True Engineering

# Анимация, состояния, переходы

# TRANSFORM-преобразование элементов

- transform : list<Transform> — преобразования для применения
- Translate — сдвинуть элемент

  x, y : real — вектор сдвига
- Scale — масштабировать элемент

  xScale, yScale : real — коэффициенты масштабирования

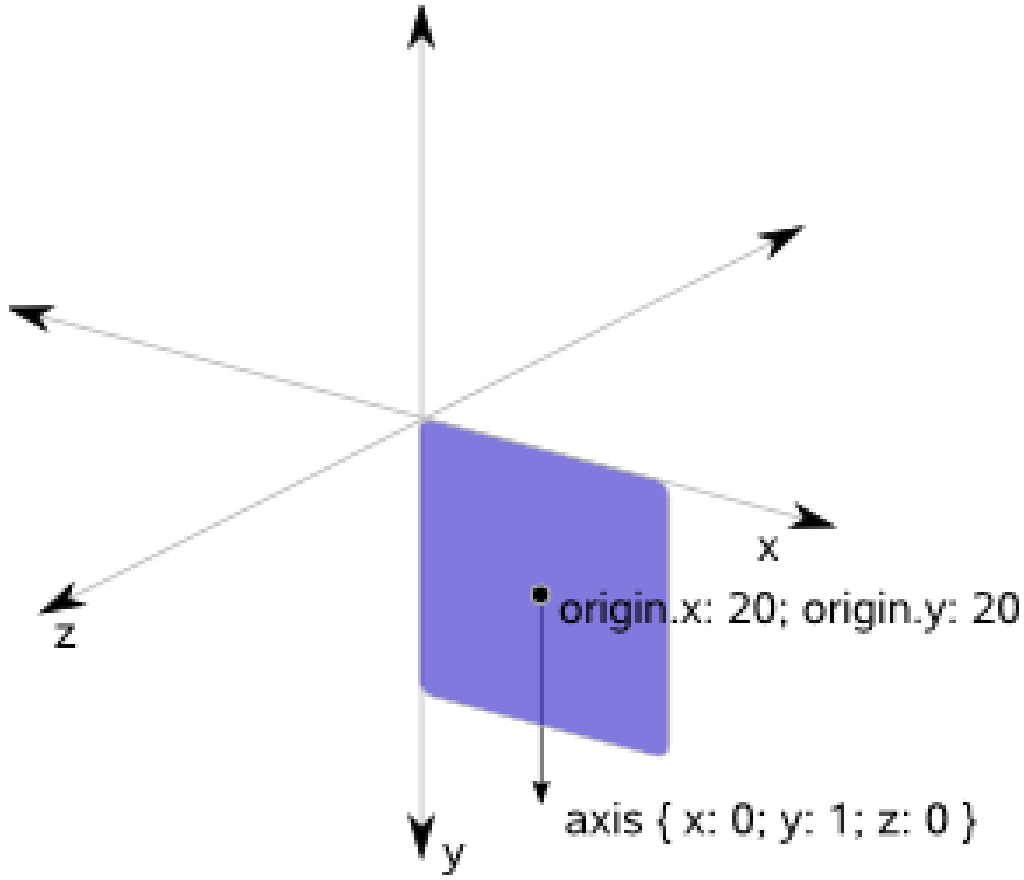
  origin.x, origin.y : real — точка отсчёта
- Rotation — повернуть элемент

  angle : real — угол поворота

  axis.x, axis.y, axis.z : real — оси поворота

  origin.x, origin.y : real — неподвижная точка

# TRANSFORM-преобразование элементов



# TIMER-периодические сигналы

- interval : int — интервал между событиями в миллисекундах
- repeat : bool — повторяются ли сигналы
- running : bool — true для старта, false для остановки
- triggeredOnStart : bool — сигнал сразу после старта
- triggered() — сигнал завершения времени
- restart() — сброс к началу
- start() — запуск
- stop() — остановка

# Трансформация с таймером

```
Image {
    source: "avrora.svg"
    transform: [
        Scale { xScale: 1.5; yScale: 2 },
        Rotation {
            id: rotation
            angle: 60
            axis { x: 1; y: 1; z: -1 }
            origin { x: 300; y: 300 }
        },
        Translate { x: 100; y: 200 }
    ]
}
Timer {
    running: true; interval: 16; repeat: true
    onTriggered: rotation.angle += 2
}
```

# Анимации-изменения свойств

- loops : int — сколько раз проигрывать анимацию
- paused : bool — на паузе ли анимация
- running : bool — воспроизводится ли анимация
- alwaysRunToEnd : bool — всегда завершать анимацию полностью, даже если был сигнал остановки
- started() — сигнал начала анимации
- stopped() — сигнал завершения анимации
- complete() — завершить анимацию полностью
- pause(), resume() — пауза, продолжение
- restart() — сброс к началу
- start(), stop() —начать, остановить

# Способы использования анимаций

- Анимация по свойствам

  Выполняется автоматически после полной загрузки элемента

- Поведение по свойствам

  Выполняется автоматически при изменении значения свойства

- Автономная анимация

  Явно запускается с использованием функции start()

  Или свойству running устанавливается значение true

# Анимации и переходы

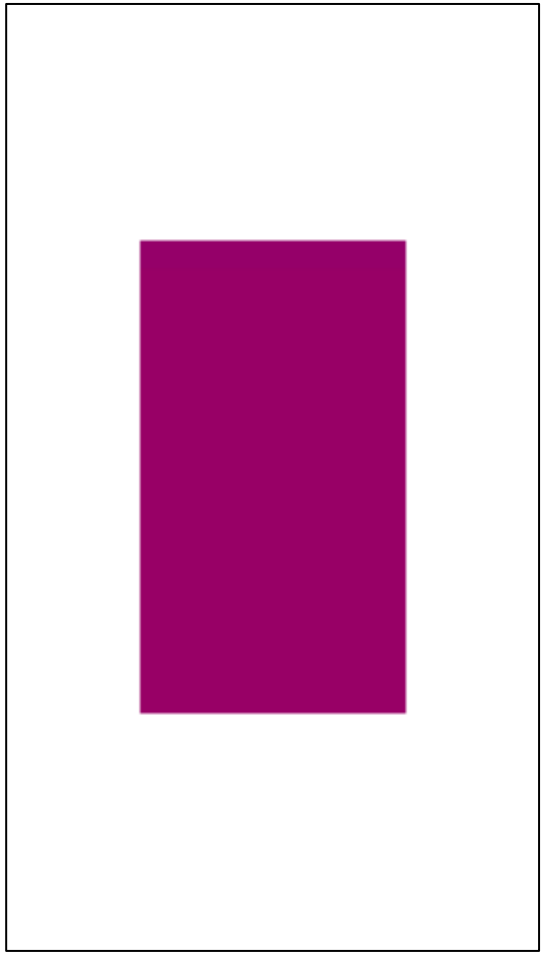
- Transition — анимируется переход между состояниями
- SequentialAnimation — последовательные анимации
- ParallelAnimation — параллельные анимации
- Behavior — анимация изменения свойств
- PropertyAction — изменение свойств без анимации
- PauseAnimation — пауза в анимации
- SmoothedAnimation — сглаживание анимации перехода
- SpringAnimation — пружинная анимация
- ScriptAction — выполнять скрипт во время анимации

# Анимации на основе типов данных

- AnchorAnimation — изменения якорей
- ParentAnimation — изменение родителя
- PathAnimation — перемещать элемент вдоль пути
- ColorAnimation — изменение цвета
- NumberAnimation — изменения чисел
- PropertyAnimation — изменения свойств
- RotationAnimation — повороты
- Vector3dAnimation — изменения значений QVector3d

# Анимации изменения свойства

```
MouseArea {
    anchors.fill: parent
    onClicked: colorAnimation.start()
}
Rectangle {
    id: rectangle
    anchors.centerIn: parent
    width: parent.width / 2; height: parent.height / 2
    color: "red"
}
PropertyAnimation {
    id: colorAnimation
    target: rectangle
    properties: "color"
    to: "blue"
    duration: 2000
}
```

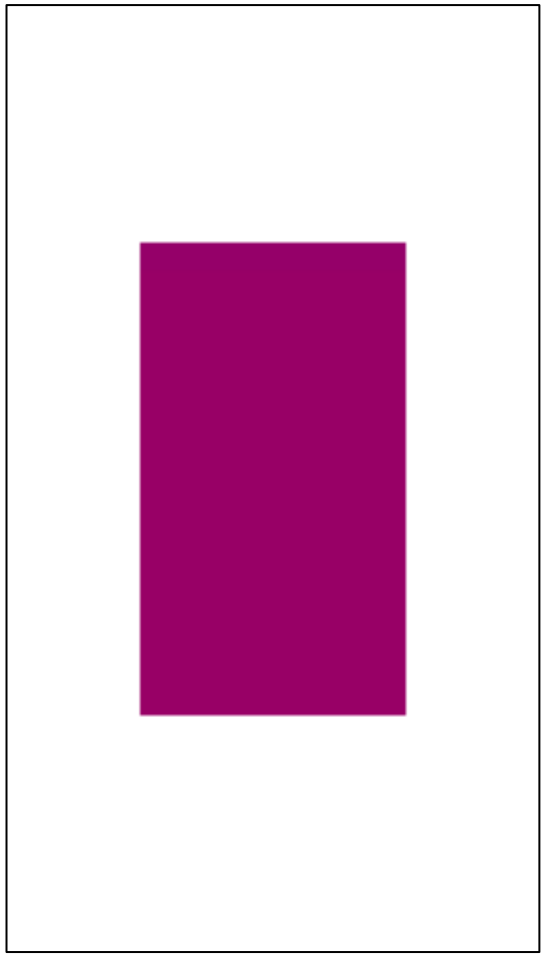


# Пример анимации Behavior

```
MouseArea {
    anchors.fill: parent
    onClicked: rectangle.color = "blue"
}


Rectangle {
    id: rectangle
    anchors.centerIn: parent
    width: parent.width / 2
    height: parent.height / 2
    color: "red"

    Behavior on color {
        ColorAnimation { duration: 2000 }
    }
}
```



# Управление анимациями

- PropertyAction — немедленные изменения свойств во время анимации
- ScriptAction — скрипты, которые будут выполняться во время анимации

# Выполнение набора анимаций

## Параллельное

```
Rectangle {
    id: rect
    width: 100; height: 100
    color: "red"

    ParallelAnimation {
        running: true

        NumberAnimation {
            target: rect; property: "x";
            to: 50; duration: 1000 }
        NumberAnimation {
            target: rect; property: "y"
            to: 50; duration: 1000 }
    }
}
```

## Последовательное

```
Rectangle {
    id: rect
    width: 100; height: 100
    color: "red"

    SequentialAnimation {
        running: true

        NumberAnimation {
            target: rect; property: "x"
            to: 50; duration: 1000 }
        NumberAnimation {
            target: rect; property: "y"
            to: 50; duration: 1000 }
    }
}
```

# Состояния и управление свойствами

- states : list<State> — возможные состояния
- State — конфигурации объектов и свойств

  name : string — название состояния

  extend : string — расширяемое состояние

  when : bool — когда наступает состояние

  changes : list<Change> — изменения для прихода в состояние
- PropertyChanges — значения свойств для состояний

  target : Object — объект для изменения свойств

  explicit : bool — задать свойства один раз вместо привязок

  restoreEntryValues : bool — восстановить предыдущие значения при выходе из состояния

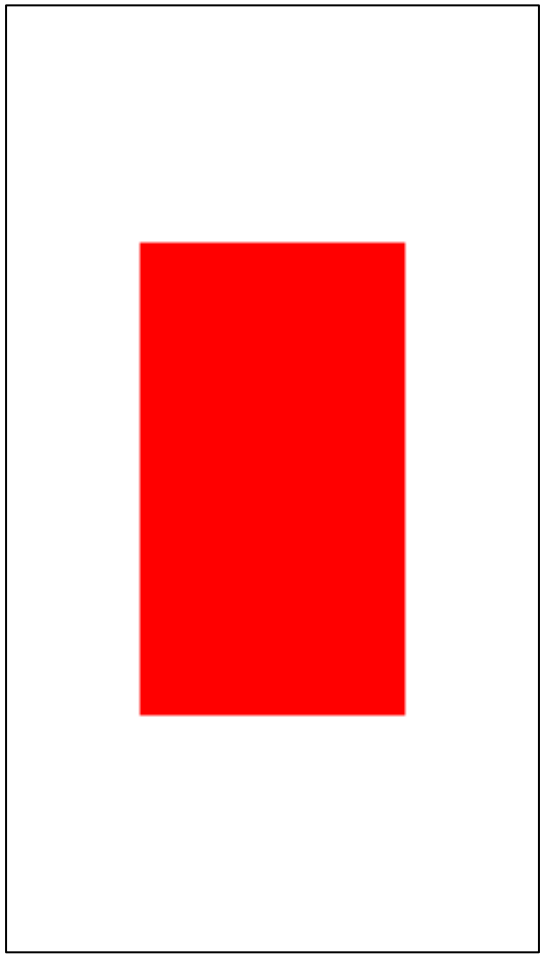
  state : string — название текущего состояния

# Пример PropertyChanges

```
MouseArea { id: mouseArea }

Rectangle {
    id: rectangle
    color: "red"
    states: State {
        name: "blue"
        when: mouseArea.pressed

        PropertyChanges  {
            target: rectangle
            color: "blue"
        }
    }
}
```



# Пример Transitions

```
MouseArea { id: mouseArea }
Rectangle {
    id: rectangle
    color: "red"
    states: State {
        name: "blue"; when: mouseArea.pressed

        PropertyChanges  { target: rectangle; color: "blue" }
    }
    transitions: [
        Transition {
            to: "blue"; ColorAnimation { duration: 2000 }
        },
        Transition {
            from: "blue"; ColorAnimation { duration: 500 }
        }
    ]
}
```

True Engineering

# Создание QML-компонентов

# Создание собственных QML компонентов

- Возможность переиспользования кода
- Исключение дублируемости кода
- Повышение читаемости кода
- Облегчение поддержки кода

# Создание собственных QML компонентов

- **TypeName.qml**

  Буквенно-цифровые символы или подчеркивания

  Название с заглавной буквы
- Импорт из другой директории

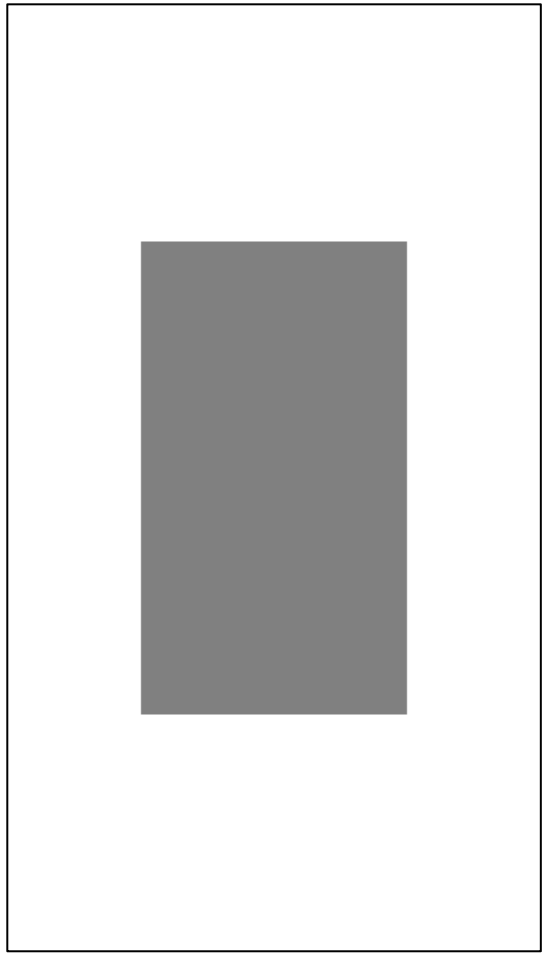
  **import "../assets"**

1. Создать **TypeName.qml**
2. Описать компонент
3. Импортировать путь к файлу
4. Использовать тип **TypeName**

# Пример собственной кнопки

```
//assets/Button.qml
import QtQuick 2.6
Rectangle {
    id: root
    signal clicked()
    MouseArea { anchors.fill: parent
                onClicked: root.clicked() }
}


//pages/ButtonPage.qml
import "../assets"
…
Button {
    anchors.centerIn: parent
    width: parent.width / 2; height: parent.height / 2
    color: "gray"
    onClicked: console.log("Button clicked")
}
```



# Пример использования контрола

```
// CustomButton.qml
Rectangle {
  Width: 100
  height: 100
  color: "red"

  MouseArea {
    anchors.fill: parent
    onClicked: console.log("Clicked!")
  }
}
```

```
// application.qml
Column {
  CustomButton { }

  Button {
    x: 100
    width: 100; height: 50
    color: "blue"
  }
}
```

# Тип Component

- progress — прогресс загрузки компонента
- status — статус загрузки компонента

  Component.Null — нет данных для компонента

  Component.Ready — компонент загружен и может использоваться

  Component.Loading — компонент загружается в настоящий момент

  Component.Error — ошибка при загрузке компонента
- url — URL-адрес, который использовался для создания компонента.
- completed() — испускается после создания объекта
- destruction() — испускается, когда объект начинает уничтожаться

# Примеры Component

```
Item {
    Component {id: redSquare
        Rectangle { color: "red" }
    }
    Loader { sourceComponent: redSquare }
    Loader { sourceComponent: redSquare; x: 20 }
}
Rectangle {
    Component.onCompleted:
        console.log("Completed Running!")
    Rectangle { Component.onCompleted:
            console.log("Nested Completed Running!")
    }
}
Rectangle {
    Component.onDestruction:
        console.log("Destruction Beginning!")
    Rectangle { Component.onDestruction:
            console.log("Nested Destruction Beginning!")
    }
}
```

# Свойства с параметром Default

- Определение свойства с default делает его свойством по умолчанию

```
// MyLabel.qml
import QtQuick 2.0
Text {
  default property var someText
  text: "Hello, " + someText.text
}
```

- Значение свойству *someText* может быть присвоено так

```
// application.qml
MyLabel { Text { text: "world!" } }
```

- Что аналогично обычному присвоению

```
// application.qml
MyLabel { someText: Text { text: "world!" } }
```

# Alias

[default] property alias <name>: <id.property>

```
// Button.qml
import QtQuick 2.0


Item {
  property alias buttonText: textItem.text
  width: 200; height: 50
  Text { id: textItem }
}
```

# Создание компонента

- Элемент Component с вложенным объектом

```
Component {
    id: buttonComponent


    Button { }
}
```

- Внешний документ QML

```
var buttonComponent = Qt.createComponent("../assets/Button.qml");
if (buttonComponent.status === Component.Ready)
    buttonComponentReady();
else
    buttonComponent.statusChanged.connect(buttonComponentReady);
```

# Loader-загрузчик динамических элементов

- source : url — ссылка на внешний компонент для загрузки
- sourceComponent : Component — компонент для загрузки
- status : enumeration — состояние загрузки

    Loader.Null — загрузчик неактивен или не указан компонент для загрузки

    Loader.Ready — компонент загружен

    Loader.Loading — компонент загружается

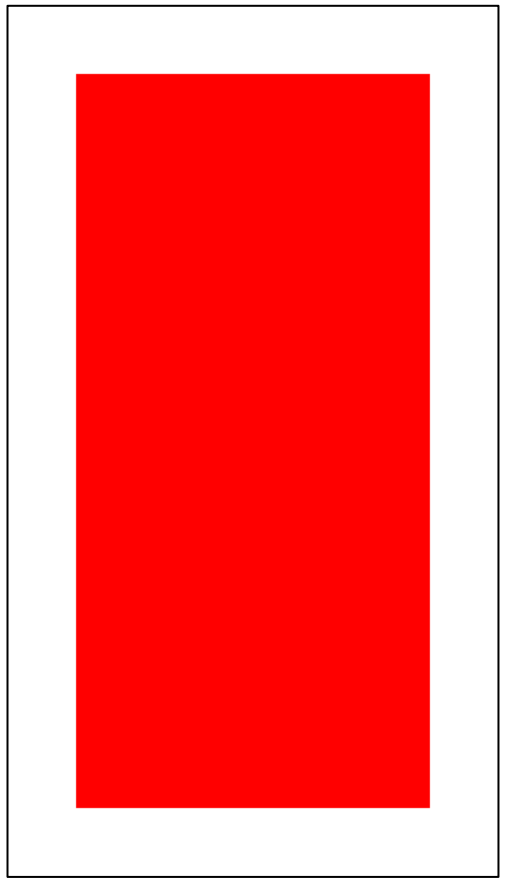
    Loader.Error — ошибка при загрузке компонента

# Пример динамической загрузки

```
Loader {
    id: loader
    anchors{ fill: parent; margins: 100 }
}


MouseArea {
    anchors.fill: parent
    onClicked: loader.source = "RedRectangle.qml"
    onDoubleClicked:
        loader.sourceComponent = blueRectangle
}


Component {
    id: blueRectangle
    Rectangle { color: "blue" }
}
```



True Engineering

**True Engineering**

630128, г. Новосибирск,
ул. Кутателадзе, 4г

(383) 363-33-51, 363-33-50
info@trueengineering.ru
trueengineering.ru

# Новосибирский Государственный Университет