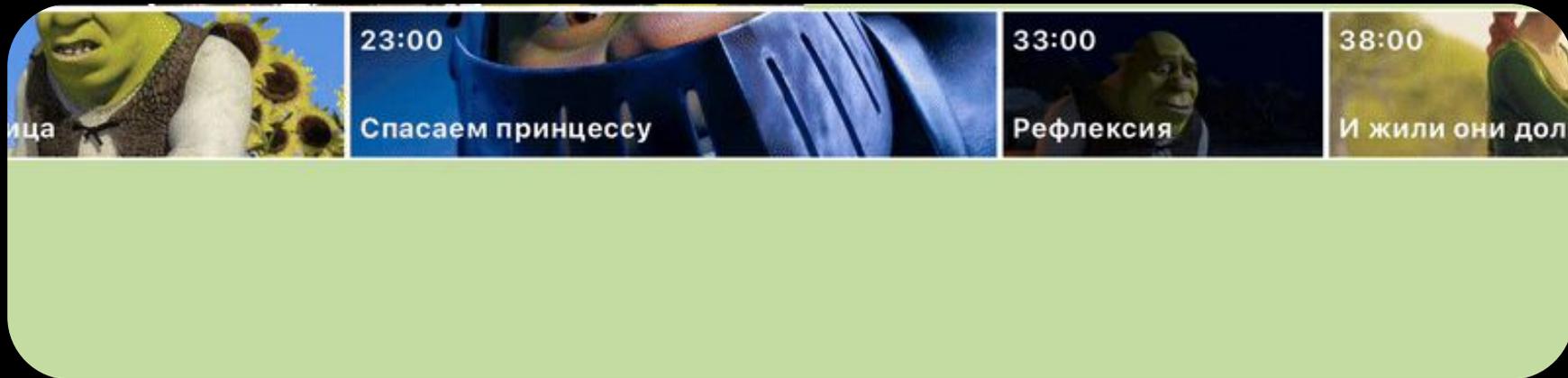
```
return MobiusScheduleView(  
    data: _data,  
    delegate: TwoDimensionalChildBuilderDelegate(  
        maxXIndex: _data.maxCellsCount - 1,  
        maxYIndex: _data.rows.length - 1,  
        builder: (context, vicinity) {  
            final row = _data.rows.elementAtOrNull(vicinity.yIndex);  
            if (row == null) return null;  
            final cell = row.cells.elementAtOrNull(vicinity.xIndex);  
            if (cell == null) return null;  
            return MyCoolCellWidget();  
        },  
    ),  
);
```

Важные вещи, о которых я не сказал

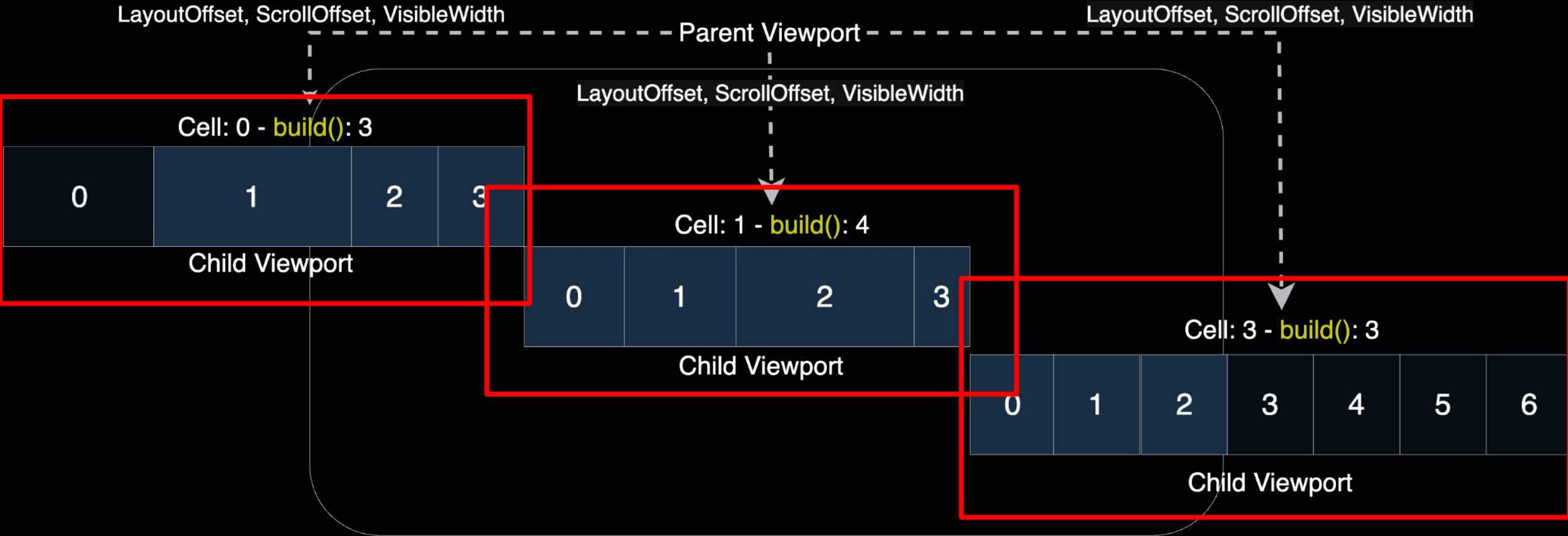
- Анимация выбранной ячейки
- Список внутри ячейки

Анимация выбранной ячейки

```
final rowHeightTween = _rowTweens[row.scheduleId] ?? Tween(  
    begin: row.collapsedHeight,  
    end: row.hasSelectedCell ? row.expandedHeight : row.collapsedHeight,  
);  
final rowHeight = rowHeightTween.evaluate(_animationController);
```



Список внутри ячейки



Список внутри ячейки

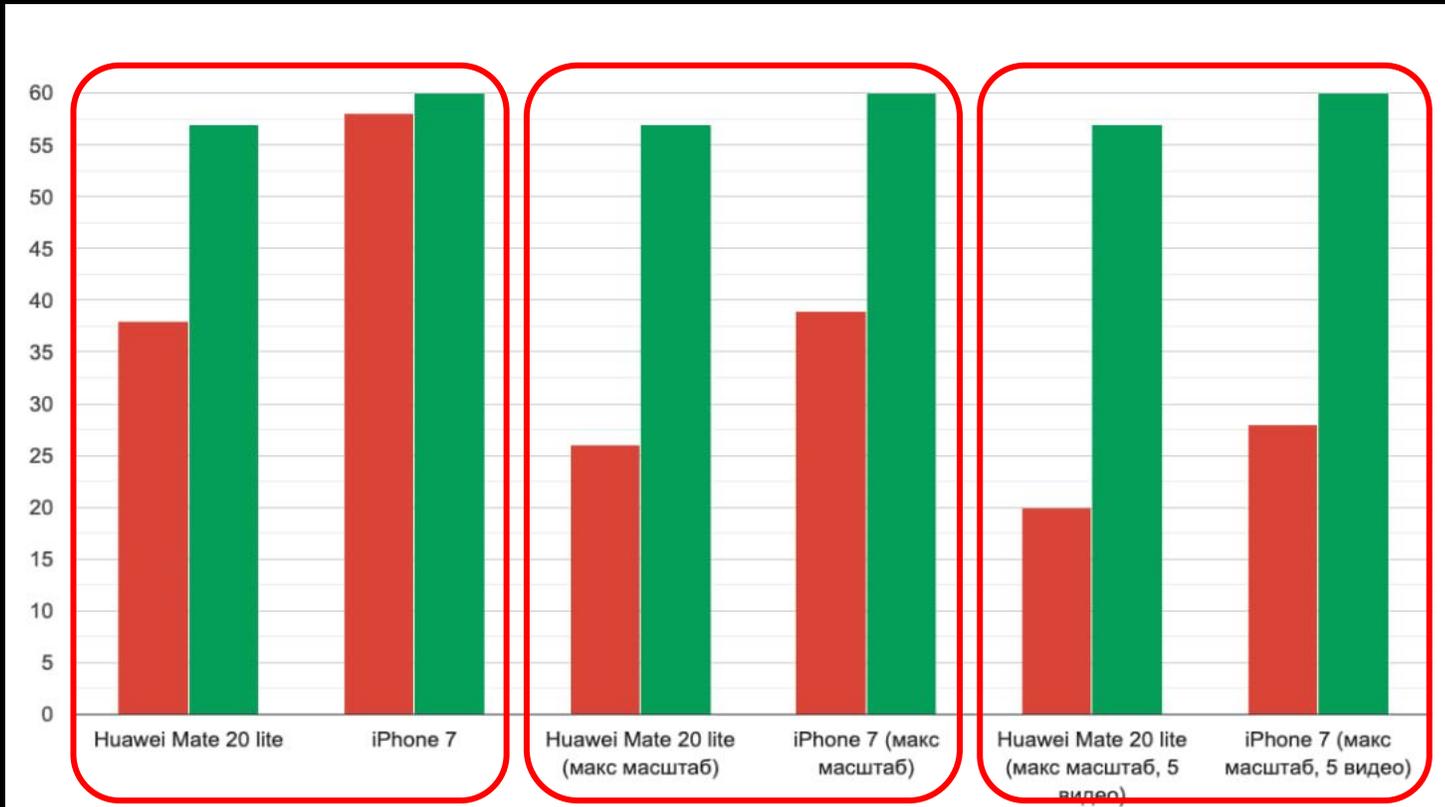
```
class RenderChildViewport extends RenderViewport {  
    @override  
    double layoutChildSequence(/*method arguments*/) {  
        final data = delegateLayout(/*method arguments*/);  
        return super.layoutChildSequence(/*delegated data arguments*/);  
    }  
}
```



Влияние на производительность приложения

Перемотка таймлайна

Средний FPS

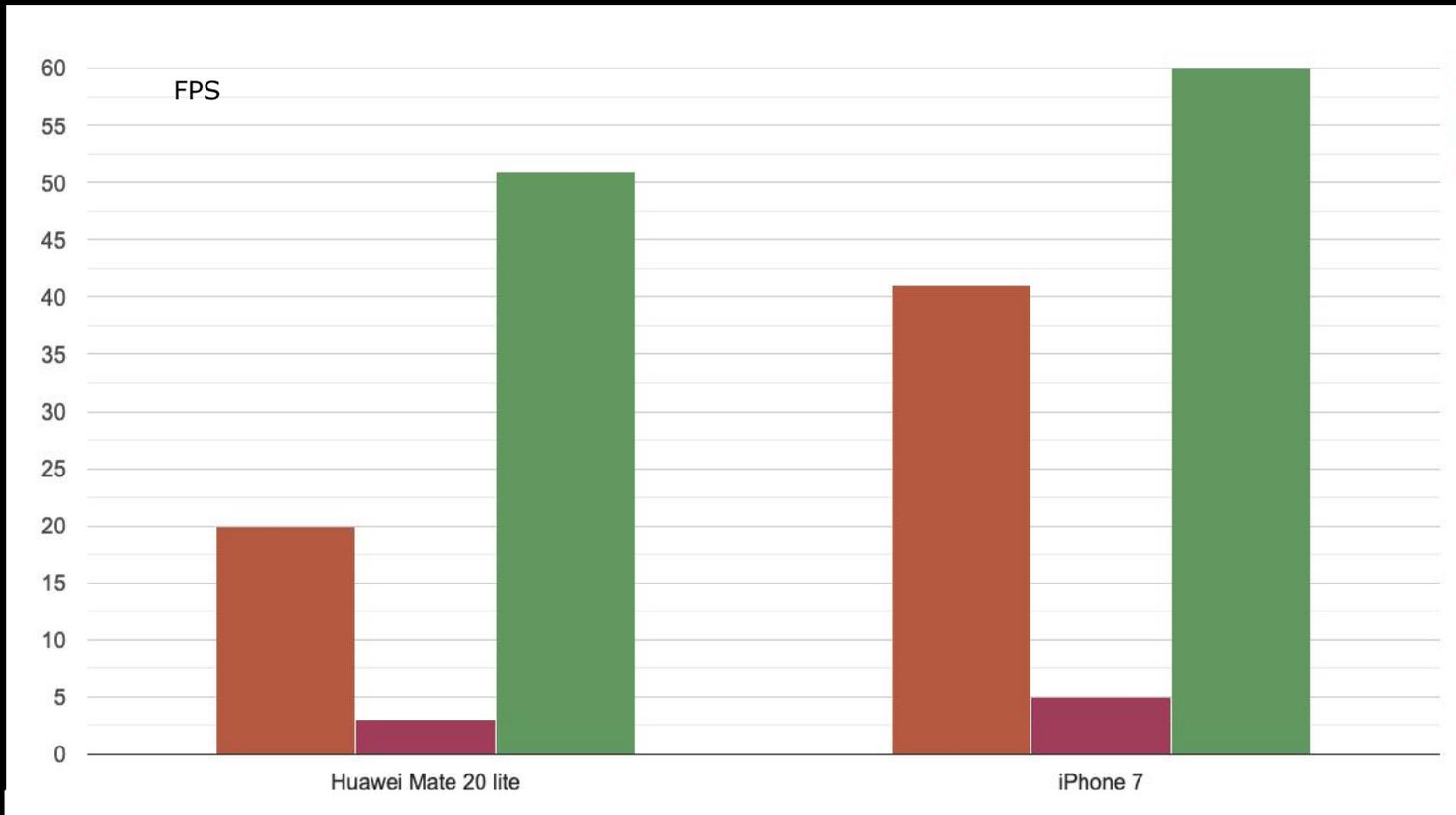


Перемотка таймлайна, 10 сек. видео. ■ было ■ стало

Влияние на производительность приложения

Масштабирование таймлайна

Средний FPS



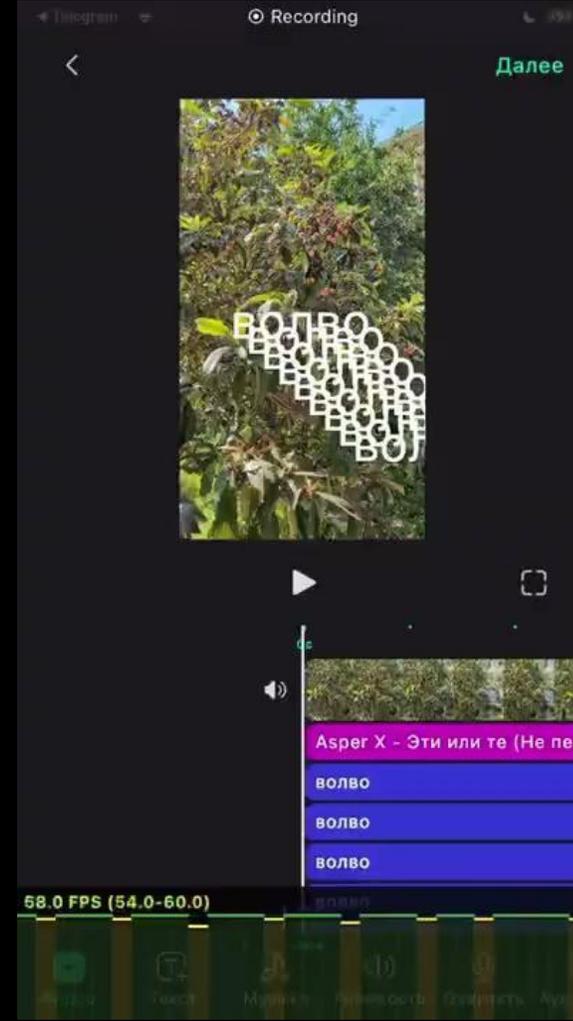
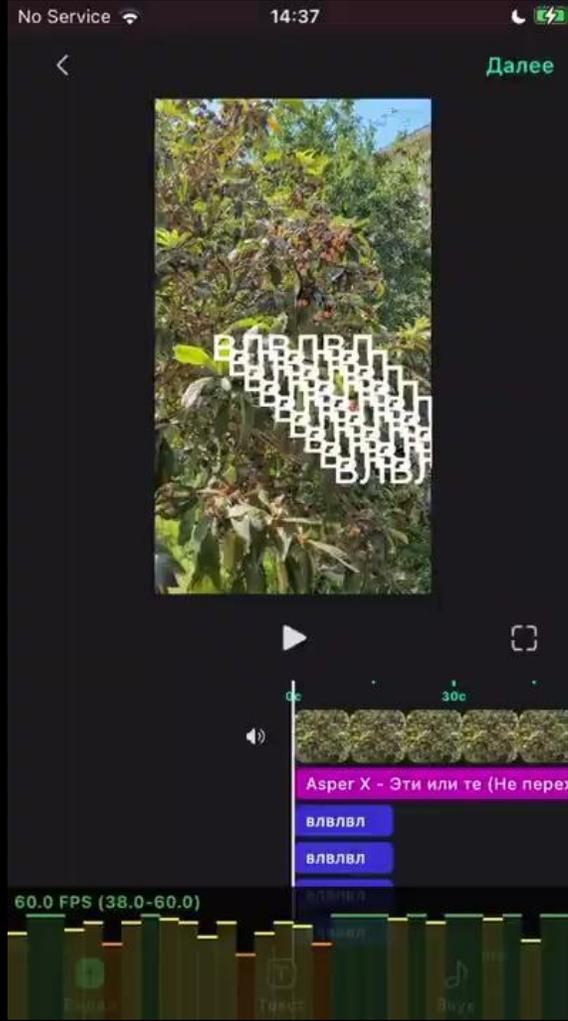
Полный масштаб туда и обратно 10 сек. видео.

■ было
 ■ min было
 ■ стало

Влияние на производительность приложения

Работа с таймлайном

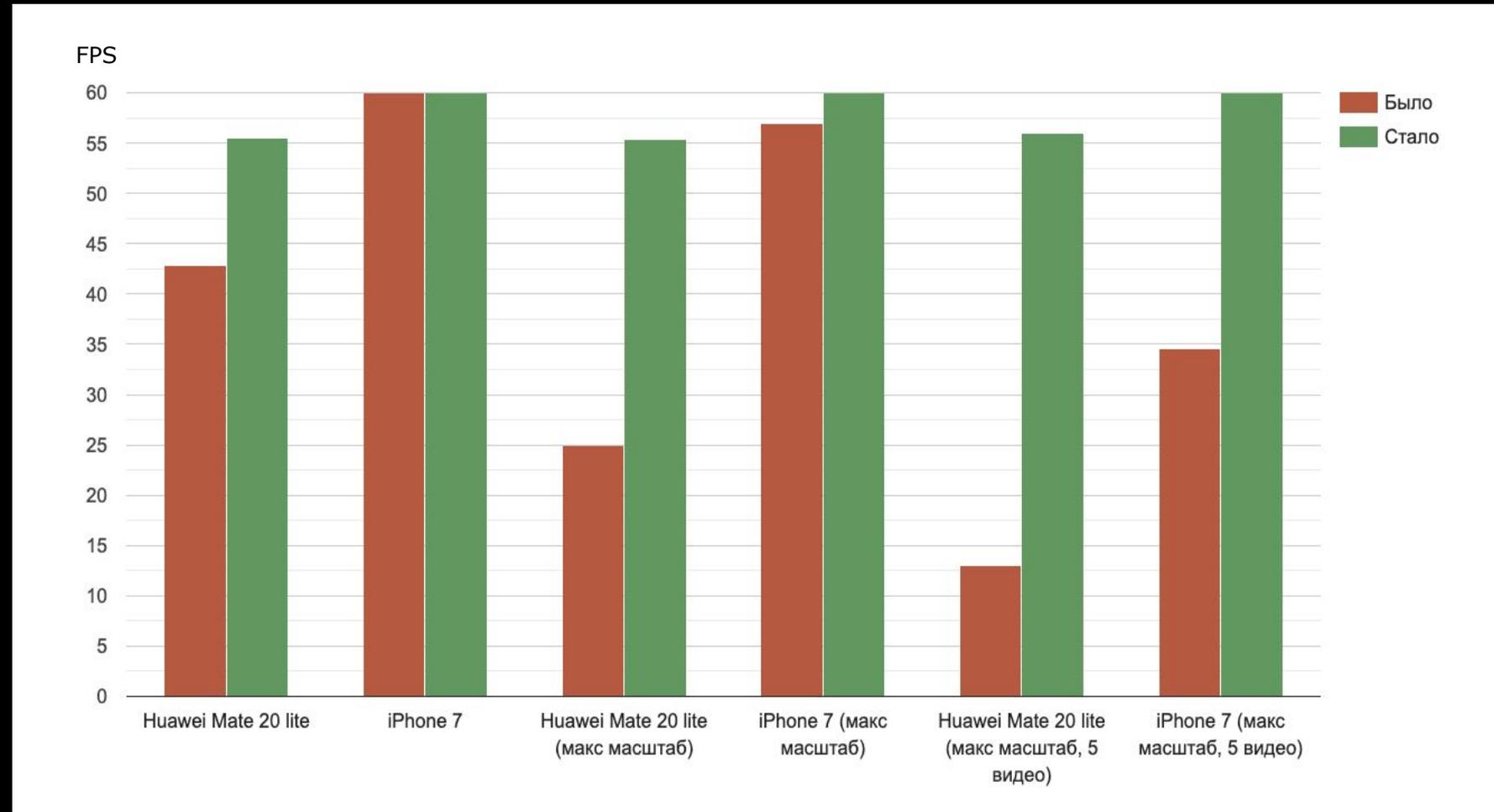
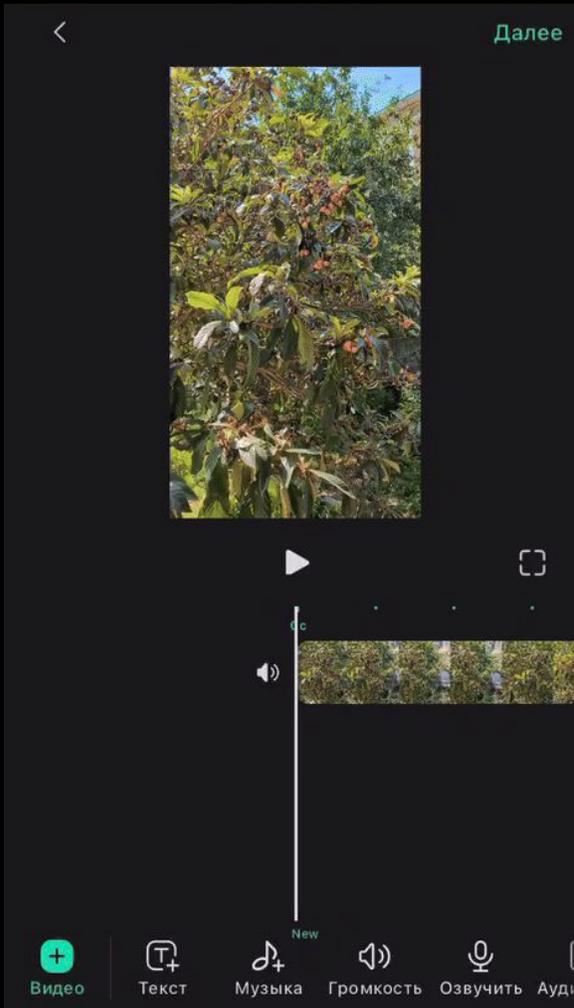
Средний FPS видео



Влияние на производительность приложения

Активное воспроизведение Средний FPS

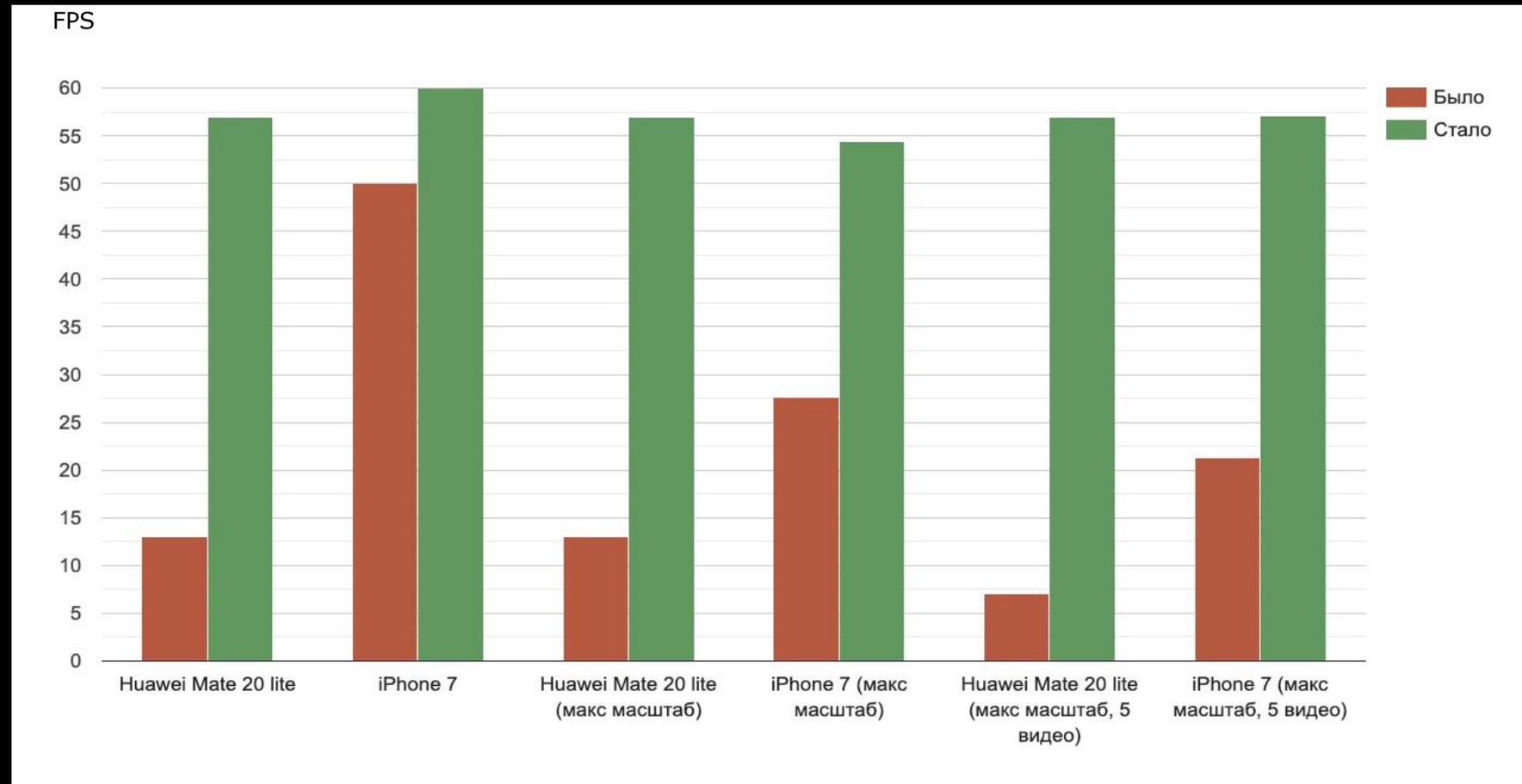
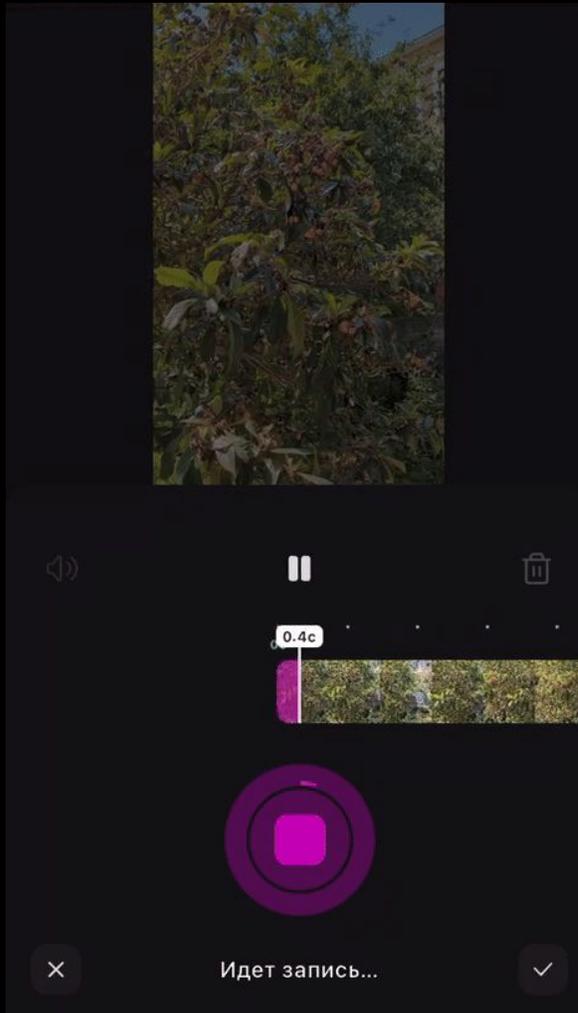
10 сек. видео



Влияние на производительность приложения

Активная запись голоса Средний FPS

10 сек. видео



Итого

- + 2D-скролл!
- + Готовые реализации от Flutter команды
- + Сумасшедший прирост производительности
- + Огромные возможности

Итого

- + 2D-скролл!
- + Готовые реализации от Flutter команды
- + Сумасшедший прирост производительности
- + Огромные возможности

- Сложна
- Нет готового механизма по типу Sliver для отображения вложенных списков
- Мало возможностей влиять на внутренние механизмы базового класса



Спасибо за внимание

