Я (не) робот

Fedor Blagodyr
Flutter Software Engineer

Что такое CAPTCHA ?

Вот что говорит сам Google

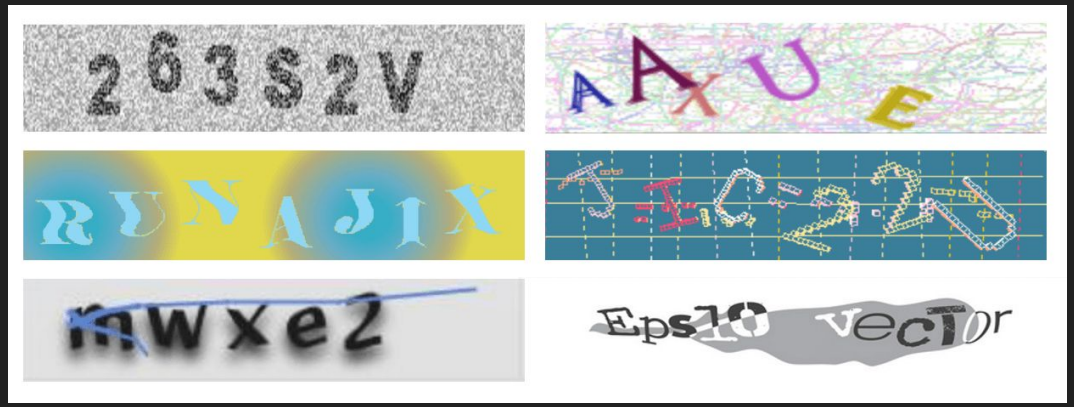
Проверочный код CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart – Полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей) – это одна из разновидностей мер безопасности, известная как аутентификация "вызов-ответ".

Проверочный код защищает от спама и кражи паролей. Для проверки необходимо выполнить простой тест, подтверждающий, что действия выполняет человек, а не компьютерная программа, пытающаяся получить доступ к защищенному паролем аккаунту.

Где это может использовать бизнес?

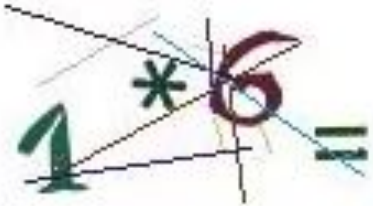
- при регистрации в сервисе
- при смене пароля существующего аккаунта
- при неудачных попытках входа в существующий аккаунт
- при изменении важной информации пользователя
- и многое другое

Распознавание текста



Традиционный тип капчи, который требует от пользователя ввести ряд цифр и букв. Чаще всего текстовая строка искажена различными цветовыми и шумовыми фильтрами, а символы в ней перечеркнуты или имеют наклон.

Логическая задача



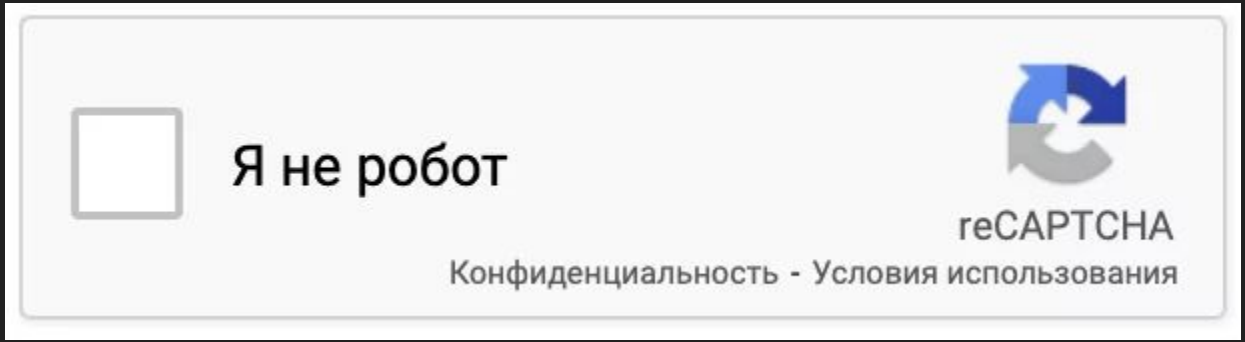
What is the result of this math equation? *

6

Solve the math equation pictured in the image, and enter the result.

Она проверяет, может ли условный пользователь перед ней мыслить.

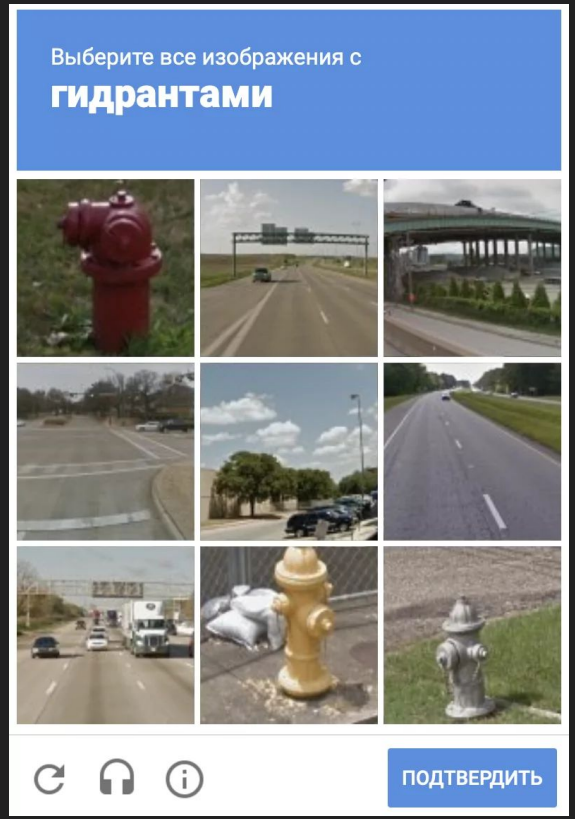
Капча «Я не робот»



Требует, чтобы пользователь установил флажок, чтобы доказать, что он не робот

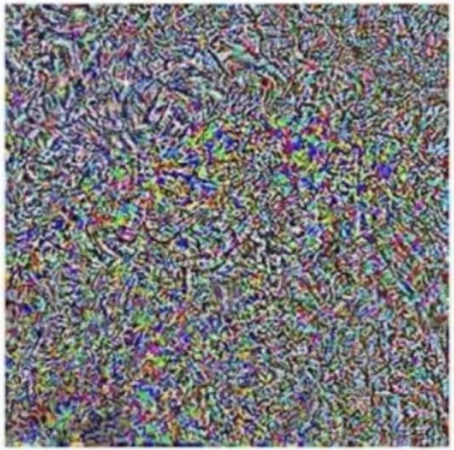
Выбор изображений

Такая капча просит пользователей идентифицировать набор фотографий (к примеру, указать все изображения с гидрантами).



Выбор изображений

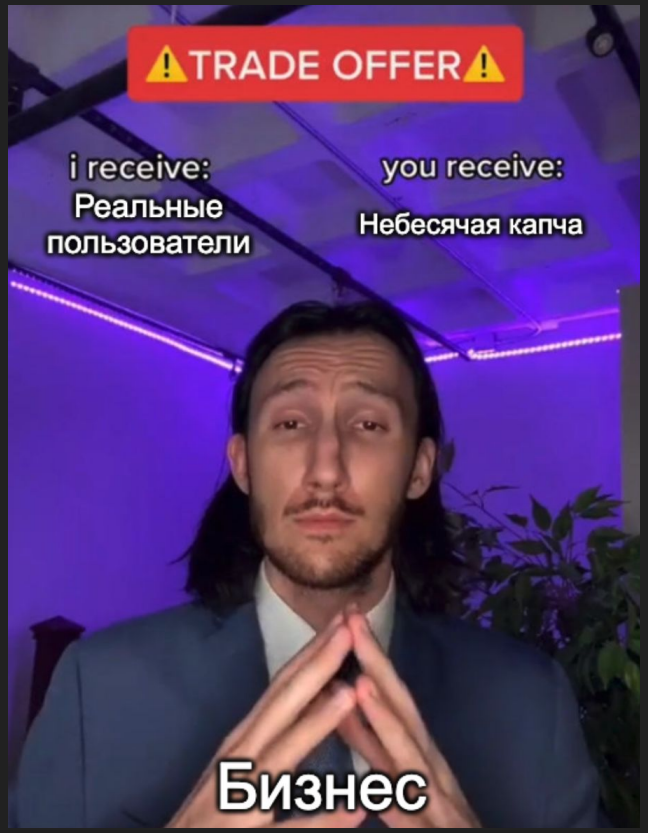
Бот понимает, что перед ним рыба



Для капчи на базовое изображение накладывается шум

Бот не понимает, что перед ним рыба, и думает, что ему показывают краба

А вообще зачем нам делать свою?



Flutter

Mobile

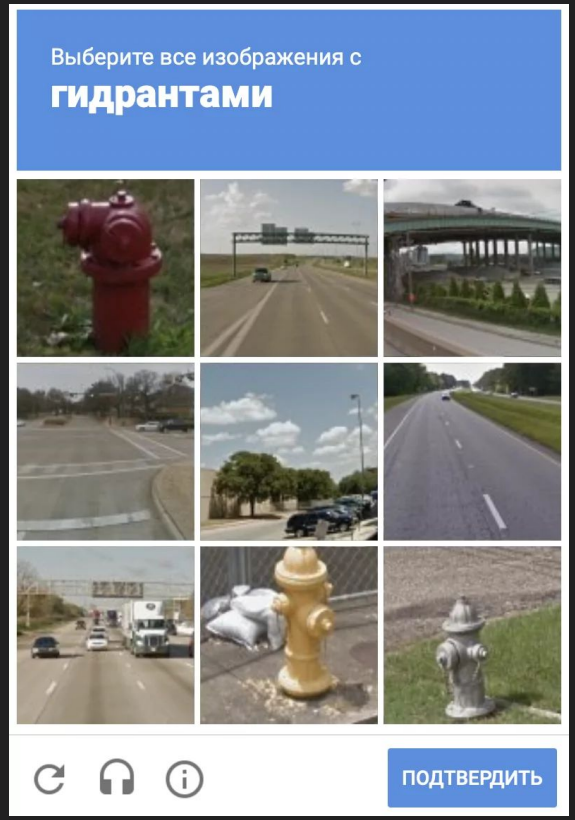
Web

Desktop

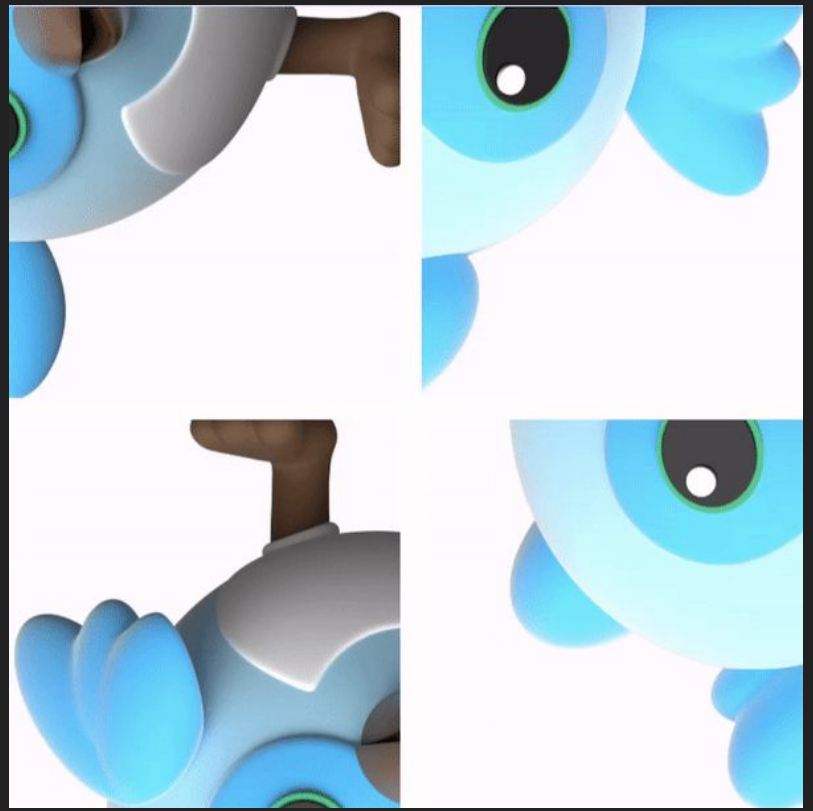
Embedded

Ок, а какую будем делать?

Возьмем за основу и натюним

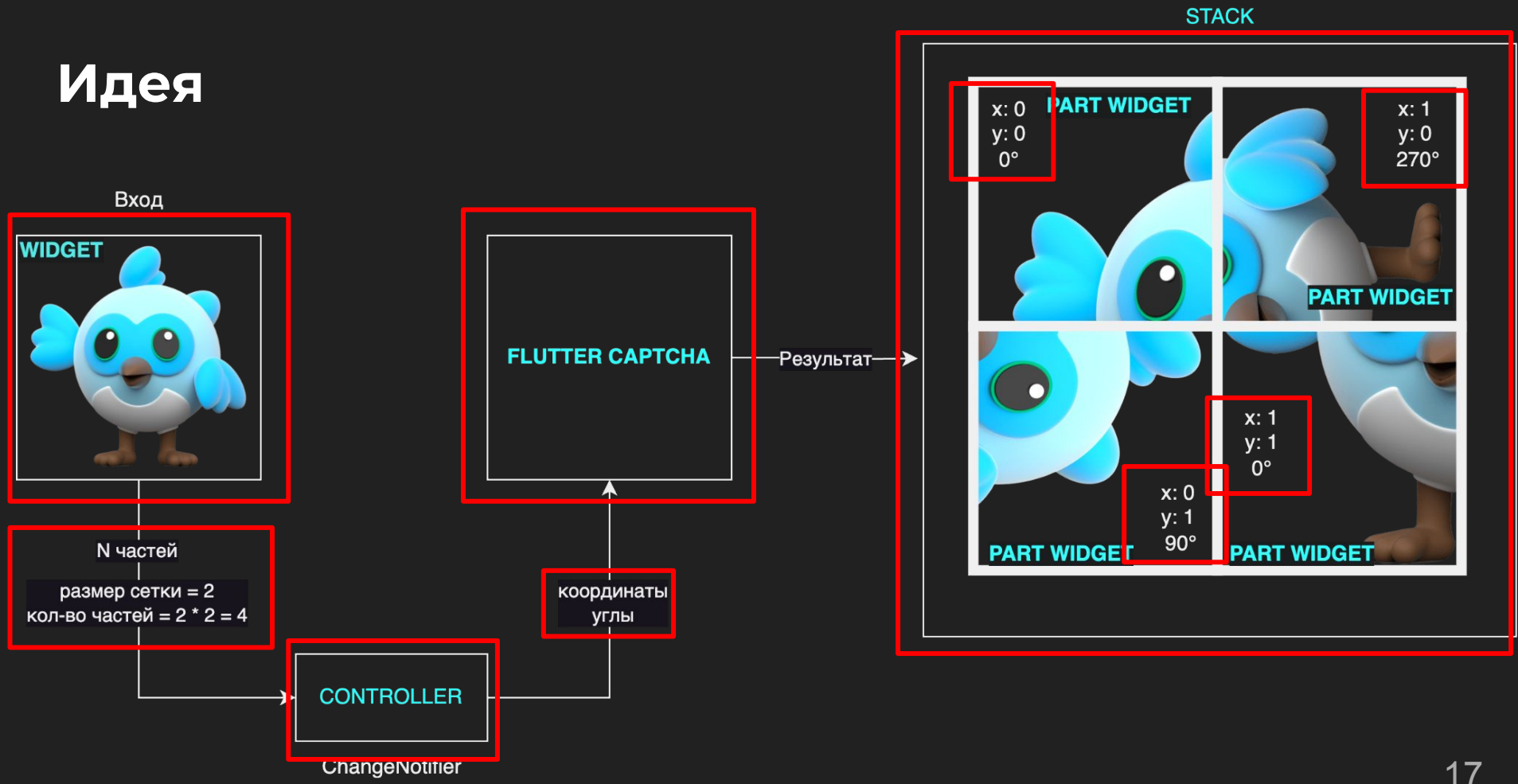


Что будет в итоге

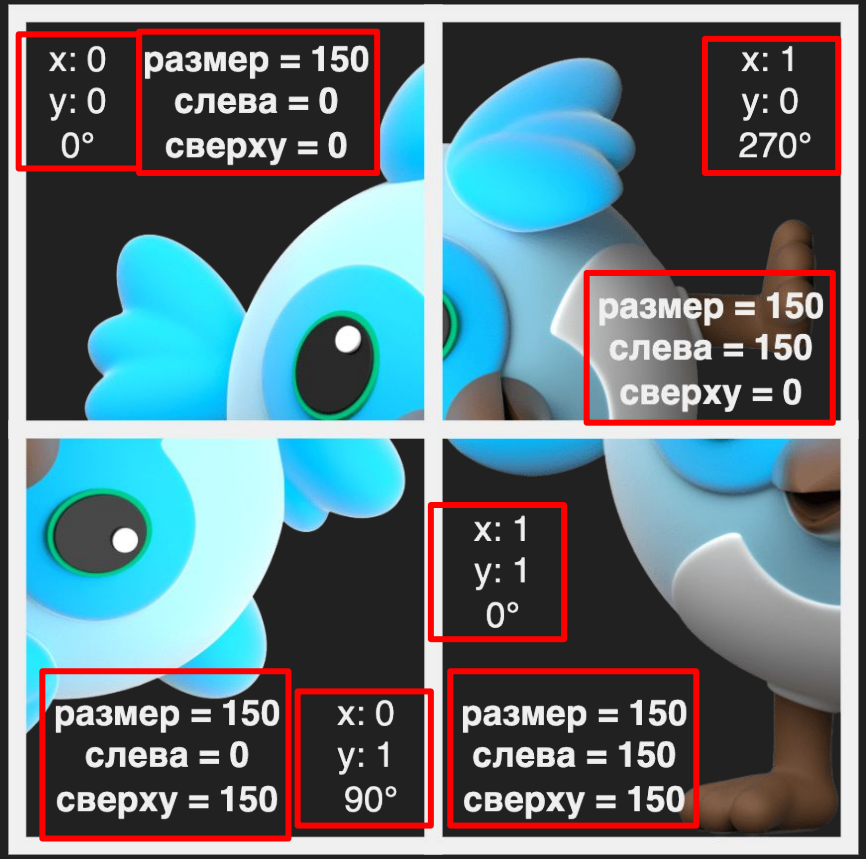



Playground

Идея


Вход
WIDGET
N частей
размер сетки = 2
кол-во частей = 2 * 2 = 4
CONTROLLER
ChangeNotifier
координаты
углы
FLUTTER CAPTCHA
Результат
STACK
x: 0
y: 0
0°
PART WIDGET
x: 1
y: 0
270°
PART WIDGET
x: 1
y: 1
0°
x: 0
y: 1
90°
PART WIDGET
PART WIDGET

Идея



Размер сетки = 2
Кол-во частей = размер сетки * размер сетки
Размер капчи в пикселях = 300
Размер части в пикселях = размер капчи в пикселях / размер сетки = 150

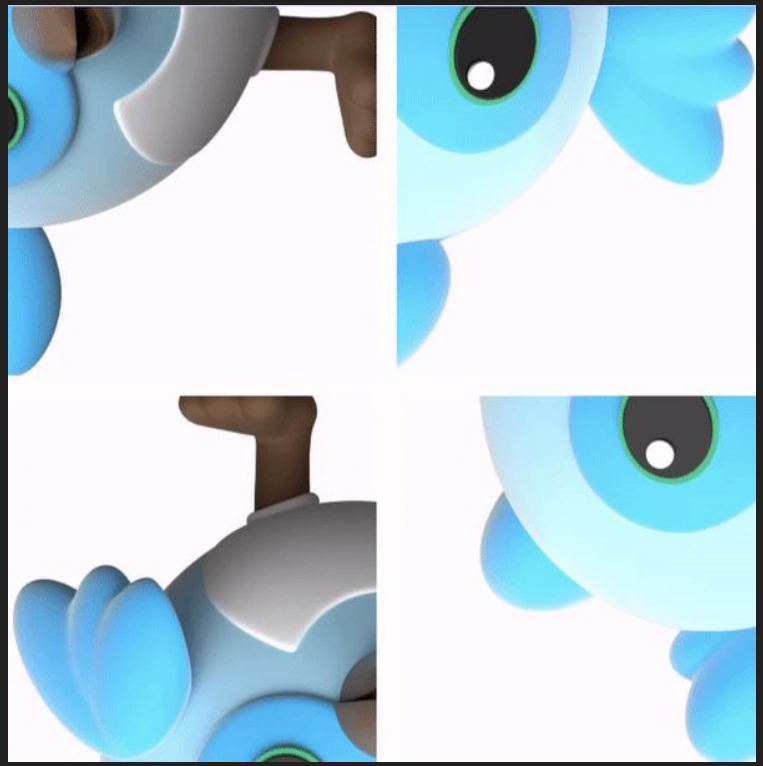
Реальная позиция рассчитывается по принципу умножения координаты на размер части

Отступ сверху = y * размер части в пикселях
Отступ слева = x * размер части в пикселях

Что для этого понадобится?

Что для этого понадобится?

- ChangeNotifier
- Draggable/DragTarget
- CustomPainter
- CustomClipper
- Random
- RenderObject
- Implicit Animations



Информация о координатах

```
typedef CaptchaPoint = ({
    int x,
    int y,
});
```

Информация о размерах

```
typedef CaptchaLayout = ({
    double dimension,
    double partSize,
});
```

Информация об угле

```
final class Angle {
    final double value;
    double get absoluteValue => value % 1;
    bool get isZero => absoluteValue == 0.0;
    Angle operator +(Angle other) => Angle._(value + other.value);

    const Angle._(this.value);
    factory Angle.zero() => const Angle._(0.0);
    factory Angle.quarter() => const Angle._(1 / 4);
    factory Angle.half() => const Angle._(2 / 4);
    factory Angle.third() => const Angle._(3 / 4);
}
```

Контроллер части

```
class FlutterCaptchaPartController extends ChangeNotifier {
    ///...
}
```

Контроллер части

```dart
final CaptchaPoint _solutionPoint;
```

```dart
CaptchaPoint _point;
CaptchaPoint get point => _point;
set point(CaptchaPoint value) {
    //...
    notifyListeners();
}
```

```dart
Angle _angle;
Angle get angle => _angle;
set angle(Angle angle) {
    //...
    notifyListeners();
}
```

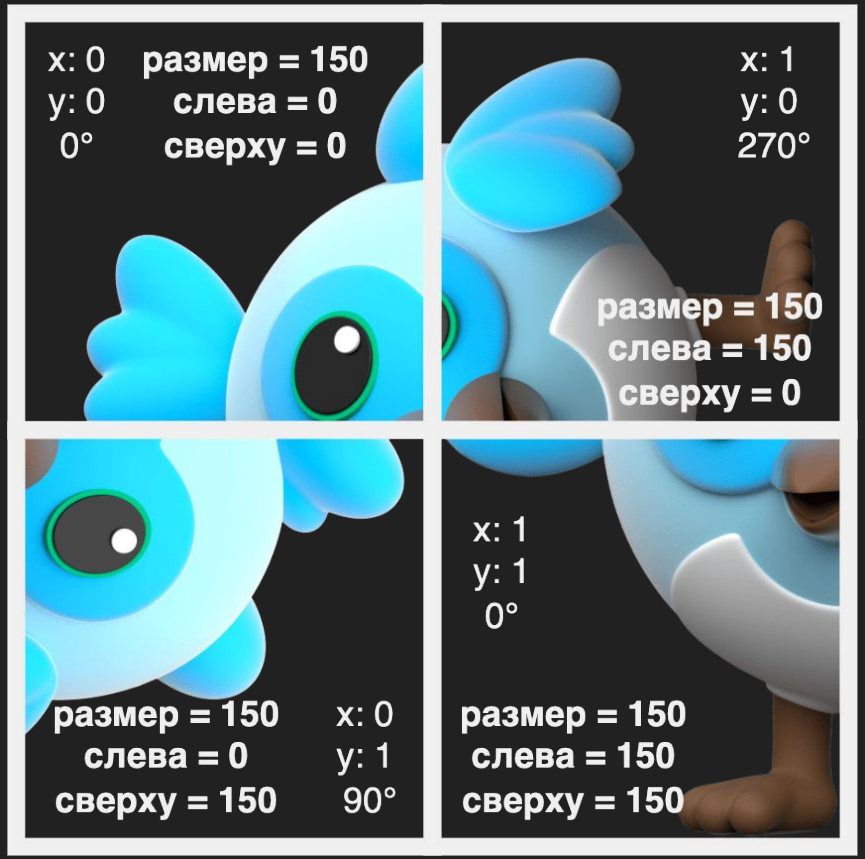
```dart
bool get solved => _angle.isZero && _point == _solutionPoint;
bool _canMove(CaptchaPoint point) => point != _point;
```

Контроллер части

```
void swapPoints(FlutterCaptchaPartController other) {
    final otherNewPosition = other.point;
    other.point = point;
    point = otherNewPosition;
}
```

Ясно
А как отображац

Идея

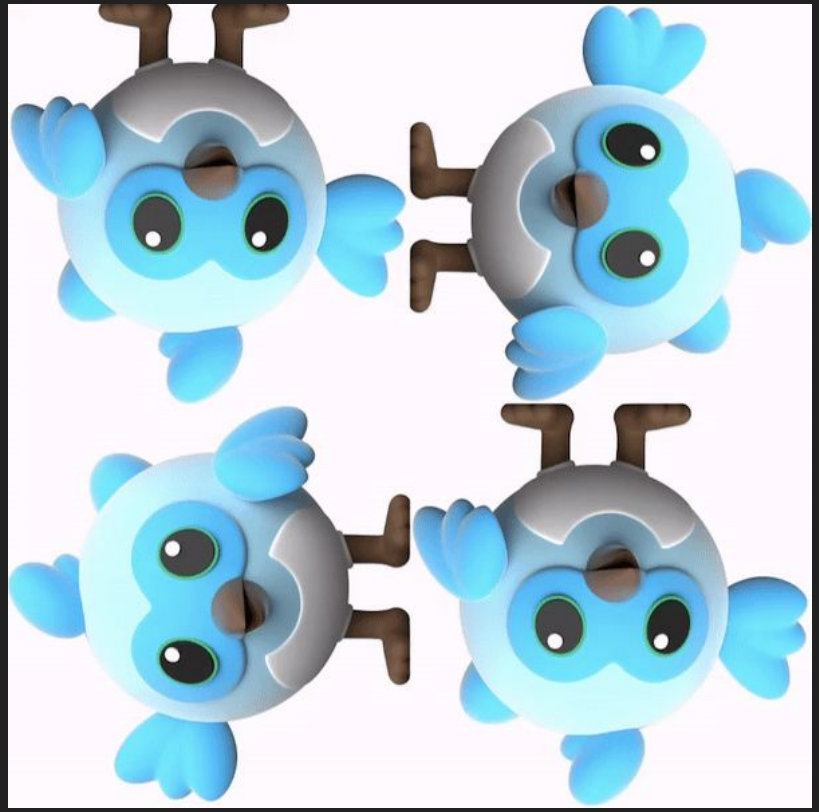


Размер сетки = 2
Кол-во частей = размер сетки * размер сетки
Размер капчи в пикселях = 300
Размер части в пикселях = размер капчи в пикселях / размер сетки = 150

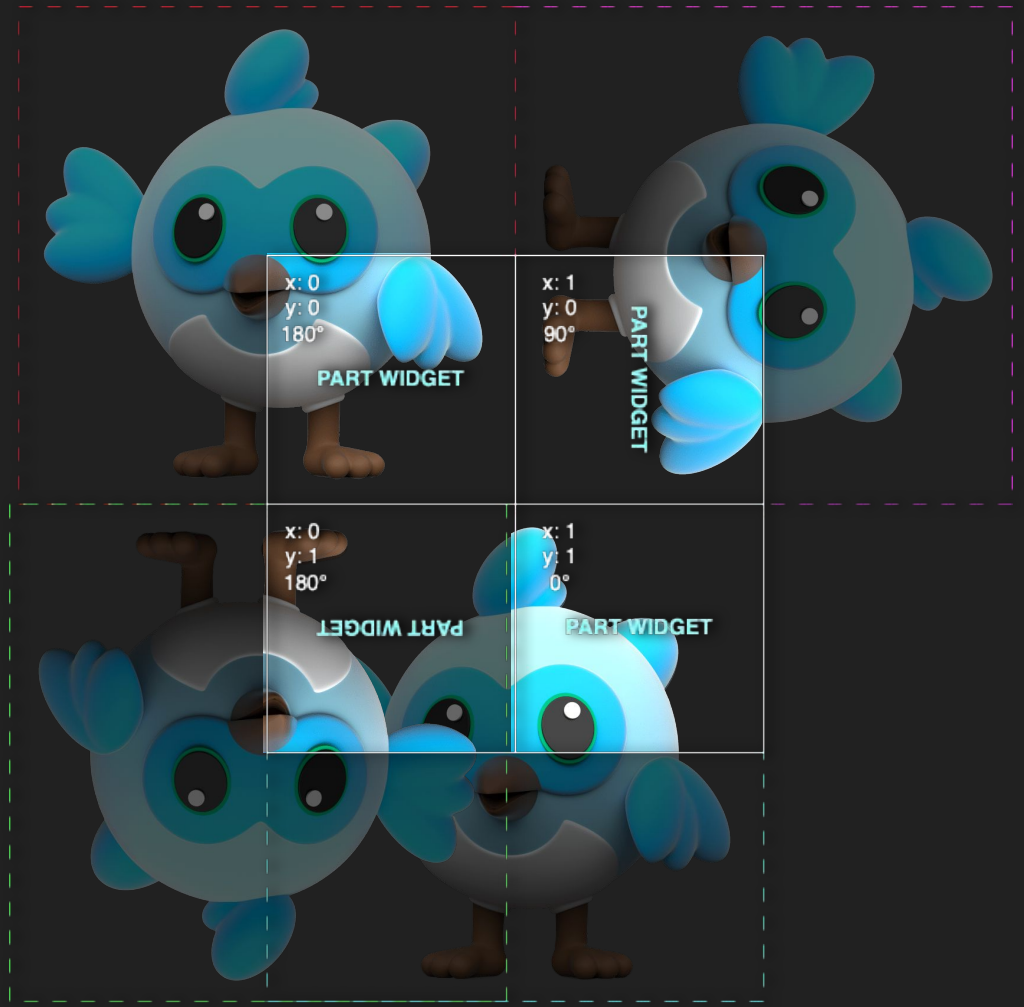
Реальная позиция рассчитывается по принципу умножения координаты на размер части

Отступ сверху = y * размер части в пикселях
Отступ слева = x * размер части в пикселях

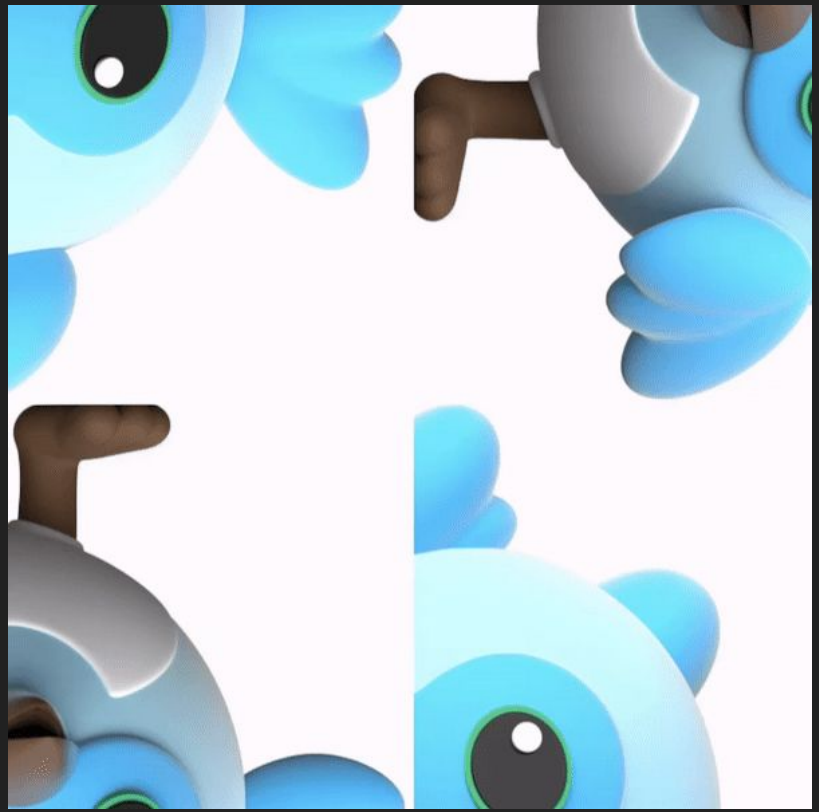
Идея



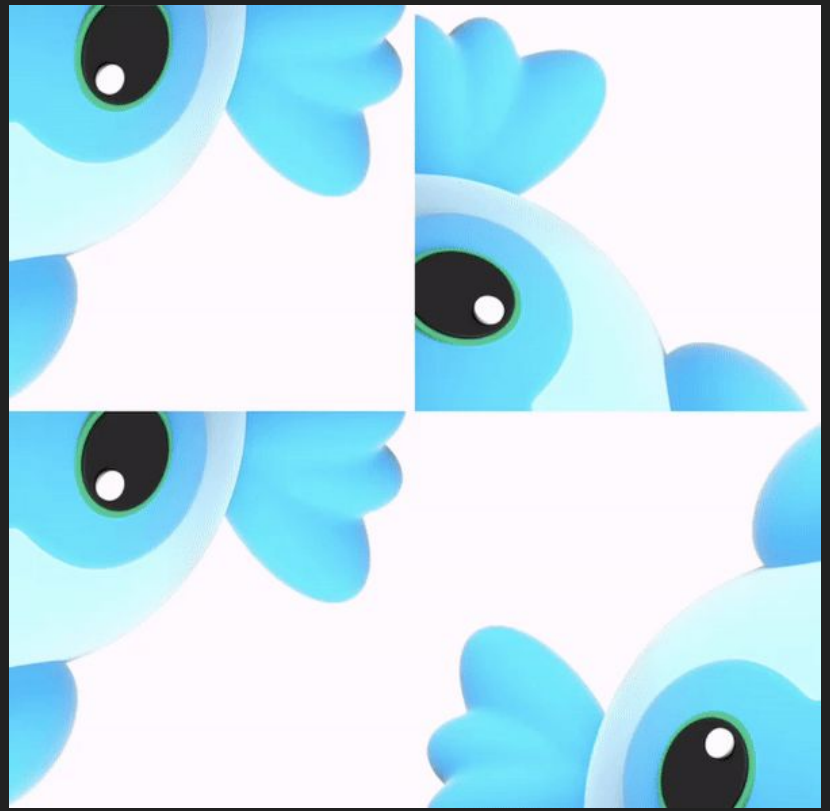
Идея


x: 0
y: 0
180°
PART WIDGET
x: 1
y: 0
90°
PART WIDGET
x: 0
y: 1
180°
PART WIDGET
x: 1
y: 1
0°
PART WIDGET

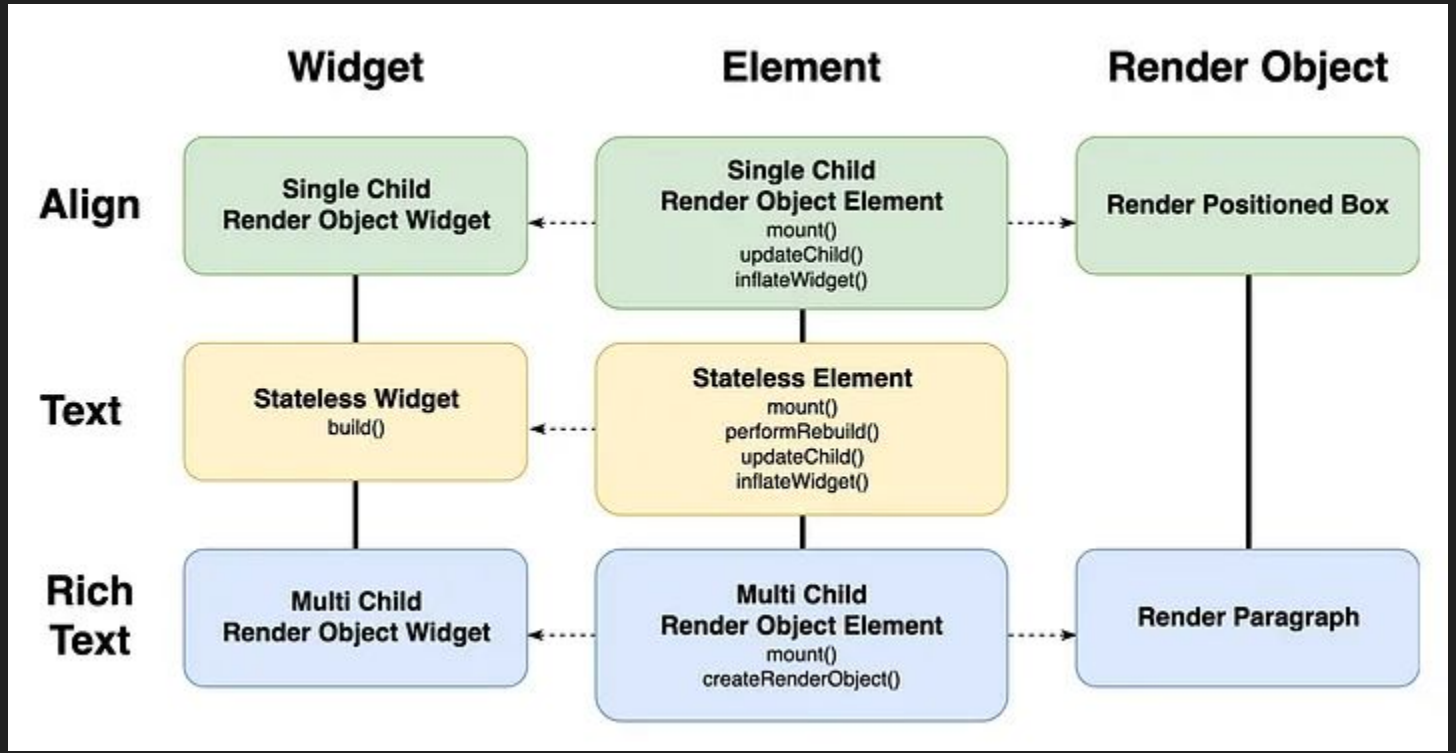
Идея



Идея



RenderObject



Часть капчи widget

```
class _Part extends SingleChildRenderObjectWidget {
    final double dimension;
    final double size;
    final CaptchaPoint solutionPoint;
    //...
}
```

Часть капчи widget

```
@override
void performLayout() {
    //...
}
```

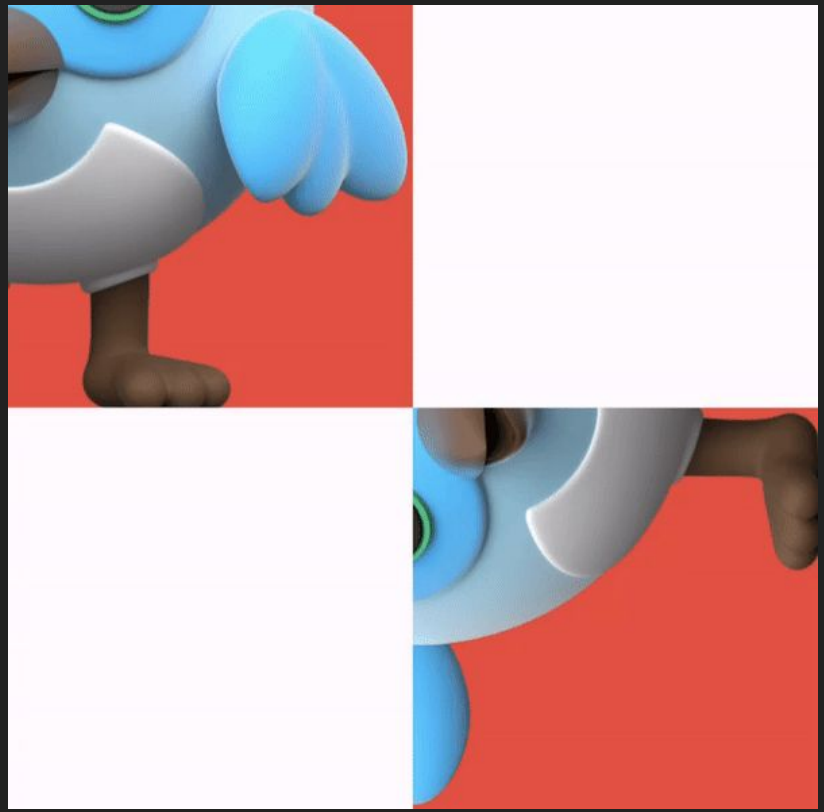
```
@override
void paint(PaintingContext context, Offset offset) {
    //...
}
```

Часть капчи widget

```
void performLayout() {
    super.performLayout();

    child!.layout(
        BoxConstraints.tightFor(
            width: _dimension,
            height: _dimension,
        ),
        parentUsesSize: true,
    );
}
```

Часть капчи widget



Часть капчи widget

```
void paint(PaintingContext context, Offset offset) {
    layer = context.pushClipRect(
        needsCompositing,
        offset,
        Rect.fromLTWH(0, 0, _partSize, _partSize),
            (context, offset) {
                context.canvas.translate(
                    -_solutionPoint.x * _partSize,
                    -_solutionPoint.y * _partSize,
                );
                super.paint(context, offset);
            },
        oldLayer: layer as ClipRectLayer?,
    );
}
```

Фух, осталось только все остальное

Часть капчи widget

```
class FlutterCaptchaPart extends StatefulWidget {
    final Widget child;
    final FlutterCaptchaPartController controller;
    final CaptchaLayout layout;
}
```

Часть капчи widget

```
class _FlutterCaptchaPartState extends State<FlutterCaptchaPart> {
    @override
    void initState() {
        super.initState();
        widget.controller.addListener(_rebuild);
    }
    @override
    void dispose() {
        widget.controller.removeListener(_rebuild);
        super.dispose();
    }
    void _rebuild() => setState(() {});
}
```

Часть капчи widget

```
@override
Widget build(BuildContext context) {
    ///...
}
```

Часть капчи widget

```
Widget child = FittedBox(
    fit: BoxFit.cover,
    child: widget.child,
);
```

Часть капчи widget

```
child = _Part(
    dimension: widget.layout.dimension,
    size: widget.layout.partSize,
    solutionPoint: widget.controller._solutionPoint,
    child: SizedBox.square(
        dimension: widget.layout.partSize,
        child: child,
    ),
);
```

Часть капчи widget

```
Widget result = DragTarget<FlutterCaptchaPartController>(
    onWillAccept: (data) => widget.controller._canMove(data!._point),
    onAcceptWithDetails: (details) =>
        widget.controller.swapPoints(details.data),
    builder: (context, _, __) => child,
);
```

Часть капчи widget

```
result = GestureDetector(
    onTap: () => widget.controller.angle += Angle.quarter(),
    child: AnimatedRotation(
        duration: const Duration(milliseconds: 300),
        turns: widget.controller.angle.value,
        child: result,
    ),
);
```

Часть капчи widget

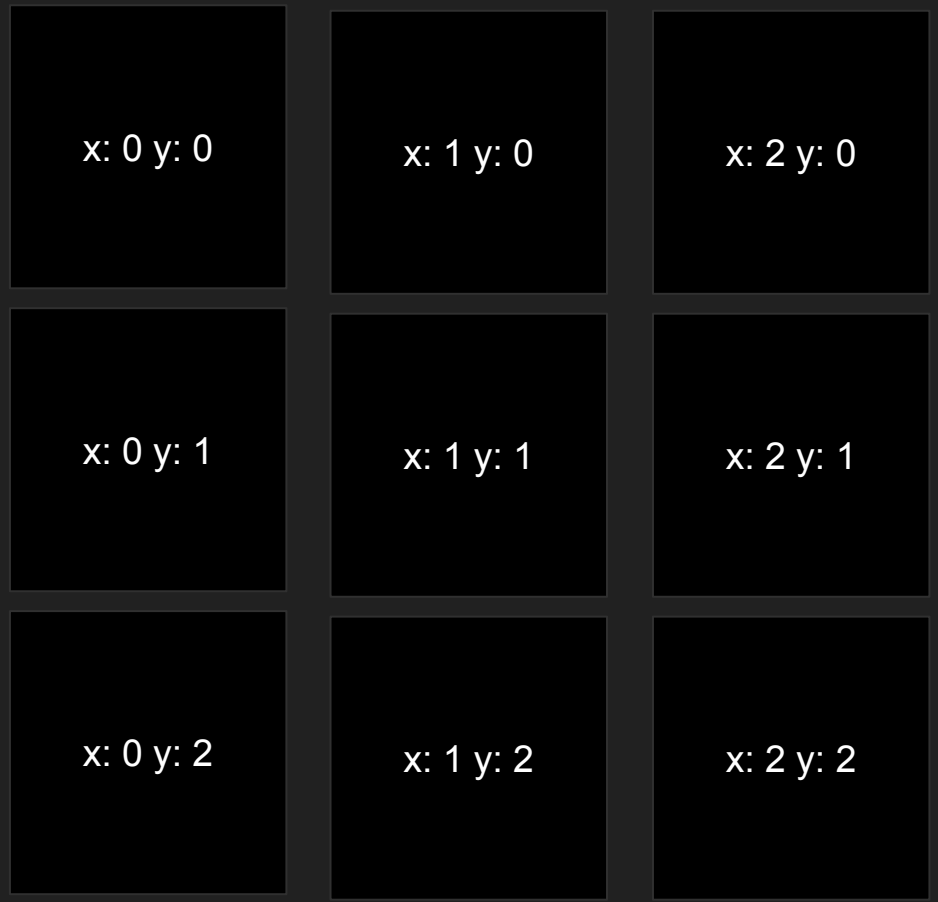
```
final rotated = Transform.rotate(
    angle: widget.controller.angle.value * (2 * math.pi),
    child: child,
);
result = Draggable(
    data: widget.controller,
    childWhenDragging: rotated,
    feedback: rotated,
    child: result,
);
```

Часть капчи widget

```
return AnimatedPositioned(
    top: widget.layout.partSize * widget.controller.point.y,
    left: widget.layout.partSize * widget.controller.point.x,
    duration: const Duration(milliseconds: 300),
    child: ClipPath(child: result),
);
```

Хилимся живем

Идея



Размер сетки = 3
Кол-во частей = 9

Контролер капчи

```
final class FlutterCaptchaController extends ChangeNotifier {
    final _random = math.Random();
    int _size;
    final _controllers = <FlutterCaptchaPartController>[];

    FlutterCaptchaController({
            int size = 2,
        }) : _size = size;
    }
    // ...
}
```

Контролер капчи

```
void init() {
    _setupControllers(_createPoints());

    notifyListeners();
}
```

```
List<CaptchaPoint> _createPoints() {
    final count = _size * _size;
    final points = <CaptchaPoint>[];

    for (var i = 0; i < count; i++) {
        final x = (i % _size);
        final y = (i ~/ _size);

        points.add((x: x, y: y));
    }

    return points;
}
```

Контролер

```
void _setupControllers(List<CaptchaPoint> points) {
    //..
}
```

Контролер капчи

```
final startPoints = points.toList();
startPoints.shuffle(_random);
```

```
final angles = [
    Angle.zero(),
    Angle.quarter(),
    Angle.half(),
    Angle.third(),
];
```

```
for (var i = 0; i < points.length; i++) {
    final startPoint = startPoints[i];
    final solutionPoint = points[i];
    final angleIndex = _random.nextInt(angles.length);
    final angle = angles[angleIndex];

    _controllers.add(
        FlutterCaptchaPartController(
                angle: angle,
                startPoint: startPoint,
                solutionPoint: solutionPoint,
            ),
    );
}
```

Контролер капчи

```
void init() {
    _setupControllers(_createPoints());

    notifyListeners();
}
```

Контролер капчи

```
bool checkSolution() =>
    _controllers.isNotEmpty && _controllers.every((e) => e.solved);
```

Ясно
А как отображац

Widget капчи

```dart
final class FlutterCaptcha extends StatefulWidget {
    final Widget child;
    final double? dimension;
    final FlutterCaptchaController controller;
    //...
}
```

Widget капчи

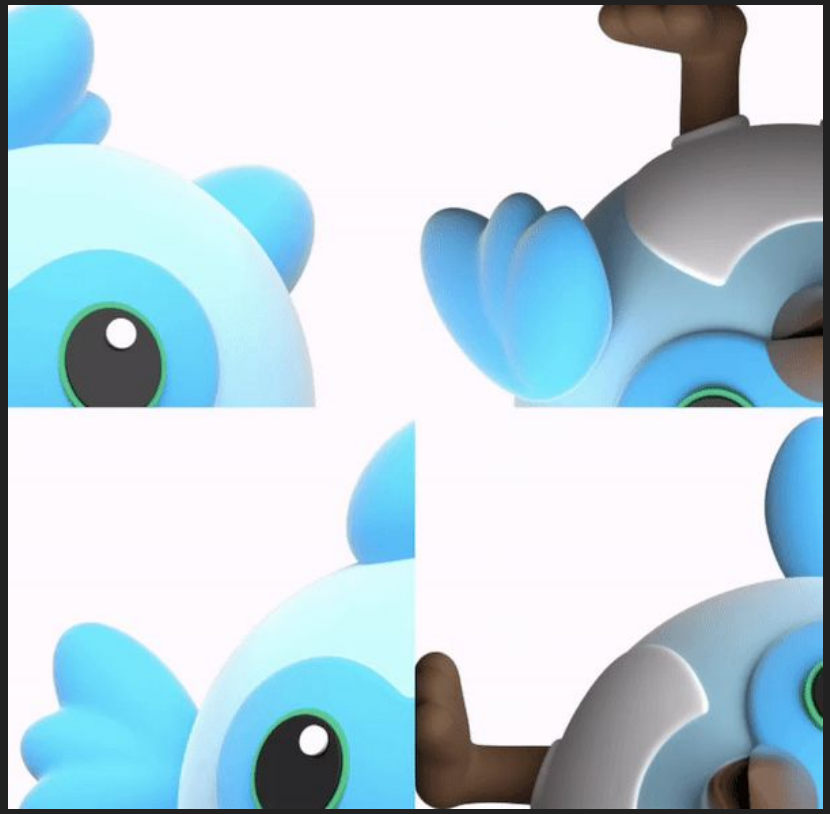
```
@override
Widget build(BuildContext context) {
    //...
}
```

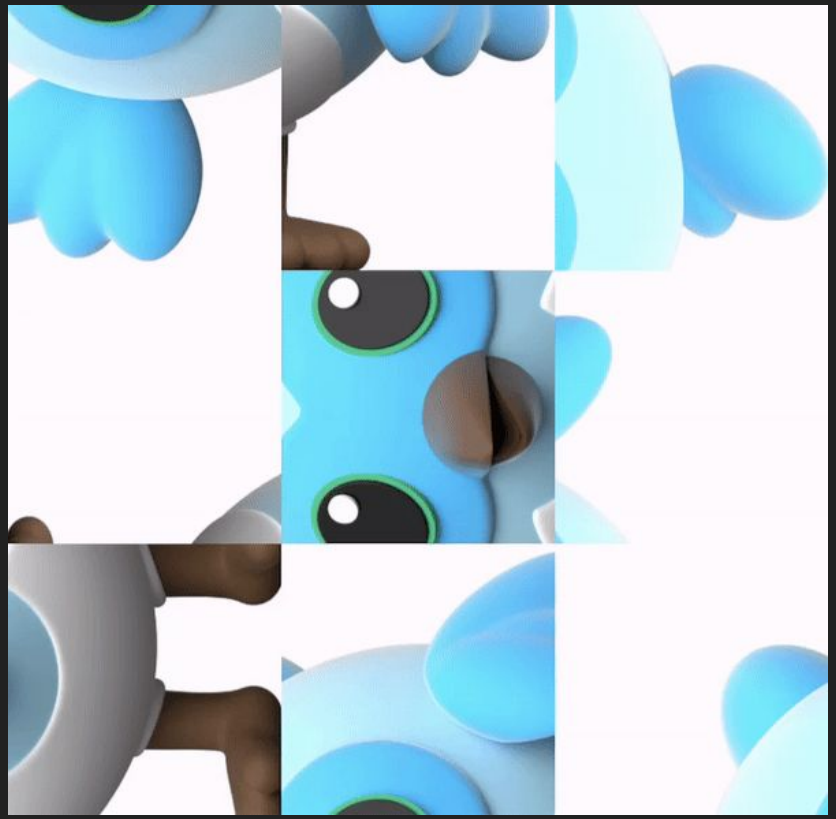
Widget капчи

```
return RepaintBoundary(
    child: LayoutBuilder(
    builder: (_, constraints) {
            //..
        },
    ),
);
```

```dart
final dimension = widget.dimension ?? constraints.biggest.shortestSide;
final partSize = dimension / widget.controller._size;
return SizedBox.square(
    dimension: dimension,
    child: Stack(
        children: [
            for (final controller in widget.controller._controllers)
                FlutterCaptchaPart(
                    key: ObjectKey(controller),
                    layout: (dimension: dimension, partSize: partSize),
                    controller: controller,
                    child: widget.child,
                ),
        ],
    ),
);
```

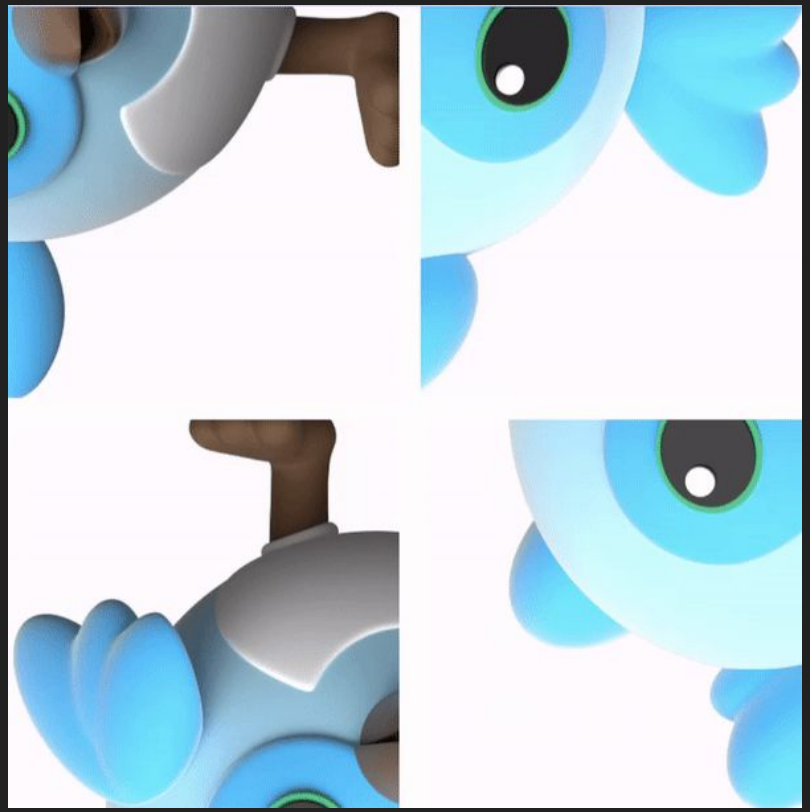
Кайфуем от того какие мы классные

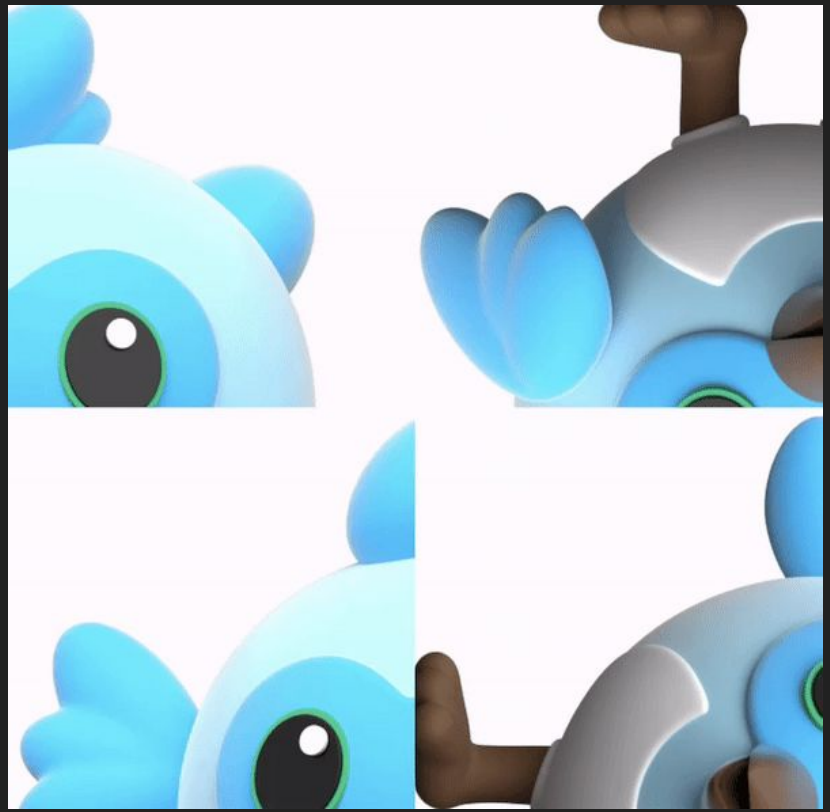




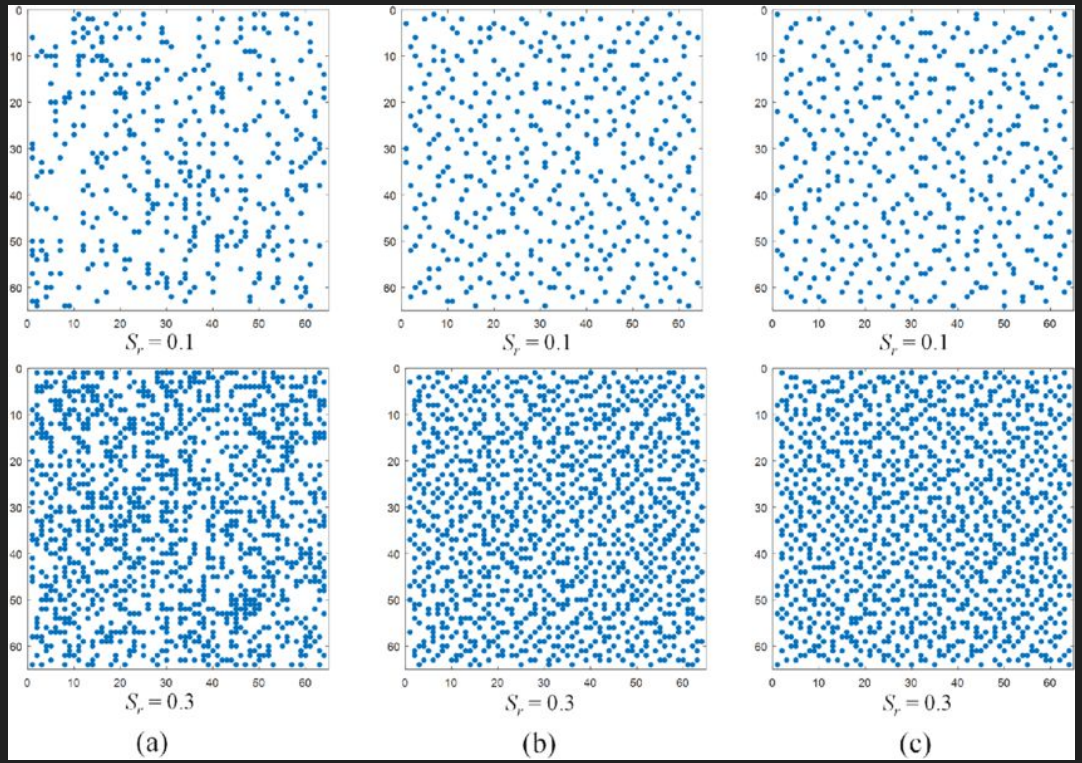
Все классно, но есть нюанс

Нюанс



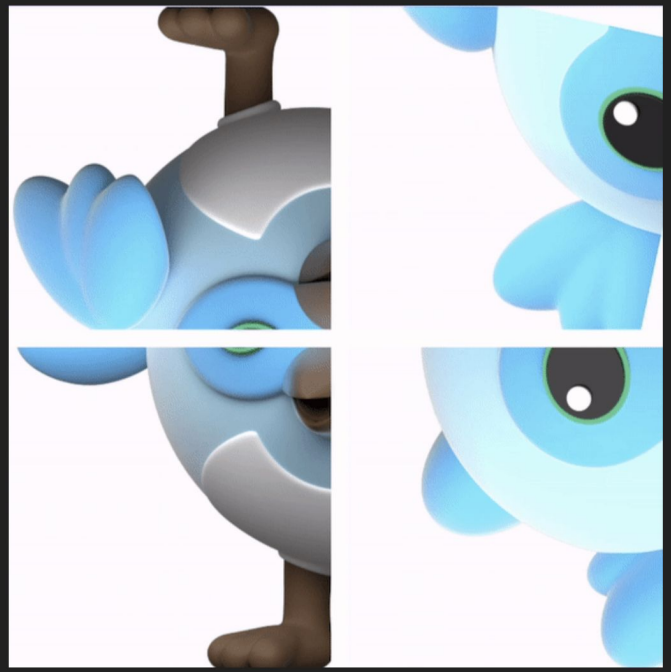


Защита от сканирования пикселей.



Боты часто используют анализ пикселей, чтобы попытаться понять содержимое изображения. Добавление линий между частями вносит визуальный шум и сложность, что затрудняет ботам сегментацию и распознавание отдельных частей. Эти линии нарушают визуальную целостность изображения, что делает сложным задачу определения границ индивидуальных частей алгоритмами.

Затруднение определения точек разделения



Добавление линий между частями помогает затруднить определение точек, где исходный виджет был разделен.

Боты обычно ищут образы и четкие грани, чтобы определить, где начинаются и заканчиваются различные элементы.

Разорванные линиями грани усложняют задачу автоматическим алгоритмам выявления точек разделения.

Сложность для алгоритмов анализа изображений

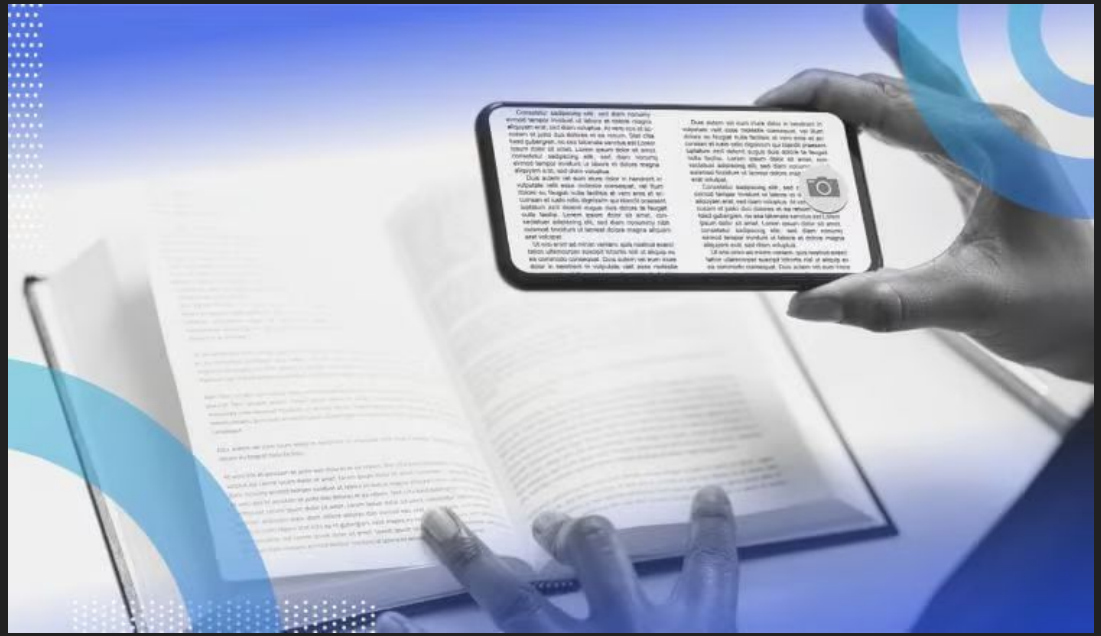
Алгоритмы машинного обучения, включая те, которые используют боты, опираются на обучающие данные для определения объектов и образов.

Введение линий и дополнительных углов создает более сложную структуру изображения, которая не соответствует стандартным образам, что затрудняет классификацию и анализ ботами.

Человек справится

В то время как боты могут быть разработаны для распознавания образов и граней, люди естественным образом лучше различают сложные изображения и идентифицируют объекты в них, даже если грани нарушены линиями или присутствуют какие-то искажения с шумами.

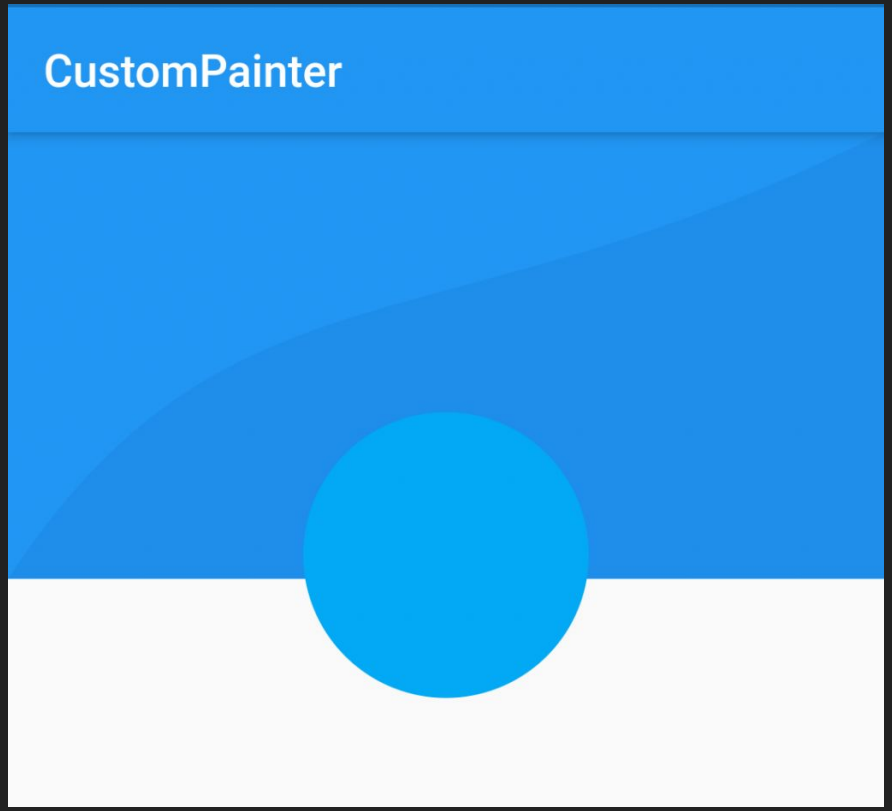
Противостояние атакам с оптическим распознаванием символов (OCR)



Добавление линий уменьшает вероятность атак с использованием OCR, так как линии мешают непрерывному потоку символов, что делает сложнее извлечение осмысленной информации из изображения.

Защитим наше приложение от Skynet

CustomPainter



Линии - разделители

```
typedef FlutterCaptchaCrossLine = ({
    double width,
    Color color,
});
```

Линии - разделители

```
class _CrossLinePainter extends CustomPainter {
    final FlutterCaptchaCrossLine crossLine;
    final int count;
    //..
}
```

Линии - разделители

```
@override
void paint(Canvas canvas, Size size) {
    //..
}
```

Линии - разделители

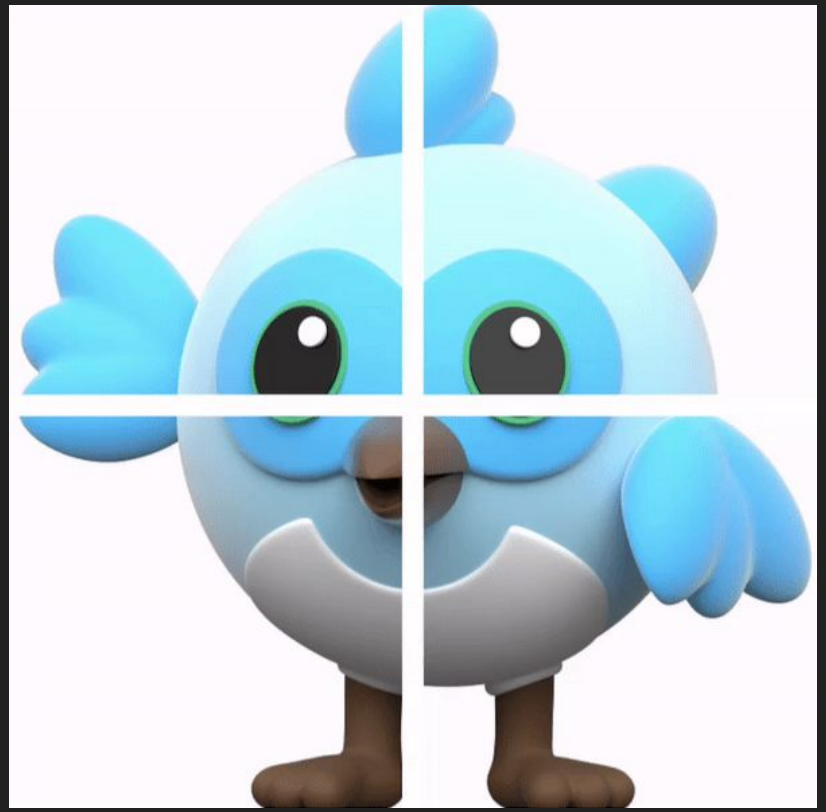
```
final double cellWidth = size.width / count;
final double cellHeight = size.height / count;
```

```
final paint = Paint()
    ..color = crossLine.color
    ..strokeWidth = crossLine.width;
```

```
for (var i = 1; i < count; i++) {
    final y = i * cellHeight;
    final x = i * cellWidth;
    canvas.drawLine(Offset(0, y), Offset(size.width, y), paint);
    canvas.drawLine(Offset(x, 0.0), Offset(x, size.height), paint);
}
```

```
Stack(
children: [
     for (final controller in widget.controller.controllers)
          FlutterCaptchaPart(
               key: ObjectKey(controller),
               layout: (dimension: dimension, partSize: partSize),
               controller: controller,
               child: widget.child,
          ),
          IgnorePointer(
               child: CustomPaint(
               size: Size.square(dimension),
               painter: _CrossLinePainter(
                    crossLine: widget.crossLine,
                    count: widget.controller.size,
               ),
          ),
     ),
]
```

Линии - разделители



Линии - разделители

```
class _FeedbackClipper extends CustomClipper<Rect> {
    final FlutterCaptchaCrossLine crossLine;
    final CaptchaPoint point;
    final CaptchaLayout layout;
}
```

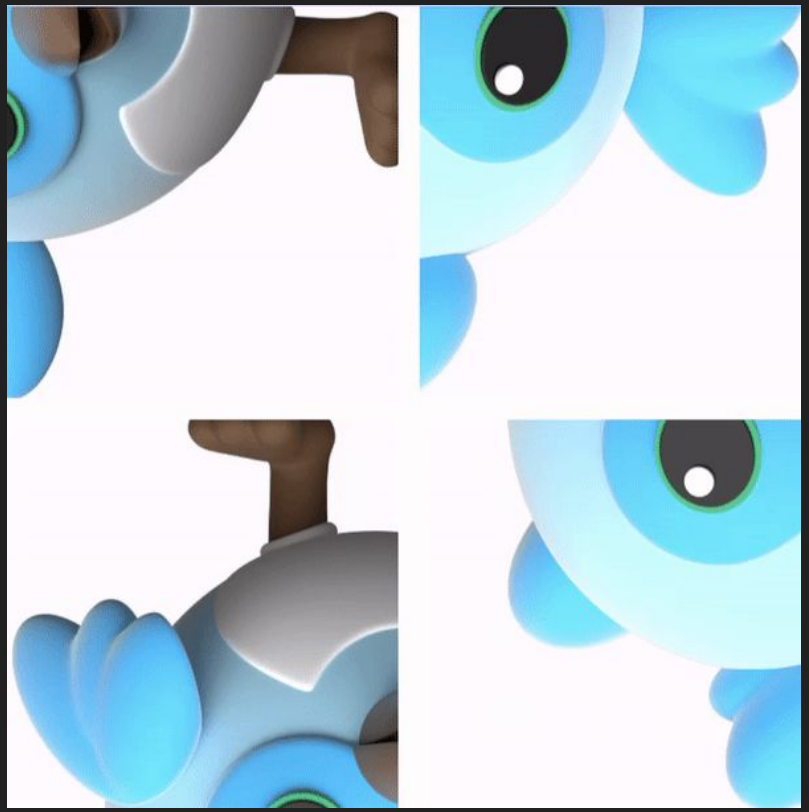
Линии - разделители

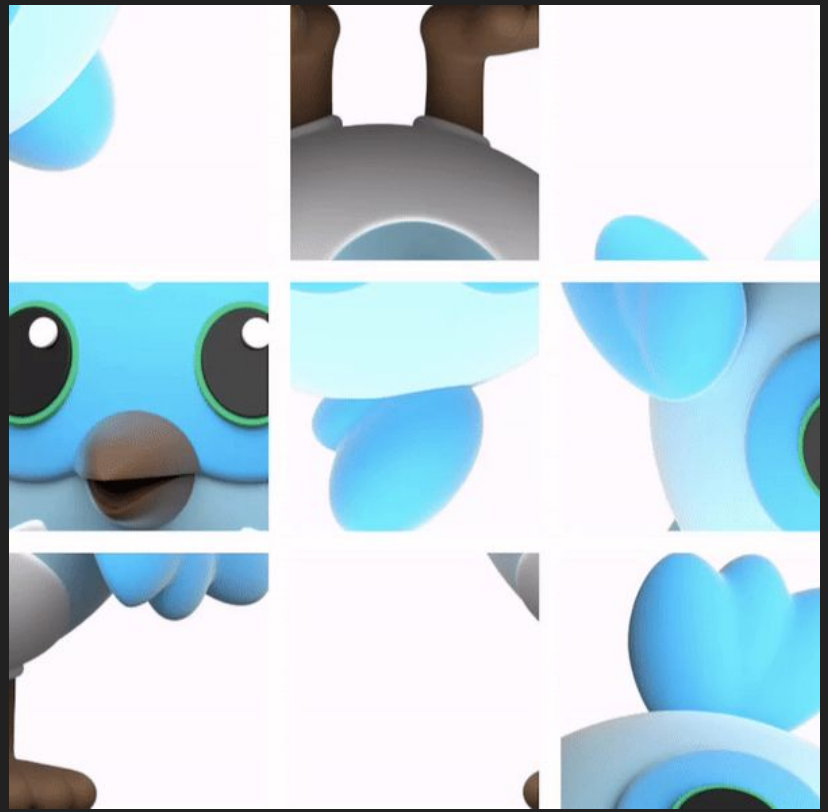
```
@override
Rect getClip(Size size) {
    //..
}
```

```dart
final isPositiveX = point.x > 0;
final isPositiveY = point.y > 0;
final isAtBoundaryX = isPositiveX && point.x != layout.dimension - layout.partSize;
final isAtBoundaryY = isPositiveY && point.y != layout.dimension - layout.partSize;
final clipValue = crossLine.width / 2;
final top = isPositiveY ? clipValue : 0.0;
final left = isPositiveX ? clipValue : 0.0;
final bottom = isAtBoundaryY || point.y == 0 ? clipValue : 0.0;
final right = isAtBoundaryX || point.x == 0 ? clipValue : 0.0;
return Rect.fromLTRB(
    left,
    top,
    layout.partSize - right,
    layout.partSize - bottom,
);
```

```
result = Draggable(
    data: rotated,
    feedback: ClipRect(
        clipper: _FeedbackClipper(
            crossLine: widget.crossLine!,
            layout: widget.layout,
            point: widget.controller.point,
        ),
        child: feedback,
        ),
    child: result,
);
```

Кайфуем от того какие мы классные (2)





Итоги

- Узнали что такое CAPTCHA
- Сделали небесячую (надеюсь) CAPTCHA с нуля голым Flutter
- Узнали как можно пользоваться RenderObject, Draggable/DragTarget, ChangeNotifier, Random, CustomPainter, CustomClipper и воображением *спанчбоб_с_радугой.png*
- Убедились, что мы не роботы (или?)
- Спасли мир от Skynet
- Сделали свой пакет!

flutter_captcha 1.0.0

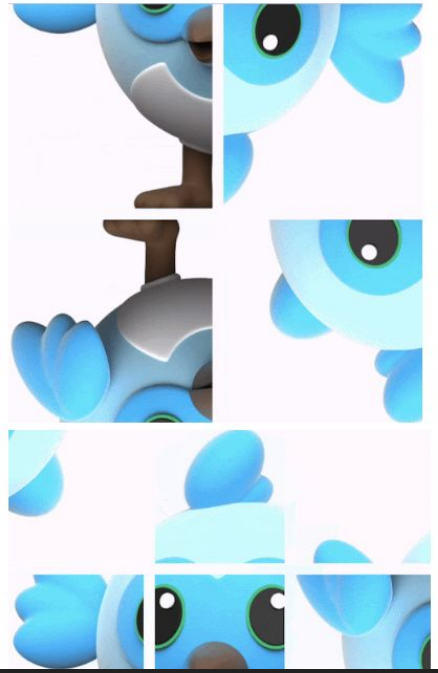
Published in the last hour • feduke-nukem.dev Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS 0

Readme Changelog Example Installing Versions Scores Admin Activity log

codecov 87%

Flutter Captcha



0 LIKES | 130 PUB POINTS | 0% POPULARITY

Publisher

feduke-nukem.dev

Metadata

Flutter captcha.

Homepage
Repository (GitHub)
View/report issues
Contributing

Documentation

API reference

License

MIT (LICENSE)

Dependencies

flutter

More

Packages that depend on flutter_captcha

Планы

Добавить возможность генерировать рандомные шумы, чтобы бота было еще сложнее разобрать изображение

Добавить разные режимы трансформаций кусочков капчи (например вращение по барабану по горизонтали/вертикали)

Спасибо за внимание