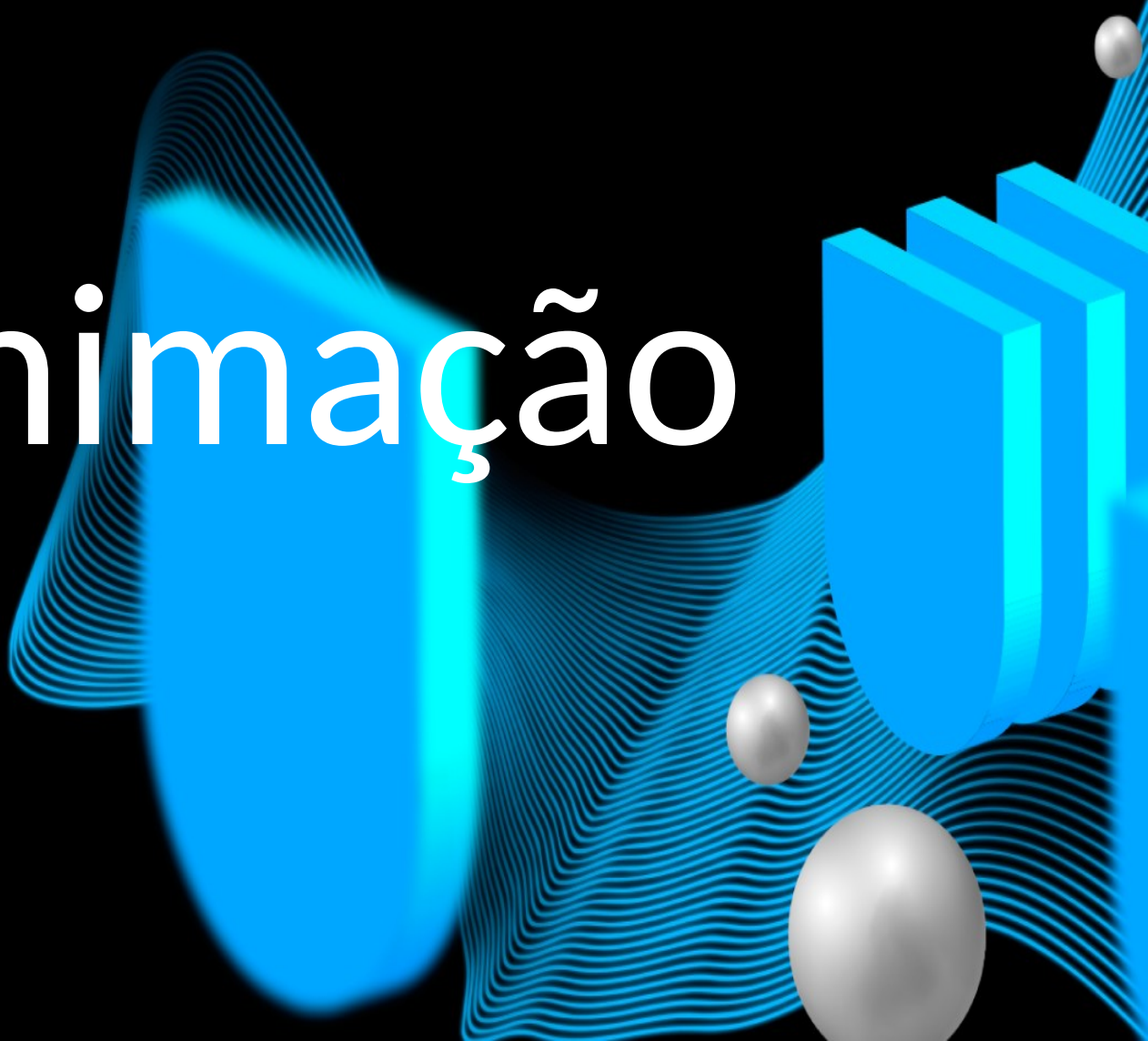

UNITAU
Universidade de Taubaté

Animação



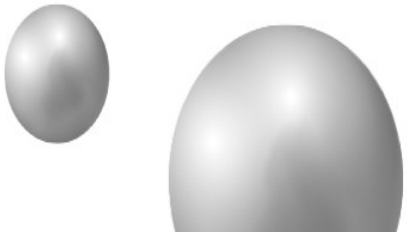
Animação

Animações são feitas recriando uma parte da Widget Tree a uma taxa de 60 a 120 frames por segundo. Para isso, é necessário alterar o estado de algum widget nessa taxa.

Isso é feito usando um **Ticker** e um **AnimationController**.

O **Ticker** é um objeto que chama uma função para cada frame da animação.

O **AnimationController** é acionado pelo **Ticker** gera um valor normalizado entre 0 e 1 para determinar o passo da animação e aciona a mudança de estado e consequentemente a reconstrução da UI para representar esse passo da animação.



Animação

Crie uma nova aplicação Flutter com o nome **animation_controller**.

Altere o arquivo **main.dart**:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue, ),
      home: Home(),
    );
  }
}
```

Animação

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home>
  with SingleTickerProviderStateMixin<Home> {
  AnimationController _controller;
  int i = 0;
```

Animação

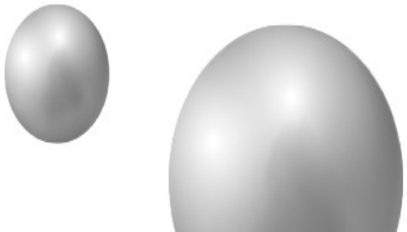
```
@override
void initState() {
  super.initState();
  _controller = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 3),
  );
  _controller.addListener(_update);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("Controller"),),
    body:  Center(
      child: Text(
        '$i m/s',
        style: Theme.of(context).textTheme.display1,
      ),
    ),
```

Animação

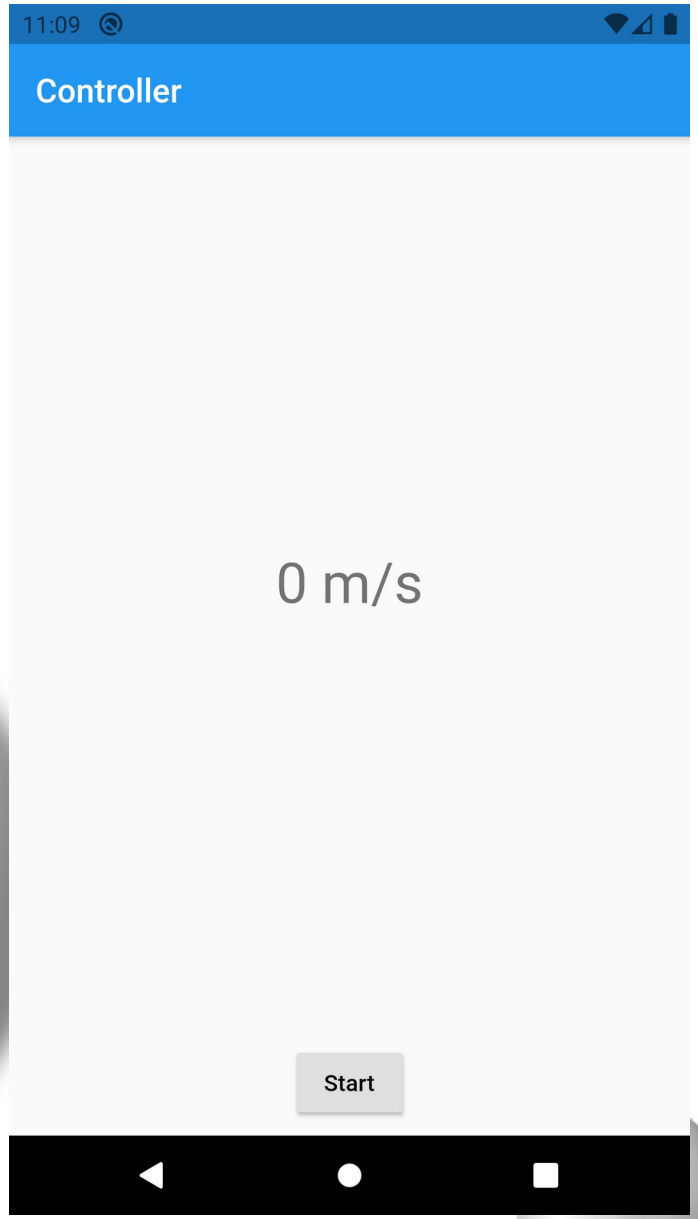
```
    bottomNavigationBar: ButtonBar(
      alignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        RaisedButton(
          child: Text('Start'),
          onPressed: () => setState(() {
            if (i==0)
              _controller.forward();
            else
              _controller.reverse();
          }),
        ),
      ],
    ),
  );
}
```

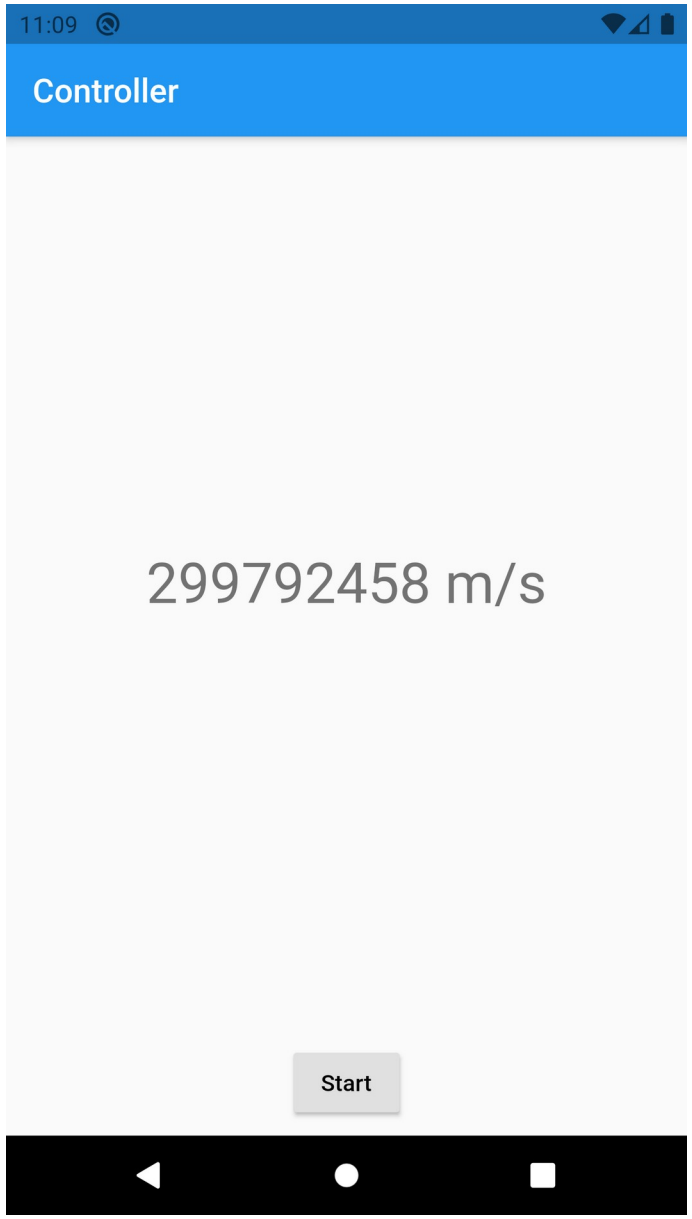
Animação

```
  void _update() {
    setState(() {
      i = (_controller.value * 299792458).round();
    });
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
}
```



Animação


11:09
Controller
0 m/s
Start


11:09
Controller
299792458 m/s
Start

Animação

Flutter disponibiliza diversos mecanismos para criar animações mais facilmente, utilizando o **Ticker** e o **AnimationController** para criar os frames que representar os passos da animação.

É recomendado o uso desses mecanismos ao invés de utilizar diretamente o **Ticker** e o **AnimationController** para gerenciar a mudança de estado da animação.

Esses mecanismos são divididos em animações implícitas e animações explícitas.

Animação

As animações implícitas utilizam widgets que embutem o ticker e o controller para gerenciar o andamento da animação. Não é necessário administrar diretamente o controller:

- Widgets com animação implícita
- Tween Animation

As animações explícitas utilizam widgets que não possuem um ticker e controller embutidos. É necessário criar um ticker e um controller e administrar diretamente o controller para gerenciar a animação:

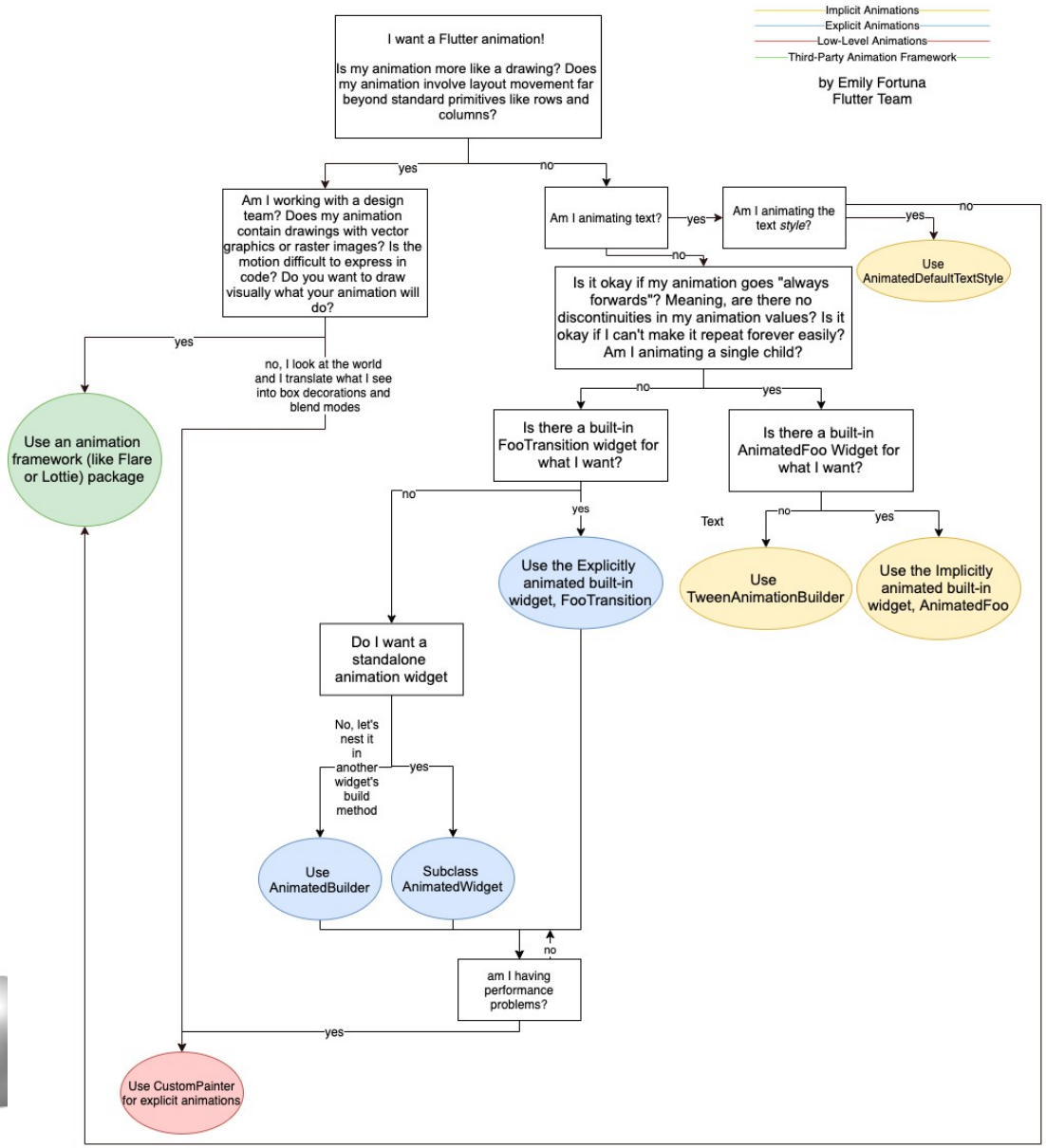
- Transições
- AnimatedBuilder
- AnimatedWidget

Animação

Além dos recursos nativos do Flutter para animações, também estão disponíveis diversos plugins disponibilizados pela comunidade para animações.

Na documentação do Flutter podemos encontrar sugestões sobre como decidir qual tipo de animação é mais adequado para cada situação.

Animação



https://flutter.dev/docs/development/ui/animations

Widgets com Animação Implícita

Para criar animações básicas sem repetição infinita e aplicada a apenas um widget ou uma parte da árvore de widgets, o Flutter disponibiliza diversos widgets que tem animação implícita de parâmetros diversos como **AnimatedContainer**, **AnimatedList**, **AnimatedPositioned**, **AnimatedDefaultTextStyle** e vários outros.

Widgets com Animação Implícita

Crie uma nova aplicação Flutter com o nome **animation_container**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório onde serão armazenadas as imagens:

```
assets:
  - assets/images/
```

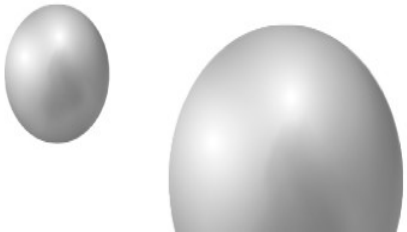
Crie o subdiretório **assets/images**.

Copie o arquivo **star.png** no subdiretório **assets/images**.

Widgets com Animação Implícita

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```



Widgets com Animação Implícita

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  int _duration = 1;
  bool _big = false;
  List<String> _curveList = [
    'linear',
    'easeInOutBack',
    'slowMiddle',
    'bounceIn'
  ];
  String _curveName = "linear";
```

Widgets com Animação Implícita

```
@override
Widget build(BuildContext context) {
  Curve _curve = Curves.linear;
  if(_curveName == 'easeInOutBack')
    _curve = Curves.easeInOutBack;
  if(_curveName == 'slowMiddle')
    _curve = Curves.slowMiddle;
  if(_curveName == 'bounceIn')
    _curve = Curves.bounceIn;
  return Scaffold(
    appBar: AppBar(title: Text("AnimatedContainer"),),
```

Widgets com Animação Implícita

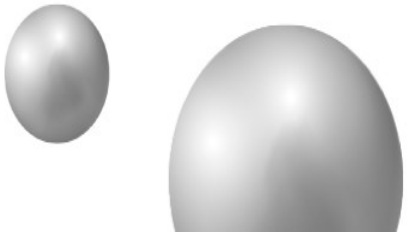
```
      body: Column(
        children: <Widget>[
          TextField(
            controller: TextEditingController(text: "$
{_duration}"),
            decoration: InputDecoration(labelText:
"Duration"),
            keyboardType: TextInputType.number,
            inputFormatters:
              [WhitelistingTextInputFormatter.digitsOnly],
            onSubmitted: (String value) => _duration =
int.parse(value),
          ),
```

Widgets com Animação Implícita

```
Row(
  children: <Widget>[
    Text("Curve:  "),
    DropdownButton<String>(
      value: _curveName,
      icon: Icon(Icons.arrow_downward),
      iconSize: 24,
      elevation: 16,
      underline: Container( height: 2, ),
      onChanged: (String value) {
        setState(() =>_curveName = value);
      },
      items:
_curveList.map<DropdownMenuItem<String>>((String value) {
        return DropdownMenuItem<String>(
          value: value,
          child: Text(value),
        );
      }).toList(),
    ),
  ],
),
```

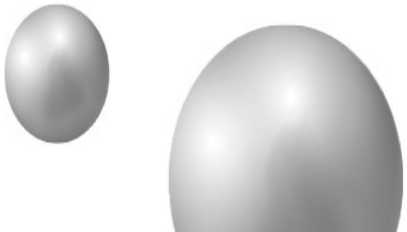
Widgets com Animação Implícita

```
            Expanded(
              child: Center(
                child: AnimatedContainer(
                  decoration: BoxDecoration(
                    gradient: RadialGradient(
                      colors: [Colors.lightGreenAccent[400],
Colors.transparent],
                      stops: [ !_big ? 0.0 : 0.7, 1.0]
                    )
                  ),
                  width: _big ? 500 : 100,
                  child:
Image.asset('assets/images/star.png'),
                  duration: Duration(seconds: _duration),
                  curve: _curve,
                ),
              ),
            ),
```

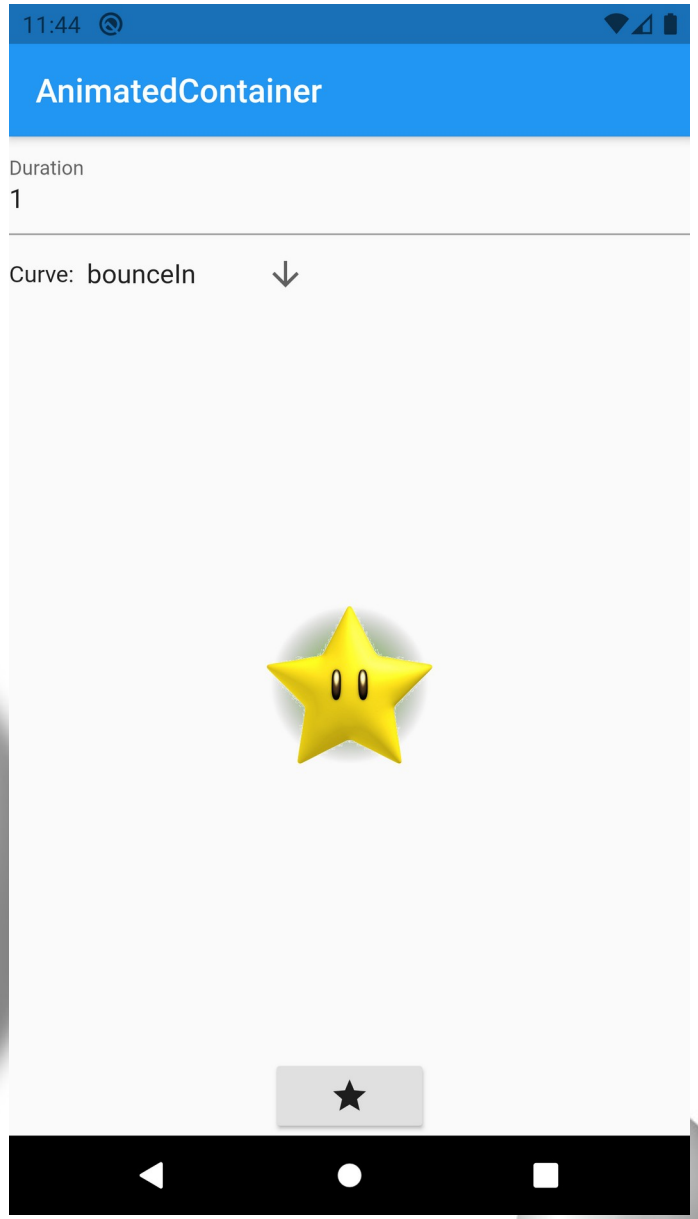


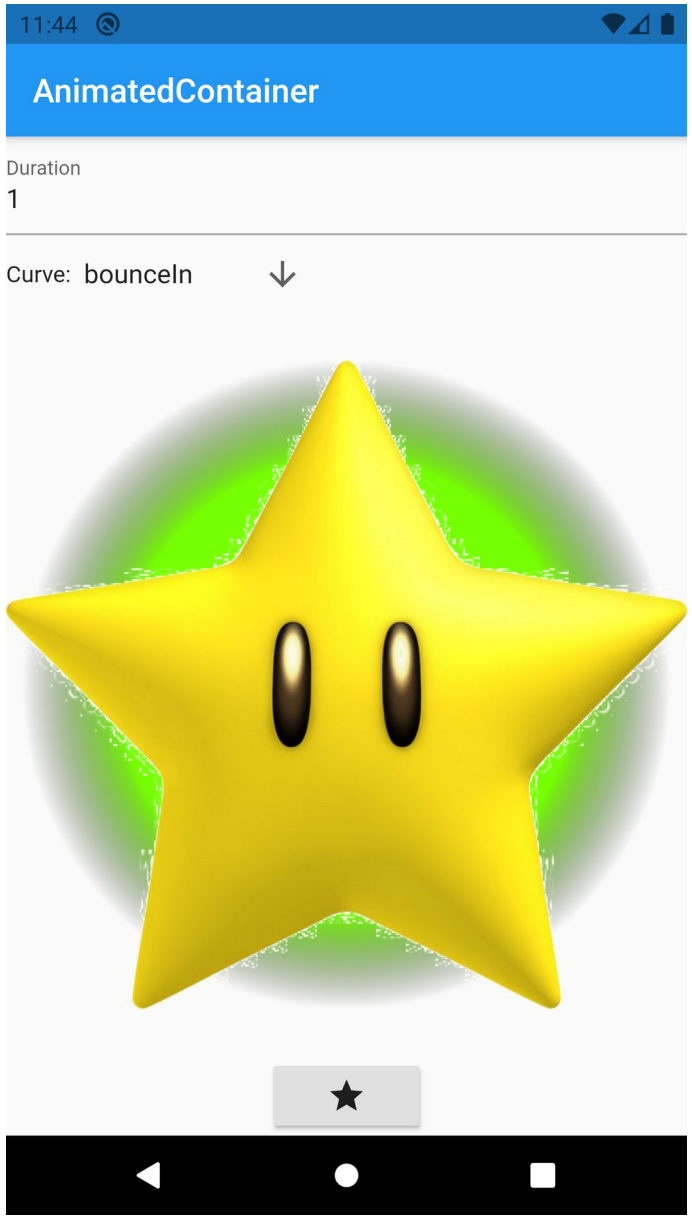
Widgets com Animação Implícita

```
            RaisedButton(
              onPressed: () => setState(() {
                _big = !_big;
              }),
              child: Icon(Icons.star),
            ),
          ],
        ),
      );
    }
}
```



Widgets com Animação Implícita





Tween Animation

Para criar animações básicas que não se encaixam nos widgets com animação implícita já existentes, o **TweenAnimationBuilder** permite criar animações delimitas dentro de um range (between) de parâmetros.

Tween Animation

Crie uma nova aplicação Flutter com o nome **animation_tween**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório onde serão armazenadas as imagens:

```
assets:
  - assets/images/
```

Crie o subdiretório **assets/images**.

Copie os arquivos **stars.jpg** e **sun.png** no subdiretório **assets/images**.

Tween Animation

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Tween Animation

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  Color _newColor = Colors.red;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Tween Animation"),),
```

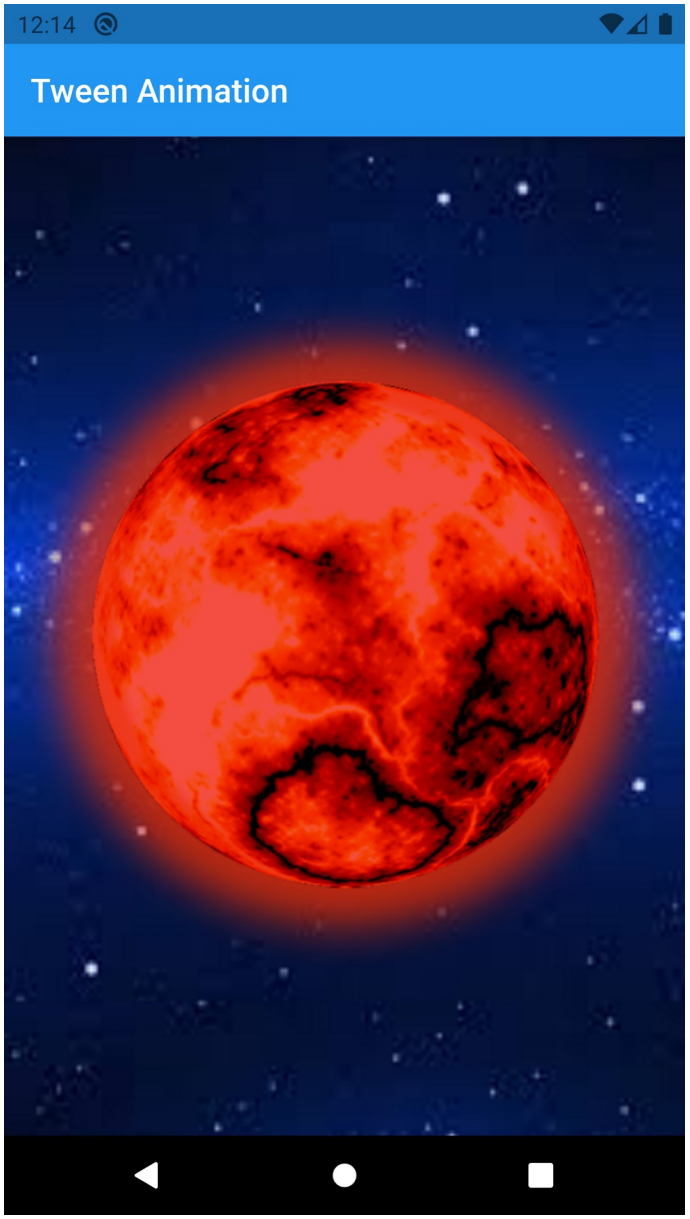
Tween Animation

```
      body: Stack(
        children: <Widget>[
          Image.asset(
            'assets/images/stars.jpg',
            fit: BoxFit.cover,
            height: double.infinity,
            width: double.infinity,
            alignment: Alignment.center,
          ),
          Center(
            child: TweenAnimationBuilder<Color>(
              tween: ColorTween(begin: Colors.white, end:
_newColor),
              duration: Duration(seconds: 4),
              onEnd: () {
                setState(() {
                  _newColor = _newColor == Colors.red ?
Colors.white : Colors.red;
                });
              },
```

Tween Animation

```
                builder: (_, Color color, __) {
                  return ColorFiltered(
                    child:
Image.asset('assets/images/sun.png'),
                    colorFilter: ColorFilter.mode(color,
BlendMode.modulate),
                  );
                },
              ),
            ),
          ],
        ),
      );
  }
}
```

Tween Animation



Transições

Para gerenciar explicitamente uma animação, existem diversos widgets para controlar a transição dos parâmetros da animação baseado no controller como **ScaleTransition**, **SizeTransition**, **RotationTransition**, **DecoratedBoxTransition**, **PositionedTransition** e outros.

Transições

Crie uma nova aplicação Flutter com o nome **animation_transition**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório onde serão armazenadas as imagens:

```
assets:
  - assets/images/
```

Crie o subdiretório **assets/images**.

Copie o arquivo **galaxy.png** no subdiretório **assets/images**.

Transições

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Transições

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> with
SingleTickerProviderStateMixin {
  AnimationController _animationController;

  @override
  void initState() {
    super.initState();
    _animationController = AnimationController(
      duration: Duration(seconds: 15),
      vsync: this, )..repeat();
  }
```

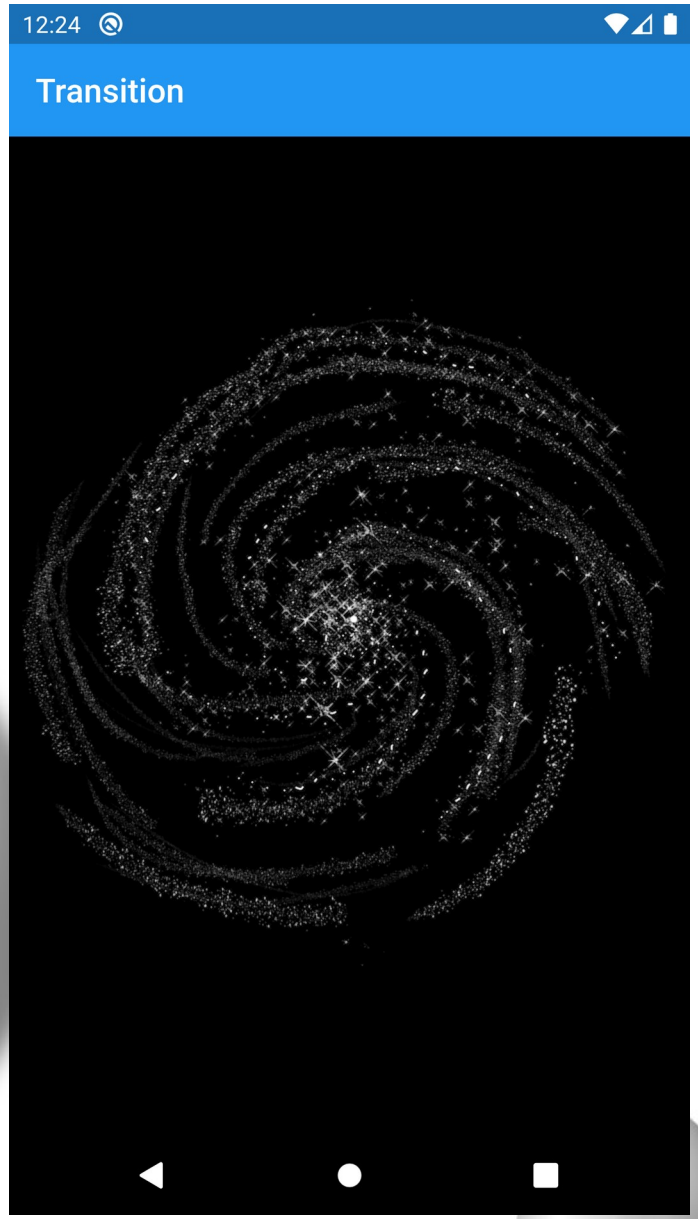
Transições

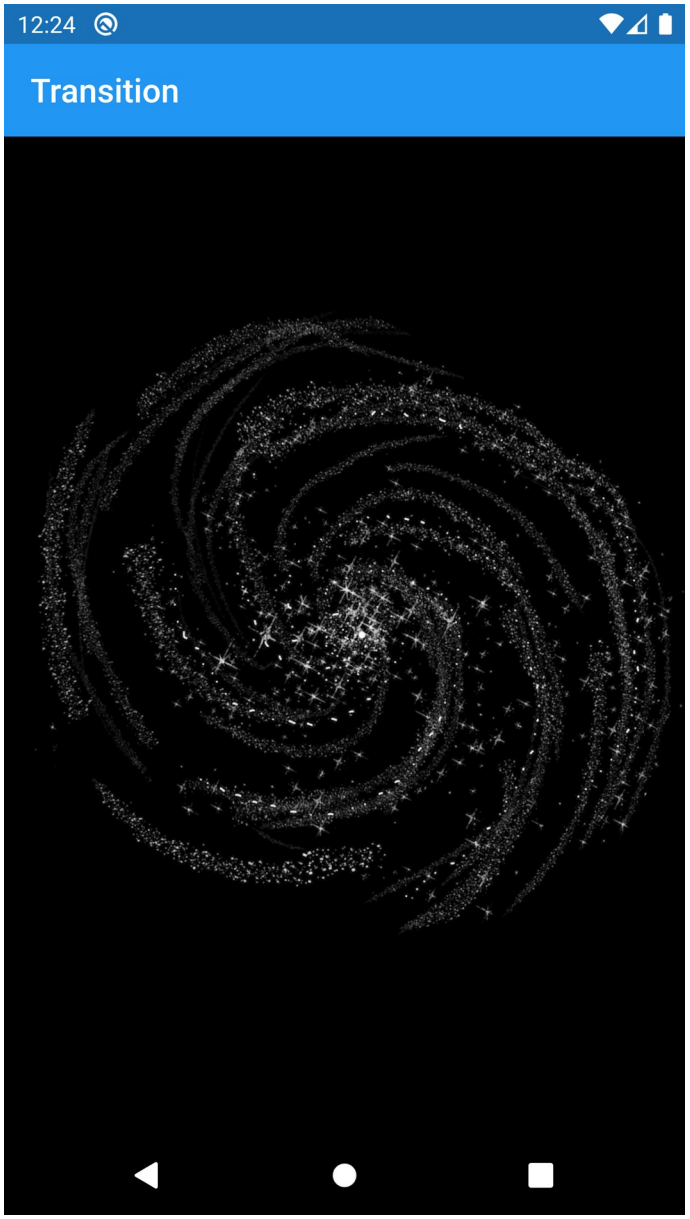
```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("Transition"),),
    body: Stack(
      children: <Widget>[
        Align(
          alignment: Alignment.bottomLeft,
          child:
          SizedBox.expand(
            child: GestureDetector(
              onTap: () {
                if (_animationController.isAnimating)
                  _animationController.stop();
                else
                  _animationController.repeat();
              },
              child: Container( color: Colors.black, ),
            )
          ),
        ),
```

Transições

```
            Align(
              alignment: Alignment.center,
              child: RotationTransition(
                child:
Image.asset('assets/images/galaxy.png',),
                alignment: Alignment.center,
                turns: _animationController,
              ),
            ),
          ],
        ),
      );
    }

    @override
    void dispose() {
      _animationController.dispose();
      super.dispose();
    }
}
```

Transições





AnimatedBuilder

Para criar animações explícitas que não se encaixam nos widgets de transição já existentes é possível utilizar **AnimatedBuilder** ou **AnimatedWidget**.

AnimatedBuilder é usado para incluir a animação dentro de um outro builder.

AnimatedBuilder

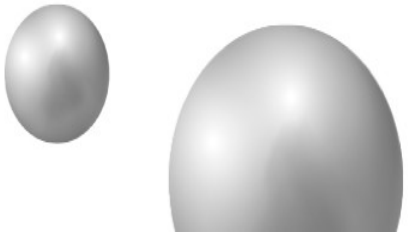
Crie uma nova aplicação Flutter com o nome **animation_builder**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório onde serão armazenadas as imagens:

```
assets:
  - assets/images/
```

Crie o subdiretório **assets/images**.

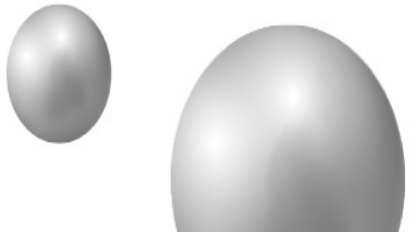
Copie os arquivos **stars.jpg** e **ufo.png** no subdiretório **assets/images**.



AnimatedBuilder

Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```



AnimatedBuilder

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
    @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> with
SingleTickerProviderStateMixin {
  AnimationController _animationController;

  @override
  void initState() {
    super.initState();
    _animationController = AnimationController(
      duration: const Duration(seconds: 5),
      vsync: this,
    )..repeat();
  }
```

AnimatedBuilder

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("AnimatedBuilder"),),
    body: Stack(
      alignment: AlignmentDirectional.center,
      children: <Widget>[
        Image.asset(
          'assets/images/stars.jpg',
          fit: BoxFit.cover,
          height: double.infinity,
          width: double.infinity,
          alignment: Alignment.center,
        ),
```

AnimatedBuilder

```
AnimatedBuilder(
  animation: _animationController,
  builder: (_, __) {
    return ClipPath(
      clipper: const BeamClipper(),
      child: Container(
        height: 1000,
        decoration: BoxDecoration(
          gradient: RadialGradient(
            radius: 1.5,
            colors: [
              Colors.yellow,
              Colors.transparent,
            ],
            stops: [0,
_animationController.value],
          ),
        ),
      ),
    );
  },
),
```

AnimatedBuilder

```
            Image.asset('assets/images/ufo.png'),
          ],
        ),
      );
  }

  @override
  void dispose() {
    _animationController.dispose();
    super.dispose();
  }
}
```

AnimatedBuilder

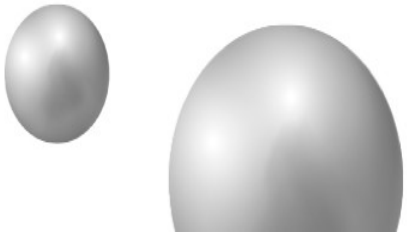
```
class BeamClipper extends CustomClipper<Path> {
  const BeamClipper();

  @override
  getClip(Size size) {
    return Path()
      ..lineTo(size.width / 2, size.height / 2)
      ..lineTo(size.width, size.height)
      ..lineTo(0, size.height)
      ..lineTo(size.width / 2, size.height / 2)
      ..close();
  }

  /// Return false always because we always clip the same
area.
  @override
  bool shouldReclip(CustomClipper oldClipper) => false;
}
```

AnimatedBuilder

AnimatedWidget

AnimatedWidget também pode ser utilizado para criar as mesmas animações, porém criando um widget próprio para a animação.

O uso de **AnimatedBuilder** ou **AnimatedWidget** é praticamente uma questão de escolha.



AnimatedWidget

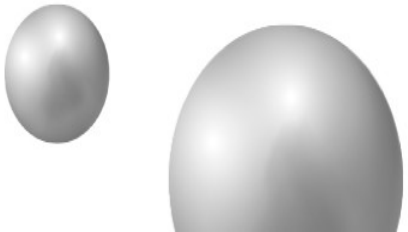
Crie uma nova aplicação Flutter com o nome **animation_widget**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório onde serão armazenadas as imagens:

```
assets:
  - assets/images/
```

Crie o subdiretório **assets/images**.

Copie os arquivos **stars.jpg**, **cow.png** e **ufo.png** no subdiretório **assets/images**.



AnimatedWidget

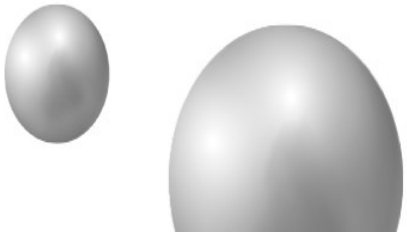
Altere o arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

AnimatedWidget

Crie o subdiretório **lib/pages**. Crie o arquivo **home.dart** no subdiretório **lib/pages**:

```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {

    @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> with
TickerProviderStateMixin {
  AnimationController _beamController;
  AnimationController _cowController;
  Animation<double> _cowAnimation;
```



AnimatedWidget

```
  @override
  void initState() {
    super.initState();
    _beamController = AnimationController(
      duration: const Duration(seconds: 5),
      vsync: this,
    );
    _beamController.addListener(() {
      if (_beamController.status ==
AnimationStatus.completed) {
        setState(() {
          _cowController.reset();
          _cowController.forward();
        });
      };
    });
    _cowController = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );
```

AnimatedWidget

```
    _cowController.addListener(() {
      if (_cowController.status ==
AnimationStatus.completed) {
        setState(() {
          _cowController.reset();
          _beamController.reset();
          _beamController.forward();
        });
      };
    });
    _cowAnimation = Tween<double>(begin: 0, end:
150).animate(_cowController);
    _beamController.forward();
  }
```

AnimatedWidget

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("AnimatedWidget"),),
    body: Stack(
      alignment: AlignmentDirectional.center,
      children: <Widget>[
        Image.asset(
          'assets/images/stars.jpg',
          fit: BoxFit.cover,
          height: double.infinity,
          width: double.infinity,
          alignment: Alignment.center,
        ),
        BeamTransition(animation: _beamController),
        AnimatedCow(animation: _cowAnimation),
        Image.asset('assets/images/ufo.png'),
      ],
    ),
  );
}
```

AnimatedWidget

```
  @override
  void dispose() {
    _beamController.dispose();
    _cowController.dispose();
    super.dispose();
  }
}
```

AnimatedWidget

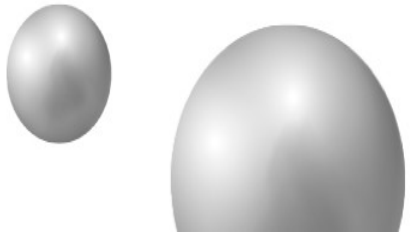
```
class BeamTransition extends AnimatedWidget {
  BeamTransition({Key key, Animation<double> animation})
      : super(key: key, listenable: animation);
  @override
  Widget build(BuildContext context) {
    final Animation<double> animation = listenable;
    return ClipPath(
      clipper: const BeamClipper(),
      child: Container(
        height: 1000,
        decoration: BoxDecoration(
          gradient: RadialGradient(
            radius: 1.5,
            colors: [Colors.yellow,Colors.transparent,],
            stops: [0, animation.value],
          ),
        ),
      ),
    );
  }
}
```

AnimatedWidget

```
class BeamClipper extends CustomClipper<Path> {
  const BeamClipper();

  @override
  getClip(Size size) {
    return Path()
      ..lineTo(size.width / 2, size.height / 2)
      ..lineTo(size.width, size.height)
      ..lineTo(0, size.height)
      ..lineTo(size.width / 2, size.height / 2)
      ..close();
  }

  /// Return false always because we always clip the same
area.
  @override
  bool shouldReclip(CustomClipper oldClipper) => false;
}
```

AnimatedWidget

```
class AnimatedCow extends AnimatedWidget {
  AnimatedCow({Key key, Animation<double> animation})
      : super(key: key, listenable: animation);

  Widget build(BuildContext context) {
    final animation = listenable as Animation<double>;
    return Positioned(
      bottom: animation.value*1.2,
      height: animation.value,
      width: animation.value,
      child: Image.asset('assets/images/cow.png',),
    );
  }
}
```



AnimatedWidget