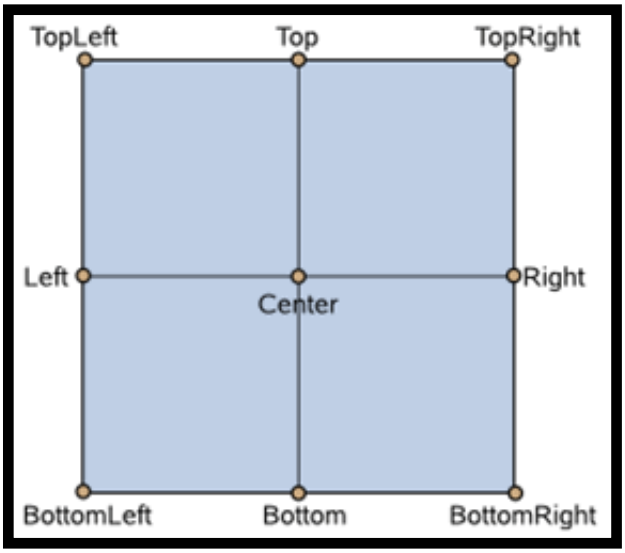
Introducción a QML - V

Transformaciones (Item transformations)

Los elementos soportan varias formas de transformación:

- Rotación (**rotation**): En torno al eje z y en grados
- Escala (**scale**): smaller
- Ambas son relativas a **transformOrigin**. Están disponibles nueve orígenes de transformación. El valor por defecto es **Item.Center**.



- Existe la posibilidad de crear listas de transformación

```
Item {
  transitions: [
    Transition { ... },
    Transition { ... }
    ...
  ]
}
```

- Soporte para Rotación en 3D
- Soporte para posición en el eje de forma aleatoria (x, y, z)
- Puedes tener factores de escala separados para el eje **x** e **y**
- Podemos trasladar un objeto (**Translate**) sin que afecte a su posición **x** e **y**

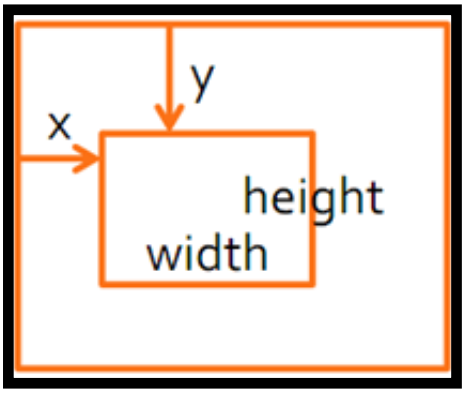
Diseños de los elementos (Item layouts)

El diseño de los elementos depende fundamentalmente de:

- Las coordenadas relativas
- Los anclajes entre elementos
- El posicionamientos de los objetos
- De la forma de posicionar los elementos: Modo fila (**Row**), Columna (**Column**), Flujo (**Flow**) y rejilla (**Grid**) en la que se ubiquen

Coordenadas de los elementos (Item coordinates)

- La posición de un elemento se define mediante **x** e **y**. Es relativa al elemento padre.
- El tamaño se define mediante **width** y **height**




```
Rectangle {
    id: parentRect
    color: "yellow"
    x: 50; y: 50; width: 50; height: 50
    Rectangle {
        id: childRect
        color: "green"
        x: 35; y: 35; width: 50; height: 50
    }
}
```

  - z define el orden de solapamiento (overlapping) de las áreas que estás dibujando.

Same z - later children above earlier children:

```
Item {
    Rectangle {
        color: "red"
        width: 100; height: 100
    }
    Rectangle {
        color: "blue"
        x: 50; y: 50; width: 100; height: 100
    }
}
```

Higher z on top:

```
Item {
    Rectangle {
        z: 1
        color: "red"
        width: 100; height: 100
    }
    Rectangle {
        color: "blue"
        x: 50; y: 50; width: 100; height: 100
    }
}
```

Same z - children above parents:

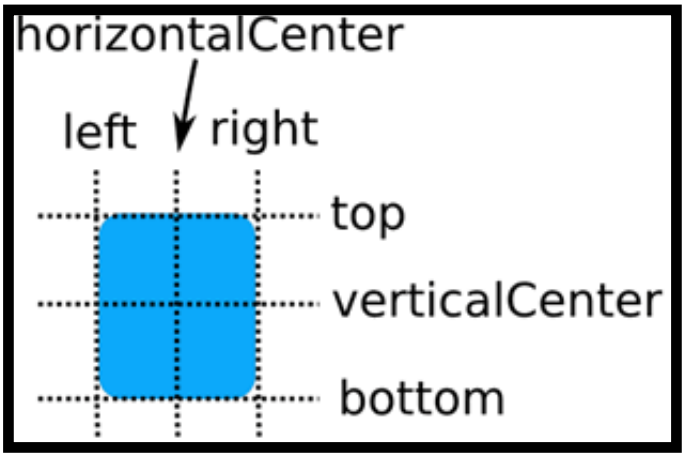
```
Item {
    Rectangle {
        color: "red"
        width: 100; height: 100
        Rectangle {
            color: "blue"
            x: 50; y: 50; width: 100; height: 100
        }
    }
}
```

Lower z below:

```
Item {
    Rectangle {
        color: "red"
        width: 100; height: 100
        Rectangle {
            z: -1
            color: "blue"
            x: 50; y: 50; width: 100; height: 100
        }
    }
}
```

Anclajes de los elementos (Item anchors)

Es la forma de posicionar un elemento respecto a otro.

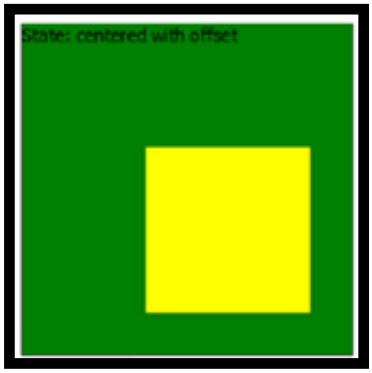


Cada elemento tiene seis líneas de anclaje:

- Anclajes básicos: **top**, **bottom**, **left**, **right**

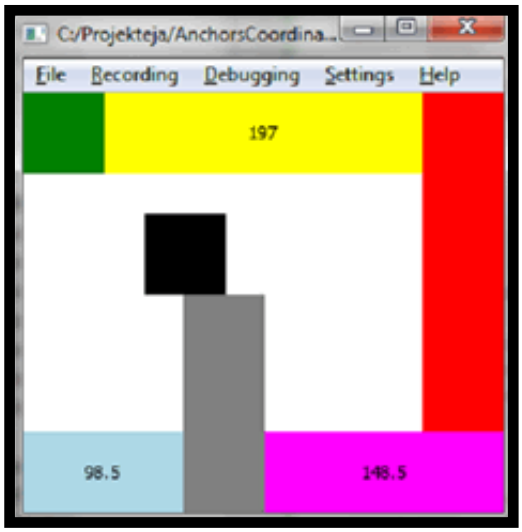
```
Rectangle {
    id: rectangle2
    color: "blue"
    anchors.left: myPic.right
    anchors.right: parent.right
    anchors.bottom: parent.bottom
    anchors.top: parent.top
    anchors.leftMargin: 5
```

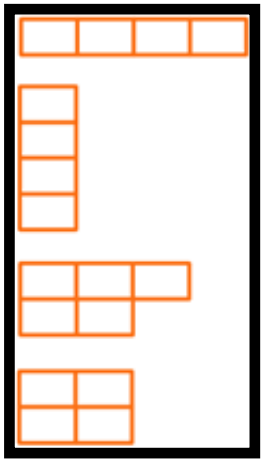
  - Anclaje para centrado: **verticalCenter**, **horizontalCenter**
  - Existe un anclaje especial para rellenado y otro para texto
  - Los anclajes disponen de modos de espaciarse mediante márgenes (**topMargin**, **bottomMargin**, **leftMargin**, **rightMargin**).
  - Los anclajes centrados soportan desplazamiento mediante offset (**verticalCenterOffset**, **horizontalCenterOffset**)



Algunas reglas a tener en cuenta con los anclajes (Anchors) y las coordenadas (Coordinates)

- Un elementos solo se puede anclar respecto al padre o sus hermanos.
- Los Anclajes siempre se sobreponen a **x** e **y**
- El ancho (**width**) o alto (**height**) son necesarios con anclajes simples
- El ancho (**width**) o alto (**height**) son sobrescritos cuando ambas caras han sido ancladas





Posicionar a los elementos (Positioners)

Existen cuatro formas de posicionar:

1. Por fila (**Row**). Para colocar elementos horizontalmente.
2. Por columna (**Column**). Para colocar elementos verticalmente.
3. Por flujo horizontal o vertical (**Flow**). Soporta salto (**wrapping**)
4. Por Rejilla (**Grid**) de dos dimensiones

Bueno…. ya estamos preparados para el primer ejercicio… en el próximo post crearemos nuestro primer ejemplo QML.