Прокрутка во Flutter

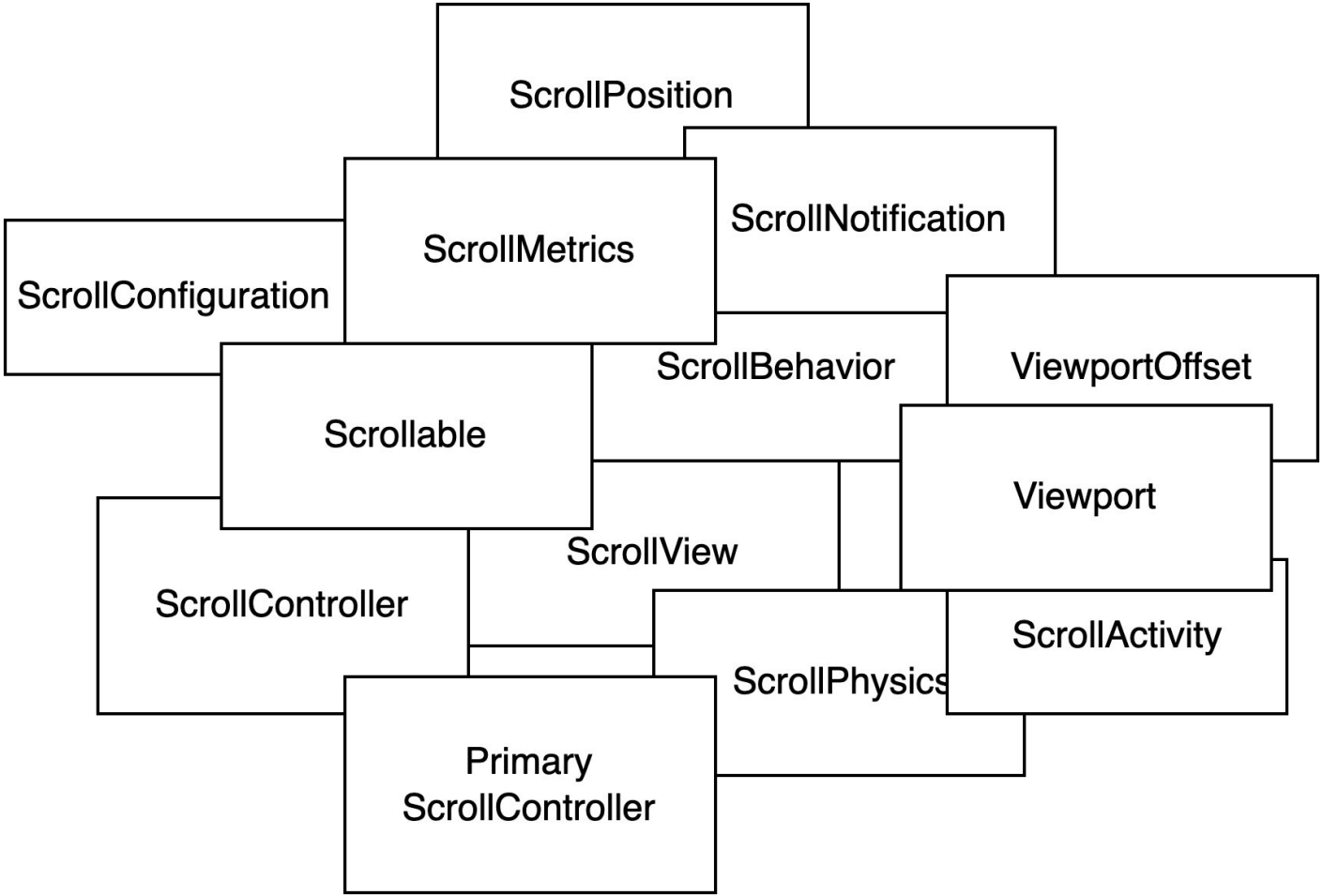
Меняем законы физики

О чем будем говорить?

Две основные части

1. Устройство скролла во Flutter;
2. Физика скролла

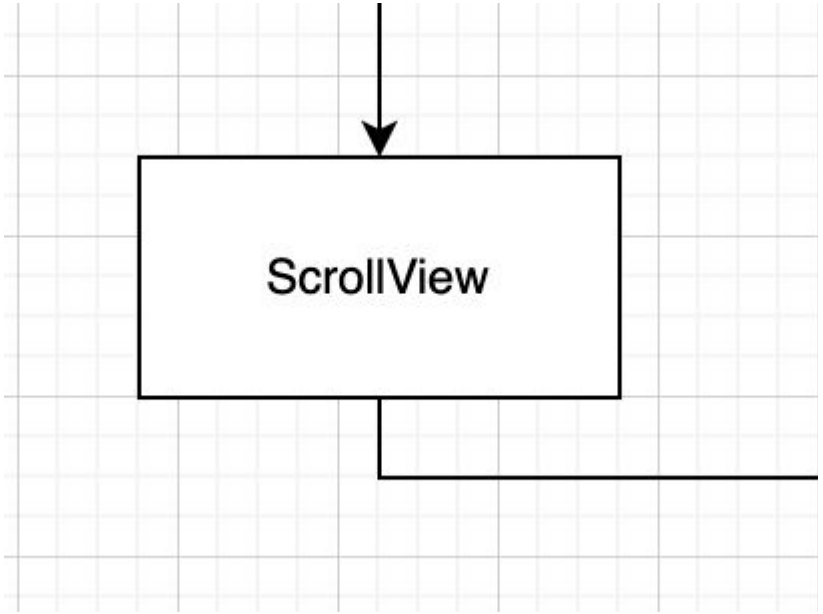
Let’s start


ScrollPosition
ScrollNotification
ScrollMetrics
ScrollConfiguration
ScrollBehavior
ViewportOffset
Scrollable
Viewport
ScrollView
ScrollController
ScrollActivity
ScrollPhysics
Primary
ScrollController

Начнем распутывать клубок

ListView/CustomScrollView

ScrollView



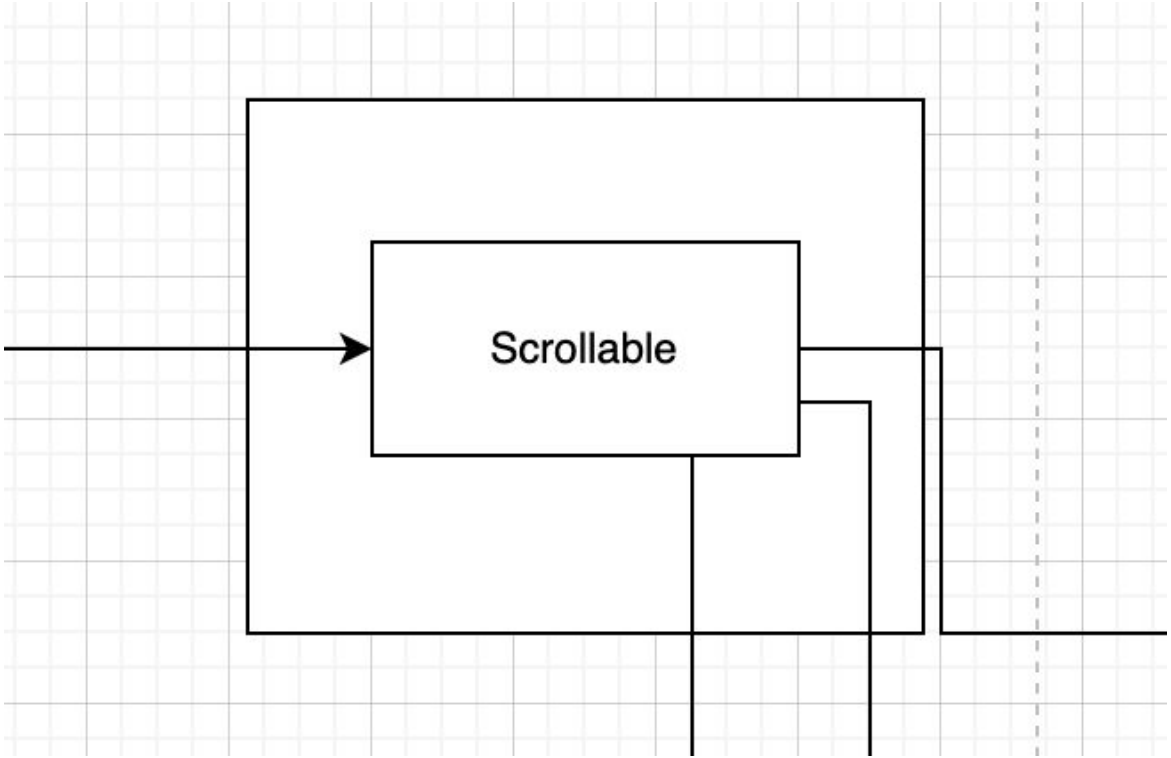
ScrollView - это лишь вершина айсберга

- По факту - лишь удобный интерфейс для стандартных списков;
- Не охватывает все “скроллы”;
- e.g: PageView не является ScrollView.

```
final Scrollable scrollable = Scrollable(
  dragStartBehavior: dragStartBehavior,
  axisDirection: axisDirection,
  controller: scrollController,
  physics: physics,
  scrollBehavior: scrollBehavior,
  semanticChildCount: semanticChildCount,
  restorationId: restorationId,
  viewportBuilder: (BuildContext context, ViewportOffset offset) {
    return buildViewport(context, offset, axisDirection, slivers);
  },
  clipBehavior: clipBehavior,
);
```

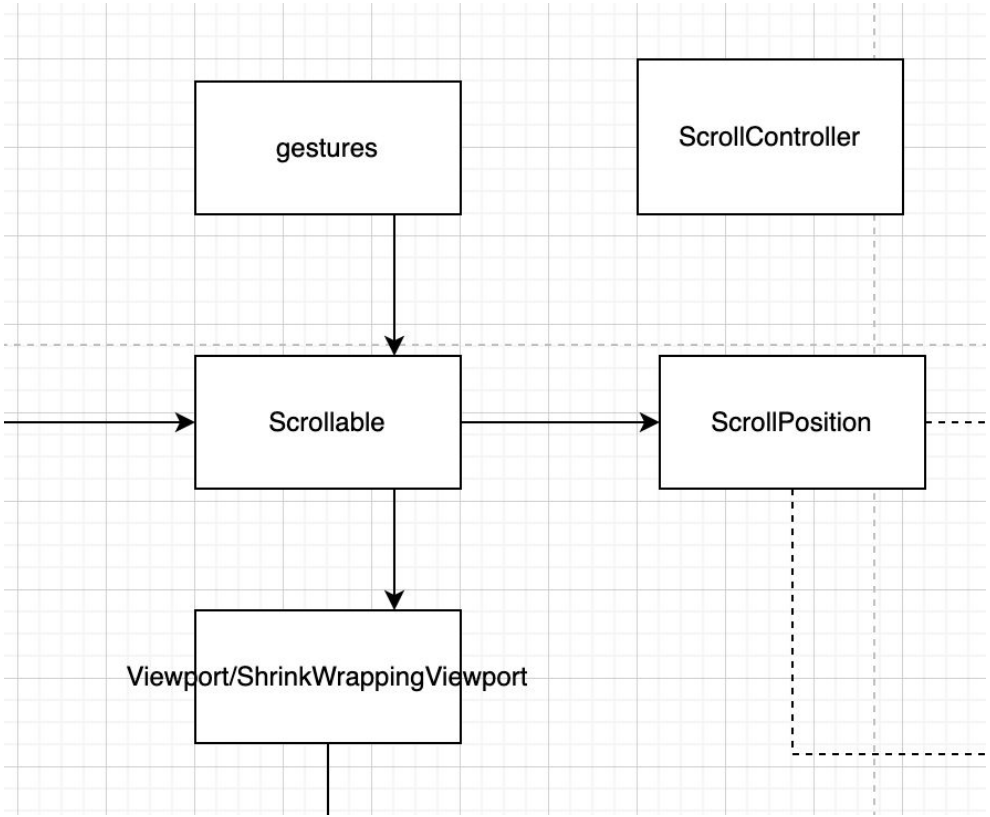
А кто тогда? Scrollable!

- объединяет все части скролла воедино
- “точка входа”



Кирпичики

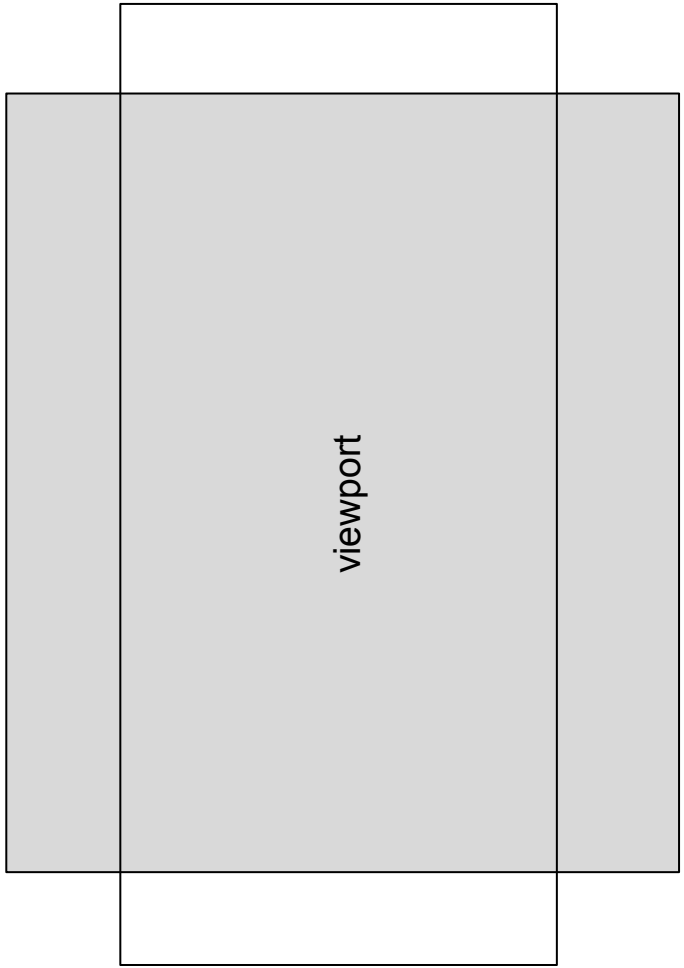
- GestureRecognizer
- Viewport
- ScrollPosition(ScrollController)



Слегка про вьюпорт

Та штука, о которой мы не будем говорить.

Осуществляет отображение данных. Имеет границы, слушает ScrollPosition(as viewportOffset).



ScrollController

“Пользовательский интерфейс” для управления позицией скролла

Умеет управлять несколькими ScrollPosition.

Создает ScrollPosition.

```
ScrollPosition createScrollPosition(
  ScrollPhysics physics,
  ScrollContext context,
  ScrollPosition? oldPosition,
) {
  return ScrollPositionWithSingleContext(
    physics: physics,
    context: context,
    initialPixels: initialScrollOffset,
    keepScrollOffset: keepScrollOffset,
    oldPosition: oldPosition,
    debugLabel: debugLabel,
  );
}
```

ScrollPosition - главный герой

- осуществляет расчет скролла
- хранит информацию метриках скролла(ScrollMetrics)
- оповещает об Viewport изменении
- расширяет ViewportOffset

Внутри ScrollPosition

- ScrollActivity - описывает некую операцию со скроллом
- ScrollPhysics - NB! - задает некоторые параметры, влияющие на скролл.

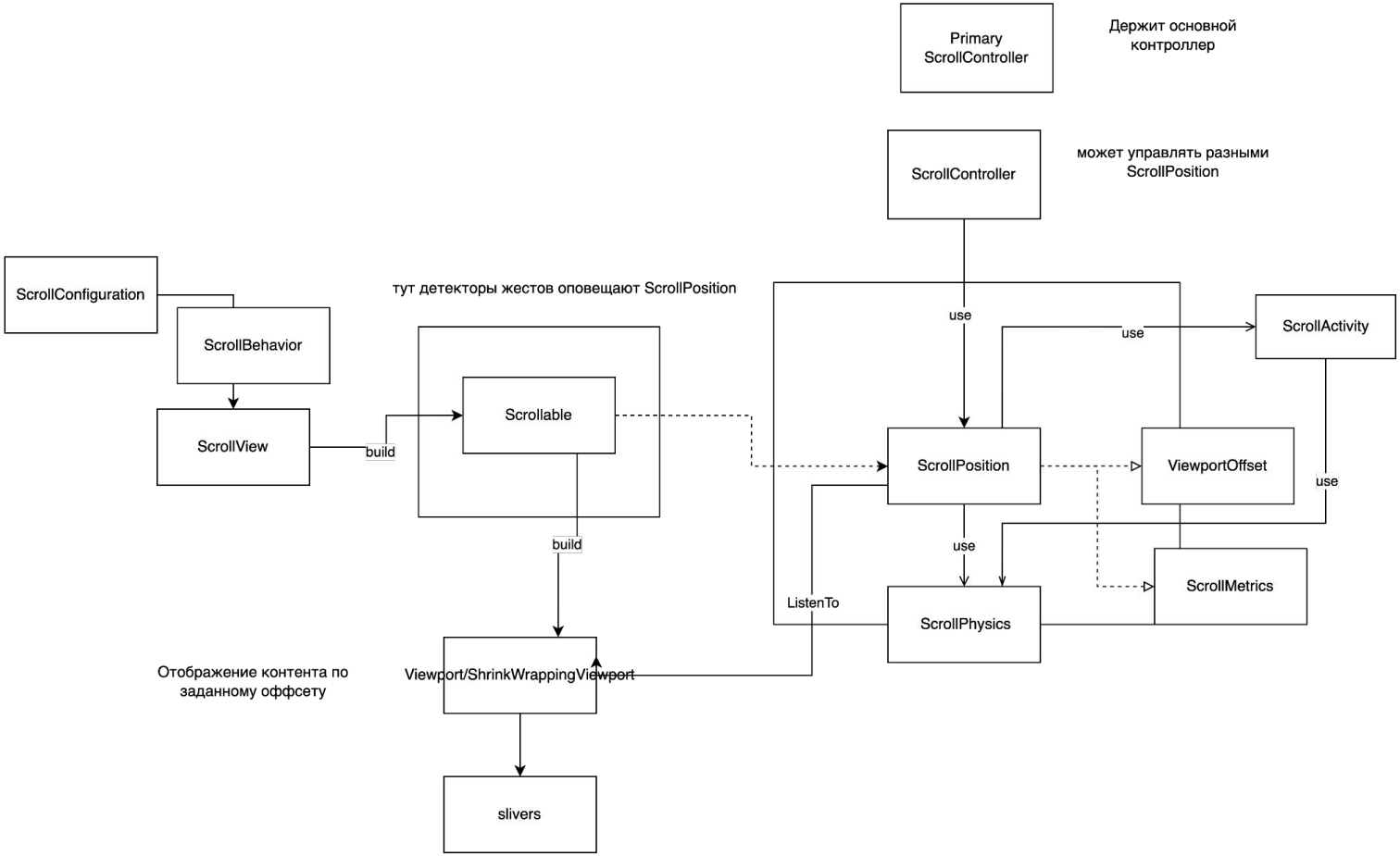
ScrollActivity

- DrivenScrollActivity - программные вызовы;
- DragScrollActivity - пользователь ведет палец;
- BallisticScrollActivity - происходит после того, как пользователь отпустил палец.

Что еще

ScrollConfiguration - дает возможность изменить поведение скролла у поддерева

ScrollBehavior - задает поведение скролла, используется в ScrollConfiguration

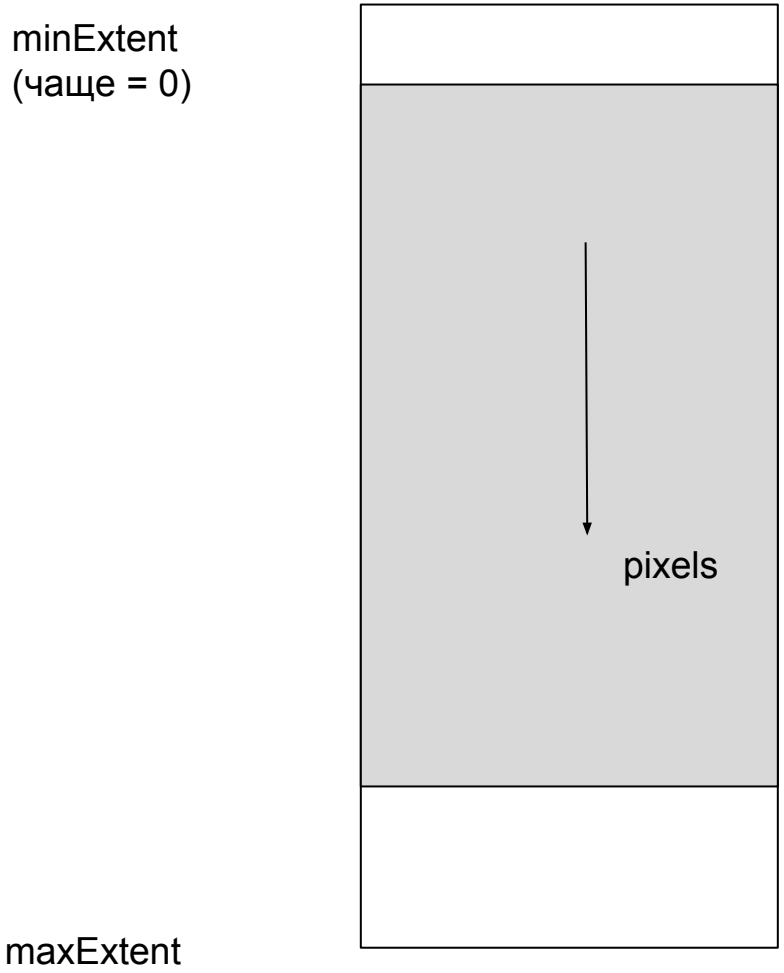

Primary ScrollController
Держит основной контроллер
ScrollController
может управлять разными ScrollPosition
ScrollConfiguration
ScrollBehavior
ScrollView
build
тут детекторы жестов оповещают ScrollPosition
Scrollable
build
use
use
ScrollActivity
ScrollPosition
ViewportOffset
use
use
ScrollMetrics
ListenTo
ScrollPhysics
Отображение контента по заданному оффсету
Viewport/ShrinkWrappingViewport
slivers

Физика скролла

Небольшой брифинг перед внедрением

ScrollMetrics, а именно они передаются в ScrollPhysics, имеют несколько важных полей:

- minScrollExtent
- maxScrollExtent
- pixels


minExtent
(чаще = 0)
pixels
maxExtent

Приложение для экспериментов

Нюанс #1

Мы не можем просто создать новый экземпляр нашей физики скролла с новым параметром и установить ее в ListView.

```
if (newPhysics?.runtimeType != oldPhysics?.runtimeType) {
  return true;
}
```

(Кратко) Некоторые геттеры

- spring - задает “пружинку” для стандартных симуляций
- tolerance - структура, определяющая точность
- minFlingDistance - минимальная дистанция для определения скролла
- minFlingVelocity , maxFlingVelocity - то же, но про скорость
- dragStartDistanceMotionThreshold - порог для определения начала перемещения(null default)
- allowImplicitScrolling - работает в случае неявного скролла ex TextField

Кратко

shouldAcceptUserOffset

Отвечает на вопрос: “Надо ли обрабатывать текущий оффсет скроллом?”

Типичный представитель: AlwaysScrollableScrollPhysics.

adjustPositionForNewDimensions

Обрабатывает кейс, когда список элементов увеличился.

Типичный представитель: RangeMaintainingScrollPhysics.

Кратко

carriedMomentum

Позволяет увеличить скорость скролла, если пользователь сделал несколько “флингов”.

“Как на айос”. По умолчанию только там и работает.

applyBoundaryConditions

Помогает вычислить факт оверскролла.

Также косвенно участвует в некоторых ScrollActivity.

Важно: надо следить за тем, что value находится не привышает дельту между прошлым и текущем значением скролла.


22:30
100%
DEBUG
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
Current value: 0.685275009255004
min: -1.0
max: 1.0

Нюанс #2

carriedMomentum используется для вычисления дополнительной скорости.

Но только, если он меньше некого значения.

```
double carriedMomentum(double existingVelocity)
```

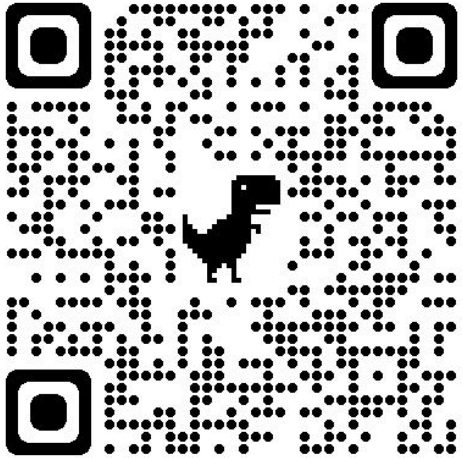
```
// Build momentum only if dragging in the same direction.
final bool isFlingingInSameDirection = velocity.sign == carriedVelocity!.sign;
// Build momentum only if the velocity of the last drag was not
// substantially lower than the carried momentum.
final bool isVelocityNotSubstantiallyLessThanCarriedMomentum =
  velocity.abs() > carriedVelocity!.abs() * momentumRetainVelocityThresholdFactor;
if(isFlingingInSameDirection && isVelocityNotSubstantiallyLessThanCarriedMomentum) {
  velocity += carriedVelocity!;
}
```

```
/// The minimum amount of velocity needed to apply the [carriedVelocity] at
/// the end of a drag. Expressed as a factor. For example with a
/// [carriedVelocity] of 2000, we will need a velocity of at least 1000 to
/// apply the [carriedVelocity] as well. If the velocity does not meet the
/// threshold, the [carriedVelocity] is lost. Decided by fair eyeballing
/// with the scroll_overlay platform test.
static const double momentumRetainVelocityThresholdFactor = 0.5;
```

applyPhysicsToUserOffset

Позволяет изменить расчет позиции скролла во время ведения пальцем.

```
double applyPhysicsToUserOffset(
    ScrollMetrics position,
    double offset,
)
```



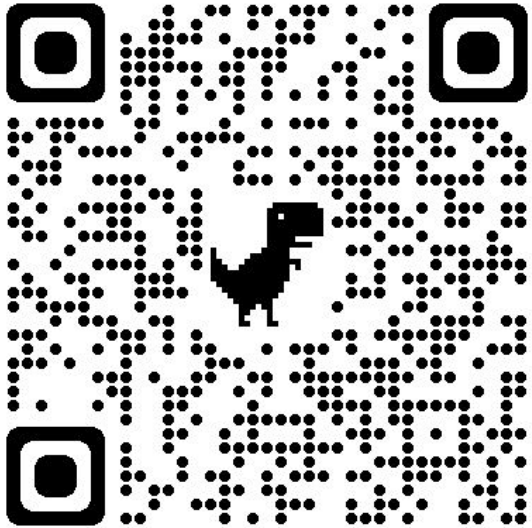

22:02
100%
DEBUG
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
Current value: 5.0
min: -100.0
max: 100.0


22:03
100%
DEBUG
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
Current value: -27.385133952763724
min: -100.0
max: 100.0

createBallisticSimulation

Задает поведение скролла после того, как мы отпустили палец.

```
Simulation? createBallisticSimulation(
  ScrollMetrics position, double velocity
)
```



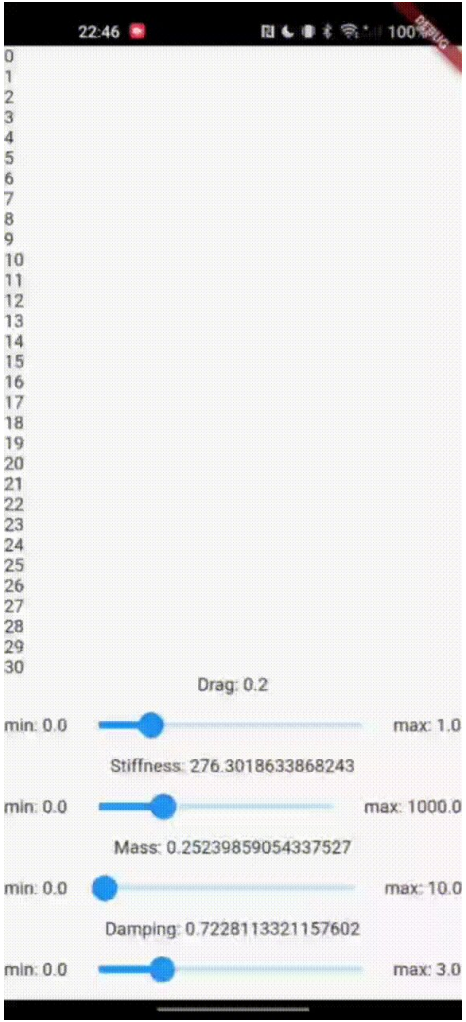
Simulation

Описывает некоторую линейную функцию на основе x, dx, time.

Например, есть GravitySimulation, FrictionSimulation etc.

Используется в анимациях(animateWith)

Чтобы правильно построить, придется вспомнить математику и физику.


22:46
100
DEBUG
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Drag: 0.2
min: 0.0
max: 1.0
Stiffness: 276.3018633868243
min: 0.0
max: 1000.0
Mass: 0.25239859054337527
min: 0.0
max: 10.0
Damping: 0.7228113321157602
min: 0.0
max: 3.0

Итак, для чего? (субъективно)

- Изменить направление скролла.
- Изменить реакцию скролла после того, как пользователь убрал палец.
- Изменить характеристики, связанные с началом скролла или “флинга”.

that’s all folks

Бонусы

app

tester from flutter team