Drawing

Unknown Author

February 26, 2014

In [1]:
```
from subprocess import Popen, STDOUT, PIPE
import sys
import os
import re

def qml_magic(args, cell):
    """Execute a block of QML code using qmlscene.

    This requires `qmlscene` to be available in your path. `qmlscene`
    does not read QML code from stdin, so this will create a temporary
    file that contains the cell's QML code. This file is deleted after
    `qmlscene` exits. All error messages referring to this file are
    redacted so they don't show the file name any more.

    All output of the QML program is printed in real time.

    If you want to pass additional arguments to `qmlscene` just append
    them to the `%%qml` magic. For instance, `%%qml -h` will print the
    `qmlscene` help text instead of executing your code.

    """
    filename = '.ipython_temp.qml'
    with open(filename, 'w') as f:
        f.write(cell)
    process = Popen(['qmlscene']+args.split()+[filename],
                    stdout=PIPE, stderr=STDOUT, universal_newlines=True)
    while True:
        nextline = process.stdout.readline()
        if not nextline:
            break
        # replace temp file url with line number
        sys.stdout.write(re.sub('file:.*{}:([0-9]+)'.format(filename),
                                r'line \1:', nextline))
        sys.stdout.flush()

    os.remove(filename)

get_ipython().register_magic_function(qml_magic, "cell", "qml")
```

In [2]:
```
%%javascript
// this switches syntax highlighting for %%qml cells to javascript
IPython.config.cell_magic_highlight['magic_javascript']['reg'].push(/^%%qml/)
IPython.config.cell_magic_highlight['magic_javascript']['reg'].push(/^%%writefile .*qm
```

```
<IPython.core.display.Javascript at 0x1106c3a90>
```

In [4]:
```
%%qml
import QtQuick 2.2
import QtQuick.Controls 1.1

ApplicationWindow {
    Canvas {
```

```
        implicitHeight: 100
        implicitWidth: 100
        anchors.centerIn: parent
        contextType: "2d"
        onPaint: {
            context.fillStyle = "red"
            context.fillRect(0, 0, width, height)
        }
    }
}
```

In [5]:

```
%%qml
import QtQuick 2.2
import QtQuick.Controls 1.1

ApplicationWindow {
    Canvas {
        implicitHeight: 100
        implicitWidth: 100
        anchors.centerIn: parent
        contextType: "2d"
        onPaint: {
            context.moveTo(0, 0)
            context.beginPath()
            context.lineTo(width, 0)
            context.lineTo(width/2, height)
            context.lineTo(0, 0)
            context.closePath()
            context.fillStyle = "green"
            context.fill()
        }
    }
}
```

In [15]:

```
%%qml
import QtQuick 2.2
import QtQuick.Controls 1.1
import QtQuick.Layouts 1.1

ApplicationWindow {
    ColumnLayout {
        anchors.fill: parent
        Slider {
            id: slider
            minimumValue: 0
            maximumValue: 1
            stepSize: 0.01
            Layout.fillWidth: true
        }
        Canvas {
            property real size: slider.value
            onSizeChanged: requestPaint()
            implicitHeight: 100
            implicitWidth: 100
            Layout.alignment: Qt.AlignVCenter | Qt.AlignHCenter
            contextType: "2d"
            onPaint: {
                context.fillStyle = "lightBlue"
                context.fillRect(0, 0, width, height)
                context.moveTo(0, 0)
                context.beginPath()
                context.lineTo(size*width, 0)
                context.lineTo(size*width/2, size*height)
                context.lineTo(0, 0)
                context.closePath()
                context.fillStyle = "red"
```

```
                context.fill()
            }
        }
    }
}
```

In [17]:

```
%%qml
import QtQuick 2.2
import QtQuick.Controls 1.1
import QtQuick.Layouts 1.1

ApplicationWindow {
    ColumnLayout {
        anchors.fill: parent
        Slider {
            id: slider
            minimumValue: 0
            maximumValue: 1
            stepSize: 0.01
            Layout.fillWidth: true
        }
        Canvas {
            property real size: slider.value
            onSizeChanged: requestPaint()
            implicitHeight: 100
            implicitWidth: 100
            Layout.alignment: Qt.AlignVCenter | Qt.AlignHCenter
            contextType: "2d"
            onPaint: {
                context.fillStyle = "lightBlue"
                context.fillRect(0, 0, width, height)
                context.save()
                context.translate(width/2, height/2)
                context.scale(width/2, height/2)

                context.moveTo(-size, -size)
                context.beginPath()
                context.lineTo(size, -size)
                context.lineTo(0, size)
                context.lineTo(-size, -size)

                context.closePath()
                context.fillStyle = "red"
                context.fill()
                context.restore()
            }
        }
    }
}
```

In [25]:

```
%%qml -I .
import QtQuick 2.2
import QtQuick.Controls 1.1
import QtQuick.Layouts 1.1
import ClockTime 1.0

ApplicationWindow {
    Time {
        id: time
    }
    ColumnLayout {
        anchors.fill: parent
        Canvas {
            property real second: time.second
            onSecondChanged: requestPaint()
            implicitHeight: 100
```

```
            implicitWidth: 100
            contextType: "2d"
            Layout.alignment: Qt.AlignVCenter | Qt.AlignHCenter
            onPaint: {
                context.clearRect(0, 0, width, height)
                context.save()

                context.translate(width/2, height/2)
                context.rotate(second/60*2*Math.PI + Math.PI)

                context.beginPath()
                context.moveTo(0, 0)
                context.lineTo(0, height/2)
                context.closePath()
                context.strokeStyle = "red"
                context.stroke()

                context.restore()
            }
        }
        Text {
            text: "%1:%2:%3".arg(time.hour).arg(time.minute).arg(time.second)
        }
    }
}
```

In [30]:

```
%%qml -I .
import QtQuick 2.2
import QtQuick.Controls 1.1
import QtQuick.Layouts 1.1
import ClockTime 1.0

ApplicationWindow {
    Time {
        id: time
    }
    ColumnLayout {
        anchors.fill: parent
        Canvas {
            property real second: time.second
            onSecondChanged: requestPaint()
            property real minute: time.minute
            onMinuteChanged: requestPaint()
            property real hour: time.hour
            onHourChanged: requestPaint()
            implicitHeight: 100
            implicitWidth: 100
            contextType: "2d"
            Layout.alignment: Qt.AlignVCenter | Qt.AlignHCenter
            onPaint: {
                context.clearRect(0, 0, width, height)
                context.save()
                context.translate(width/2, height/2)

                function drawHand(angle, begin, end) {
                    context.save()
                    context.rotate(angle + Math.PI)
                    context.beginPath()
                    context.moveTo(0, begin)
                    context.lineTo(0, end)
                    context.closePath()
                    context.stroke()
                    context.restore()
                }

                context.lineJoin = "bevel"
```

```
                context.strokeStyle = "black"
                context.lineWidth = 1
                for (var h=0; h<12; h++) {
                    drawHand(h/12*2*Math.PI, height/2*0.8, height/2)
                }

                context.strokeStyle = "black"
                context.lineWidth = 3
                drawHand(hour/12*2*Math.PI, 0, height/2*0.6)

                context.strokeStyle = "black"
                context.lineWidth = 2
                drawHand(minute/60*2*Math.PI, 0, height/2)

                context.strokeStyle = "red"
                context.lineWidth = 1
                drawHand(second/60*2*Math.PI, 0, height/2)


                context.restore()
            }
        }
        Text {
            text: "%1:%2:%3".arg(time.hour).arg(time.minute).arg(time.second)
        }
    }
}
```

In [34]:

```
%%qml -I .
import QtQuick 2.2
import QtQuick.Controls 1.1
import QtQuick.Layouts 1.1
import ClockTime 1.0

ApplicationWindow {
    Time {
        id: time
        onSecondChanged: PropertyAnimation {
            target: clock
            property: "second"
            to: time.second
            duration: time.second === 0 ? 0 : 300
            easing.type: Easing.OutElastic
        }
    }
    ColumnLayout {
        anchors.fill: parent
        Canvas {
            id: clock
            property real second
            onSecondChanged: requestPaint()
            property real minute: time.minute
            onMinuteChanged: requestPaint()
            property real hour: time.hour
            onHourChanged: requestPaint()
            implicitHeight: 100
            implicitWidth: 100
            contextType: "2d"
            Layout.alignment: Qt.AlignVCenter | Qt.AlignHCenter
            onPaint: {
                context.clearRect(0, 0, width, height)
                context.save()
                context.translate(width/2, height/2)

                function drawHand(angle, begin, end) {
                    context.save()
```

```
                    context.rotate(angle + Math.PI)
                    context.beginPath()
                    context.moveTo(0, begin)
                    context.lineTo(0, end)
                    context.closePath()
                    context.stroke()
                    context.restore()
                }

                context.lineJoin = "bevel"

                context.strokeStyle = "black"
                context.lineWidth = 1
                for (var h=0; h<12; h++) {
                    drawHand(h/12*2*Math.PI, height/2*0.8, height/2)
                }

                context.strokeStyle = "black"
                context.lineWidth = 3
                drawHand(hour/12*2*Math.PI, 0, height/2*0.6)

                context.strokeStyle = "black"
                context.lineWidth = 2
                drawHand(minute/60*2*Math.PI, 0, height/2)

                context.strokeStyle = "red"
                context.lineWidth = 1
                drawHand(second/60*2*Math.PI, 0, height/2)


                context.restore()
            }
        }
        Text {
            text: "%1:%2:%3".arg(time.hour).arg(time.minute).arg(time.second)
        }
    }
}
```

In []: