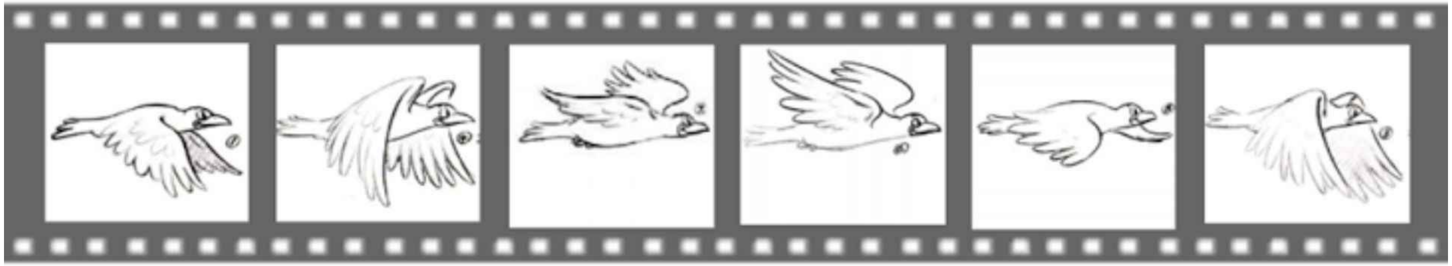
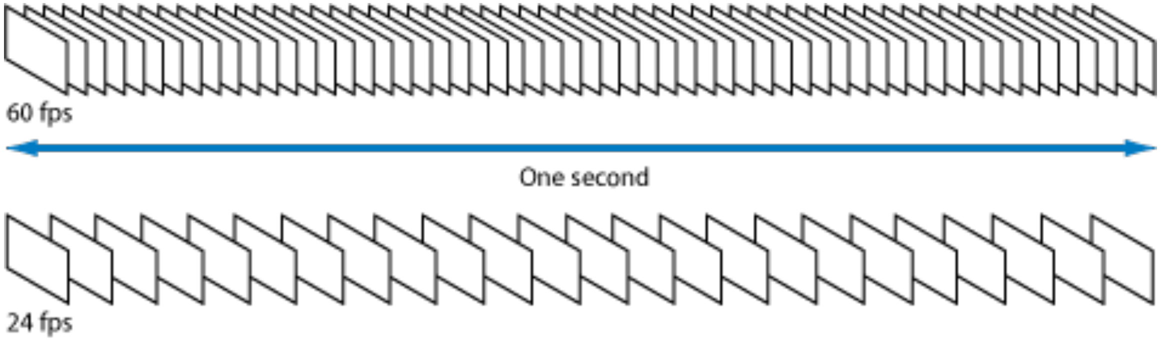
Animations

Shan-Hung Wu
CS, NTHU

Animation

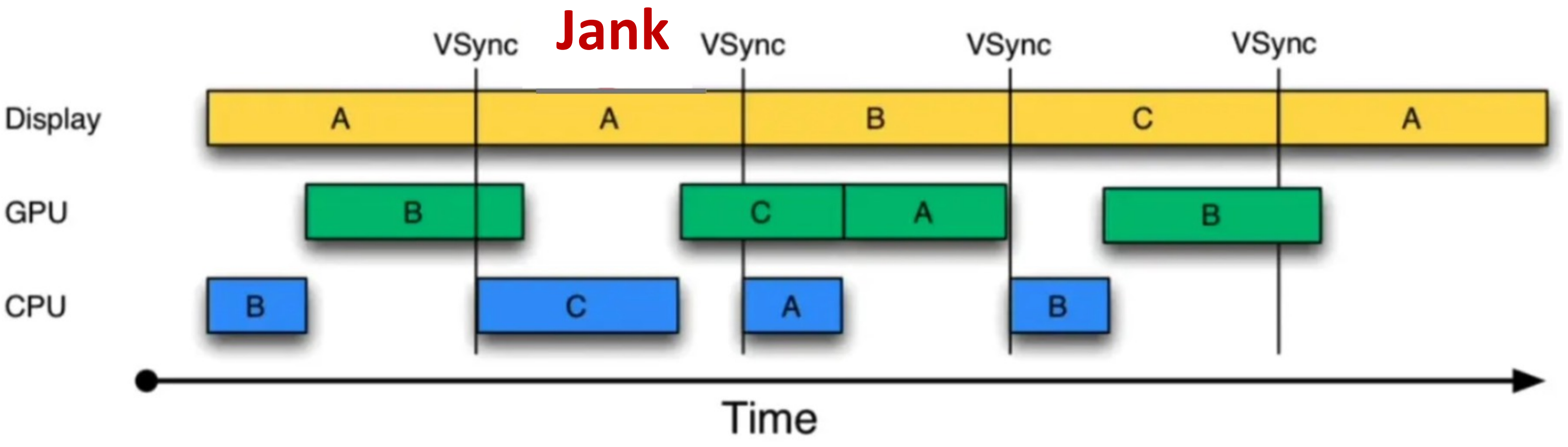




- Illusion of successive frames played at high speed

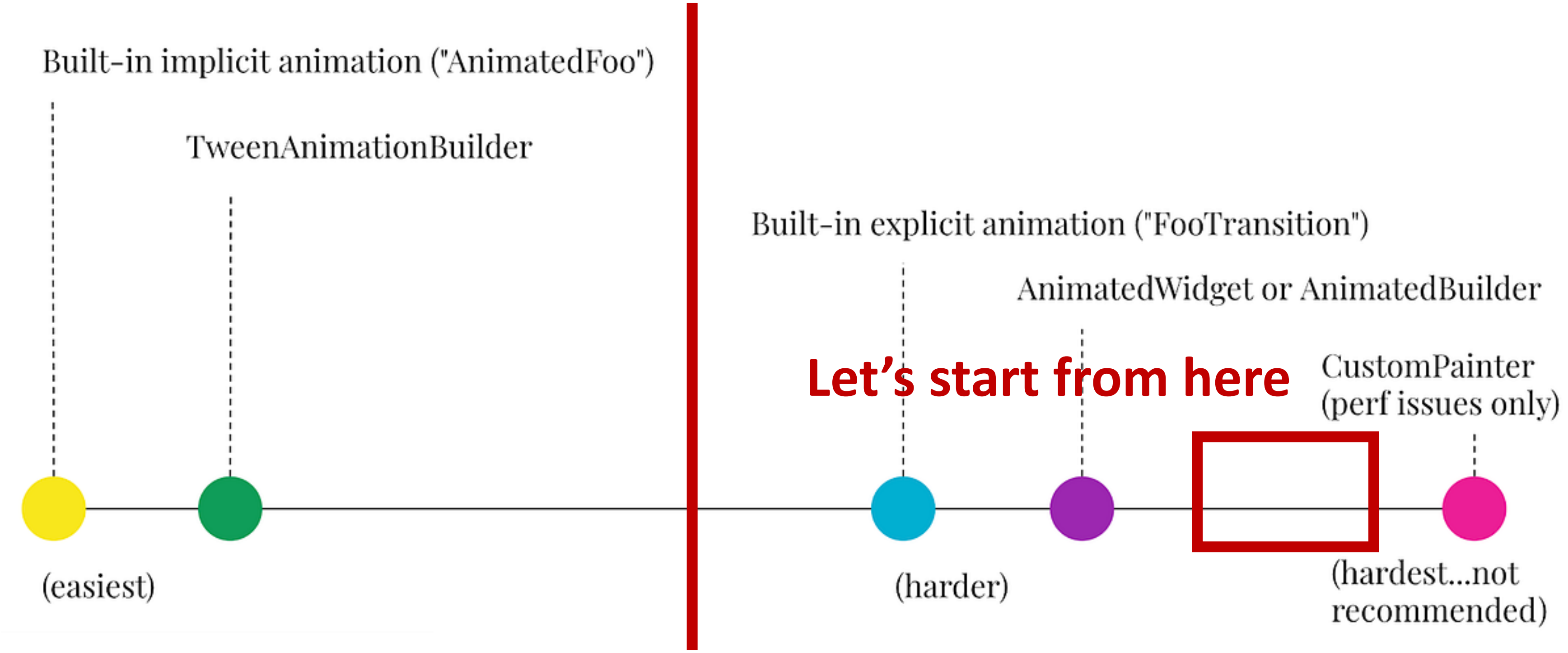
Animation in Flutter

- A quick succession of calling `setState()` in `StatefulWidget`
- Who calls `setSate()` at target fps?
- ***Ticker*** a.k.a. `vsync`


Jank

Explicit vs. Implicit Animations

- ***Explicit animation***: you create ticker explicitly
- ***Implicit animation***: you do not


Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not recommended)

Let's start from here

Demo 1

- Explicit “progress” animation using **`Ticker`**

- Your responsibilities:
- Calculate the animated value (state) during animation
- Manage ticker’s lifecycle
- Call `setState()` to update UI
- Minimize rebuild/layout costs of widget tree

Demo 2

- Explicit “progress” animation using **`AnimationController`**

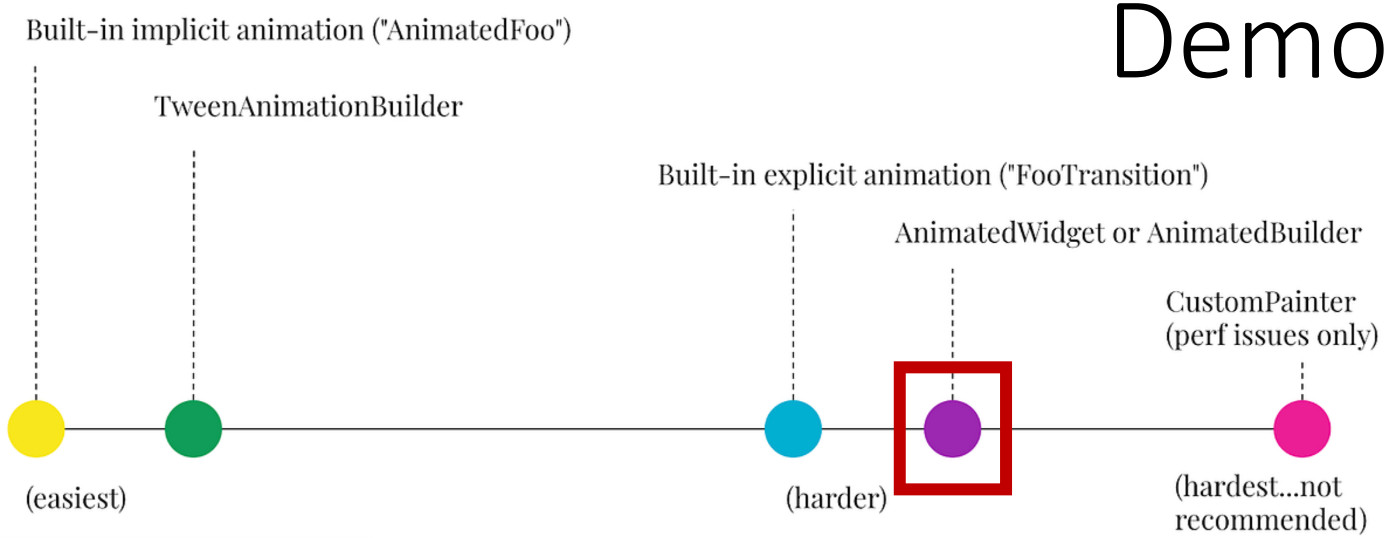
- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
- Manage ~~ticker’s~~ controller’s lifecycle
- ~~Call `setState()` to update UI~~
- Minimize rebuild/layout costs of widget tree

Demo 3

- Explicit “slide” animation using `AnimationController`

- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
  - Use **`Tween`** and **`Curves`** to smoothen animation
- Manage ~~ticker’s~~ controller’s lifecycle
- ~~Call `setState()` to update UI~~
- Minimize rebuild/layout costs of widget tree

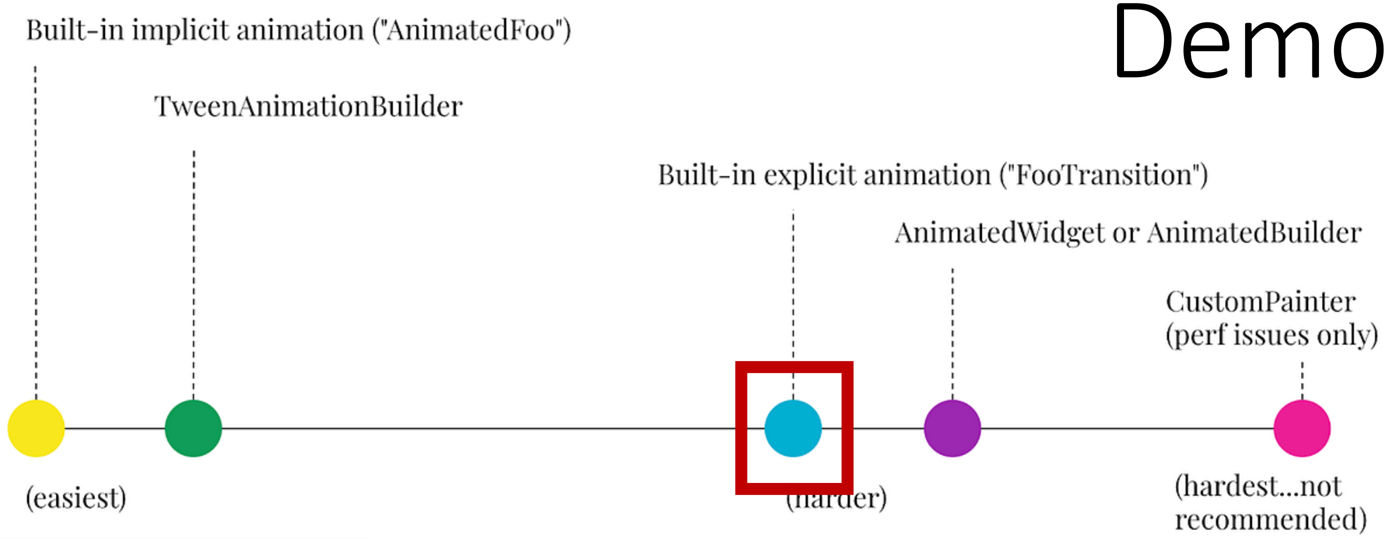
Demo 4



Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not
recommended)

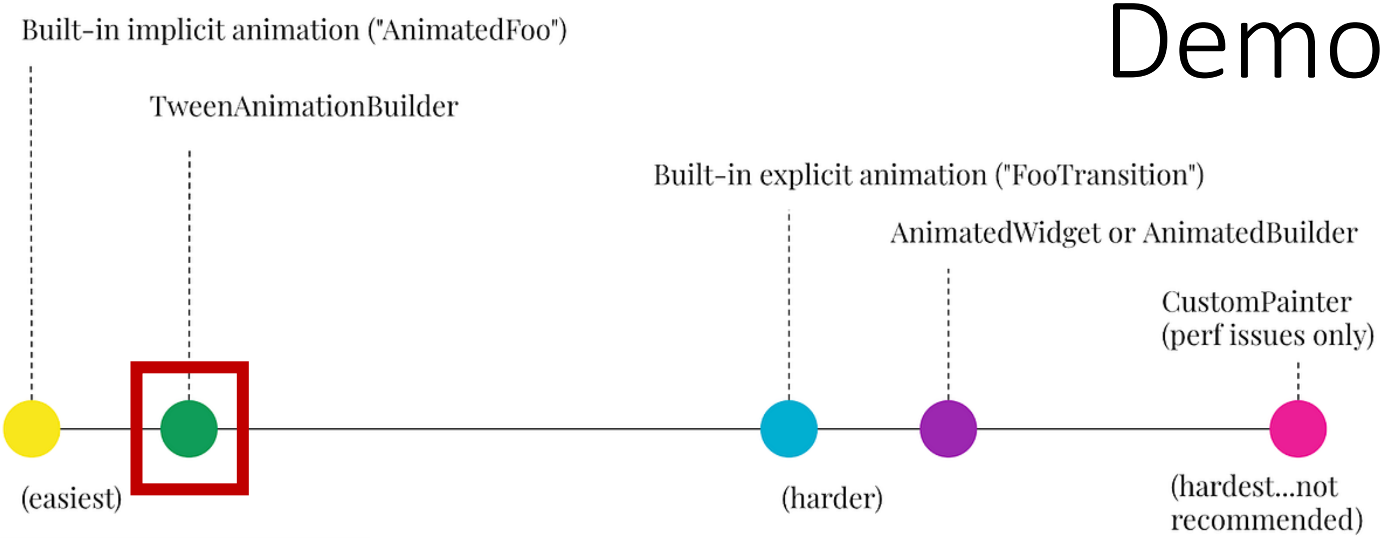
- Explicit “slide” animation using **`AnimatedBuilder`**
- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
  - Use `Tween` and `Curves` to smoothen animation
- Manage ~~ticker’s~~ controller’s lifecycle
- ~~Call `setState()` to update UI~~
- Minimize rebuild/layout costs of widget tree
  - Specify **`child`** that won’t change

Demo 5


Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not
recommended)

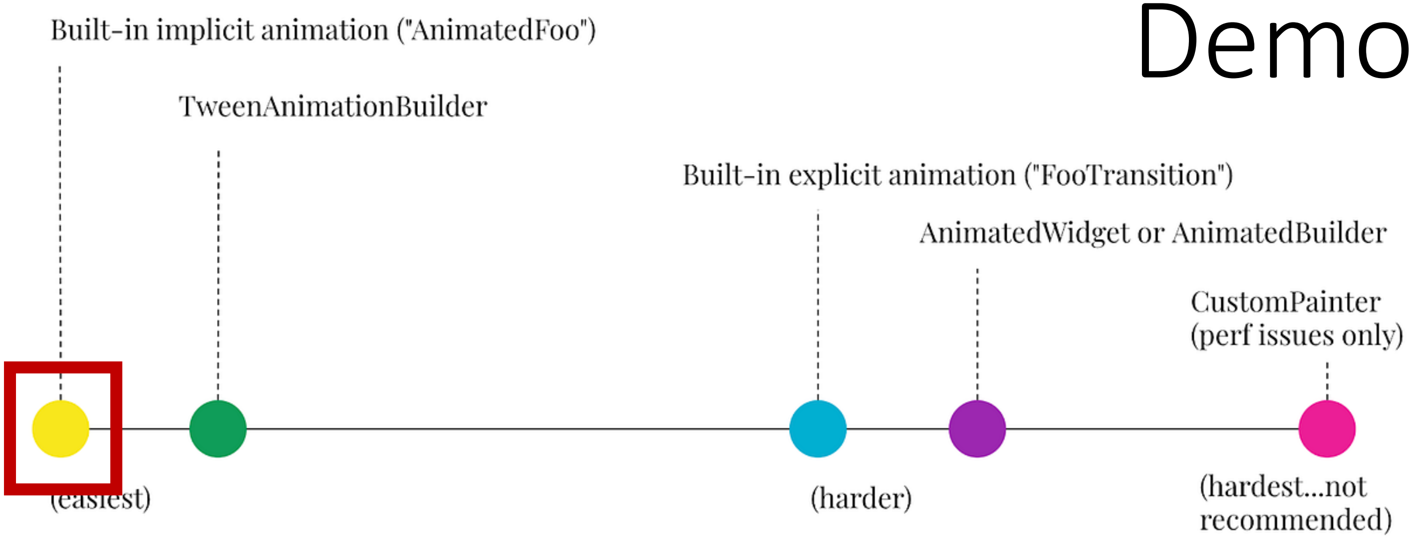
- Explicit "slide" animation using **SlideTransition**
- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
  - Use `Tween` and `Curves` to smoothen animation
- Manage ~~ticker's~~ controller's lifecycle
- ~~Call `setState()` to update UI~~
- ~~Minimize rebuild/layout costs of widget tree~~
  - Specify `child` that won't change

Demo 6


Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not
recommended)

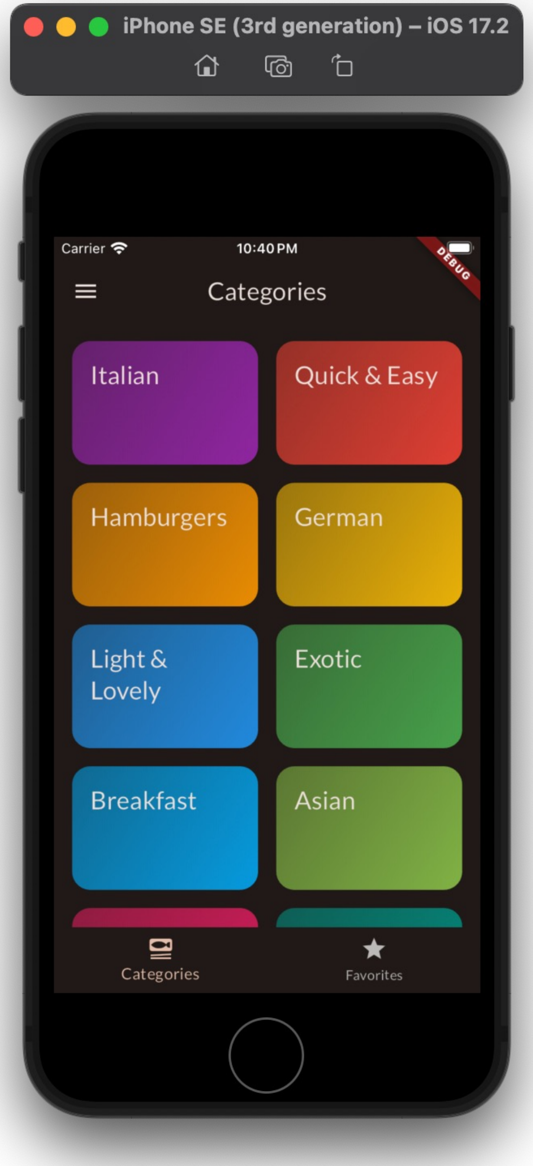
- Implicit “slide” animation using **`TweenAnimationBuilder`**
- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
  - Use `Tween` and `Curves` to smoothen animation
  - Limitation: ***less control*** (via `end` only) over animation state
- ~~Manage controller’s lifecycle~~
- ~~Call `setState()` to update UI~~
- ~~Minimize rebuild/layout costs of widget tree~~
  - Specify `child` that won’t change

Demo 7


Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not
recommended)

- Implicit "slide" animation using **`AnimatedSlide`**
- Your responsibilities:
- ~~Calculate the animated value (state) during animation~~
  - Use `Tween` and `Curves` to smoothen animation
  - Limitation: ***no control*** over animation state
- ~~Manage controller's lifecycle~~
- ~~Call `setState()` to update UI~~
- ~~Minimize rebuild/layout costs of widget tree~~
  - Specify `child` that won't change

Animations in Meals App



- Explicit slide-in animation for categories
- Implicit `AnimatedSwitcher` for toggling favorite
- Animated page transitions

Implicit `AnimatedSwitcher`

- Animates “switching” of changing child

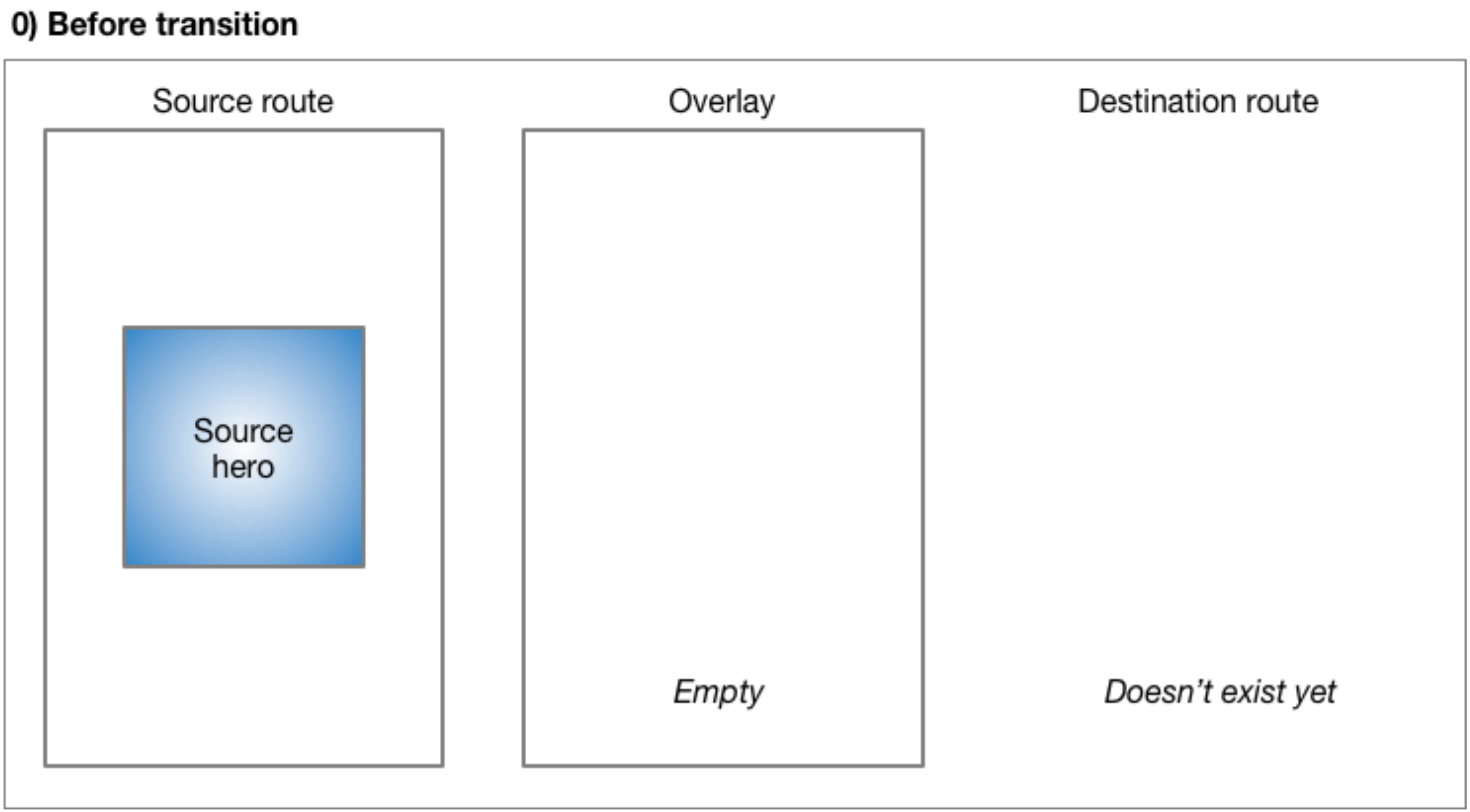
- Use `key` to help flutter detect changes
- `transitionBuilder` prop applies to ***both*** the entering and exiting child widgets
- But in ***opposite*** directions
  - E.g., one fade in, another fade out
- To rotate both widgets in same direction, we need to determine if `child` is entering or exiting
- Then, apply `ReverseAnimation()` to one of them

Animating Page Transitions

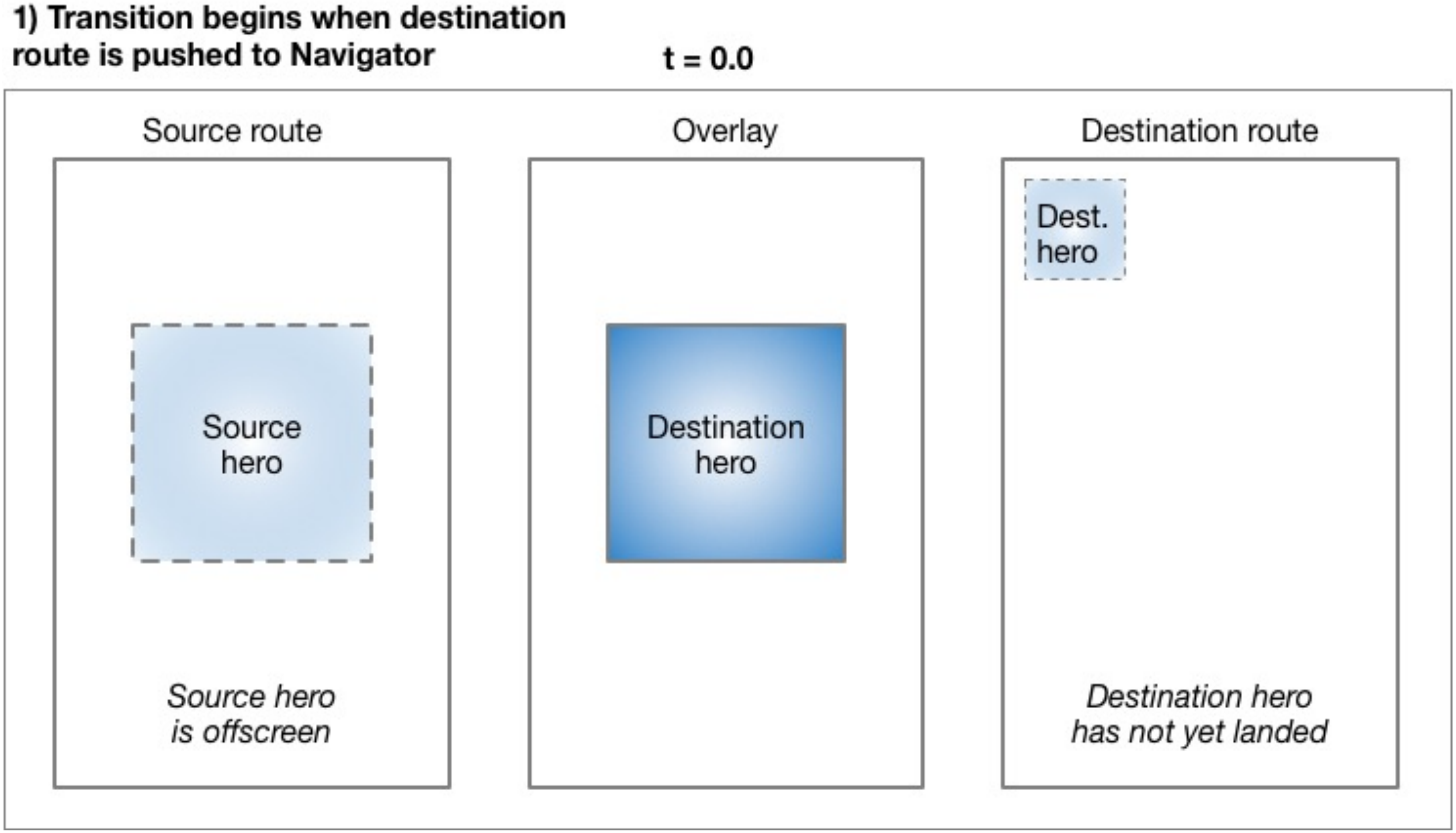
- No page transitions across `HomePage` tabs
  - Use `pageBuilder` prop to specify transition animation in route config

- Hero animation from `MealsPage` to `MealDetailsPage`

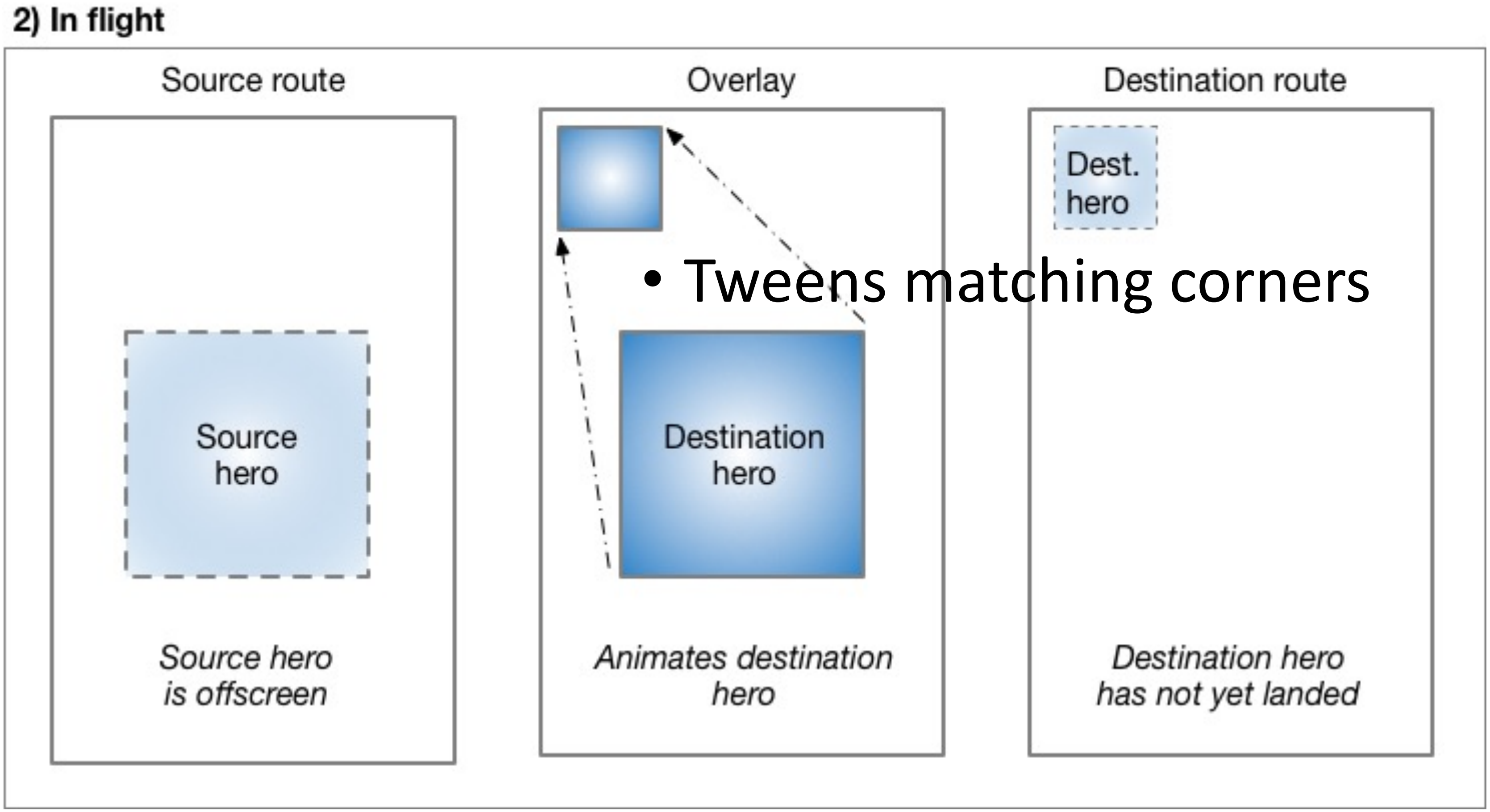
Hero Page Transition



Hero Page Transition

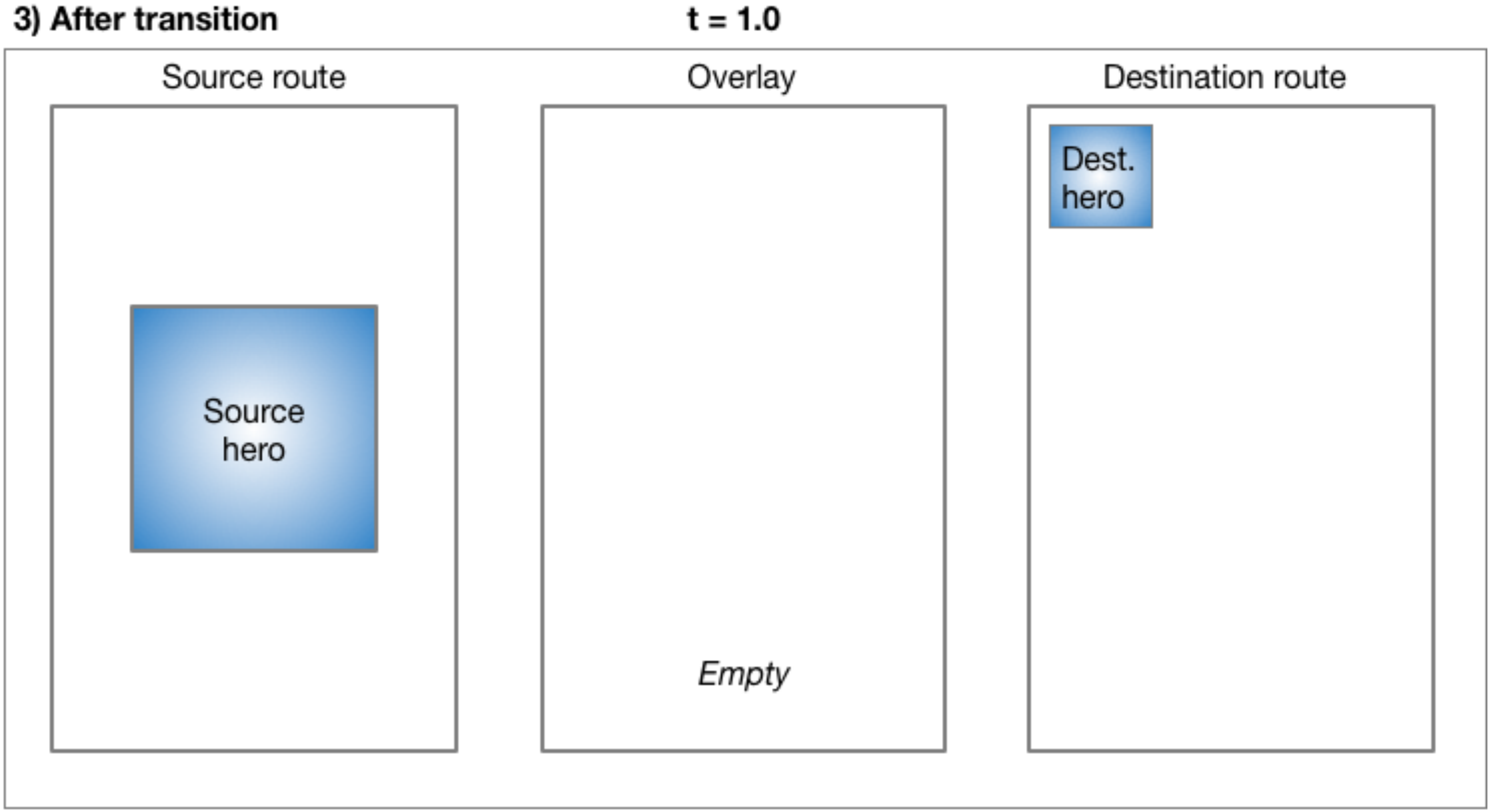


Hero Page Transition



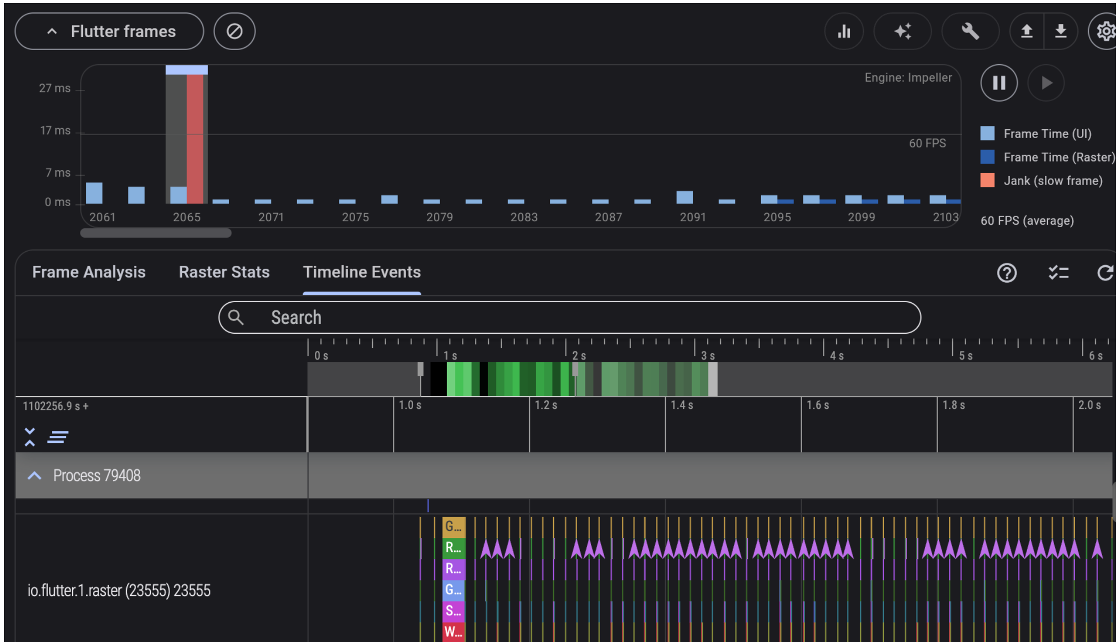
- Tweens matching corners

Hero Page Transition



Remarks

- Add animations only ***after*** your app has functional values
- Use **`Performance`** tab in Dev Tool to identify bottlenecks



Assigned Readings

- Radial hero animations

- Staggered animations
  - Example: staggered menu