Alkalmi Python – 7. rész

PCLinuxOS Magazine – 2019. augusztus

Írta: Peter Kelly (critter)

És akkor most valami teljesen más

Más, mert megmutatom hogyan készítsünk grafikai alkalmazást a Designer alkalmazása nélkül. Ehhez jön még, hogy valaki más kódjával indítok. Ez nem valami otromba plágium, ami rosszallást válthat ki és akár jogi következményei is lehetnének, hanem kód újrahasznosításról. A Python és a Qt egyaránt elég liberálisan licencelt szabad szoftver, de kereskedelmi, legalábbis a Qt, aminél a licencnek megfelelően kell eljárni. A mi használtunk rendben lesz. A kód, amit használni akarok, Qt példák között kezdte pályafutását, ám én elég erőteljesen módosítani fogom. Sőt mi több, beágyazom a példákra vonatkozó licencet is. Erre mindig ügyelj akkor, amikor más kódot vonsz be. Miért nem használom a Designer-t? Nagyon egyszerű, a tervező nem támogat néhány olyan megoldást, amit ebben az alkalmazásban el akarok érni. Valójában a Designer egyetlen dolgot támogat teljesen ez esetben, a Qwidget, amit az űrlapunk alapjaként használni fogunk.

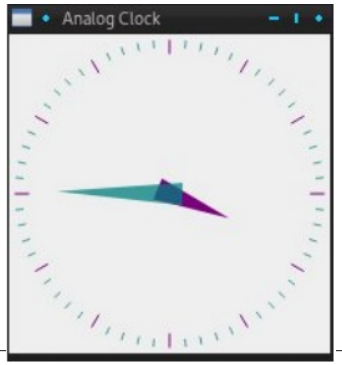
Pár éve KDE-t és alatta Cairo clock-ot használtam, ami egy jópofa analóg asztali óra. A mostani KDE-val és Plasma-val adott analóg órát egyáltalán nem szeretem és újra fel akartam tenni a Cairo clock-ot, amikor rájöttem, hogy Python-nal és Qt5-tel megcsinálhatom a sajátomat és talán fejleszthetek egy kicsit rajta. Ez lesz a következő projektünk.

Az asztali órák általában mindig futnak, ezért valószínűleg nem a rendszertálcán jelenítjük meg. Ez csak néhány asztal esetében jelentkező probléma, ám ha téged érint, erre az esetre bemutatok egy kódot megoldásként. Ám, ha a kódot alkalmazni akarod, akkor a tárolóból telepítened kell a wmctrl-t (a nálam futó openbox esetében telepítenem kellett, de nálad esetleg nem lehet szükség rá).

```
#############################################################################
## Copyright (C) 2013 Riverbank Computing Limited.
## Copyright (C) 2010 Nokia Corporation and/or its subsidiary(-ies).
## All rights reserved.
##
## This file is part of the examples of PyQt.
##
## $QT_BEGIN_LICENSE:BSD$
## You may use this file under the terms of the BSD license as follows:
##
## "Redistribution and use in source and binary forms, with or without
## modification, are permitted provided that the following conditions are
## met:
##  * Redistributions of source code must retain the above copyright
##  notice, this list of conditions and the following disclaimer.
##  * Redistributions in binary form must reproduce the above copyright
##  notice, this list of conditions and the following disclaimer in
##  the documentation and/or other materials provided with the
##  distribution.
##  * Neither the name of Nokia Corporation and its Subsidiary(-ies) nor
##  the names of its contributors may be used to endorse or promote
##  products derived from this software without specific prior
##  written permission.
##
## THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
## "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
## LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
## A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
## OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
## SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
## LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
## DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
## THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
## (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
## OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE."
## $QT_END_LICENSE$
##
#############################################################################
```

Ez az eredeti licenc, amit az alapul szolgáló példa tartalmazott.

Q5_aclock_py



Ez a Qt5 minta órája, Jól működik, de néhány funkció hiányzik belőle.

Az én alap órám – átlátszatlan, a számjelzők pontok és kiegészítettem digitális dátummal és idővel, valamint egy címkével. Választani lehet a folyamatos, vagy léptető nagymutató közül is.

Óra arab számokkal, a percjelzők pálcikák és felirat. Az átlátszóság eltünteti az ablakkeretet.

Óra elforgatott római számokkal és sötét témával.

A képernyőképekkel képet alkothatsz arról, amit elérhetünk. Átméretezhető. Így lehet egy igazán NAGY órád, ha a számítógéptől távol dolgozol. Az opciók mindegyike parancssorból elérhető, amiben súgó is van.

$:> ./qt5_aclock.py -h
usage: qt5_aclock.py [-h] [-l L] [-n] [-r] [-d] [-b] [-s] [-t] [-x X] [-y Y]

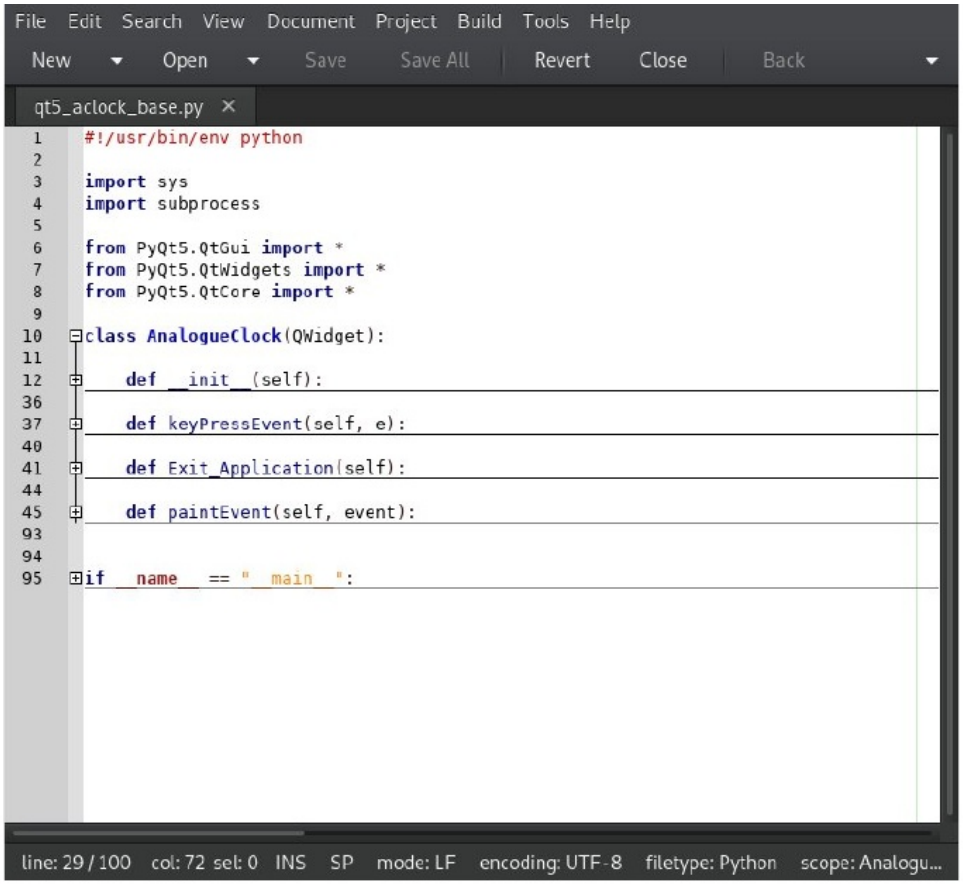
Ezzel egy analóg órát kapsz folyamatos, vagy lépegető nagymutatóval és kiválasztott számokkal. Pontok, vagy pálcikák a percjelzők, világos, vagy sötét a téma és szövegmező opcionálisan.

optional arguments:
-h, --help show this help message and exit
-l L permits customization by adding text to the clockface (maximum 12 characters). quote if spaces included (default: Qt5 Clock)
-n adds Arabic numerals to the display (default: False)
-r adds Roman numerals to the display (default: False)
-d apply the dark theme (default: False)
-b Use batons for minute marks, default is dots (default: False)
-s changes the default stepping second hand to a sweeping motion (default: False)
-t enables transparency - compositing must be enabled and active (default: False)
-x X x startup position (default: 0)
-y Y y startup position (default: 0)

© Casualsoft 2018

Ez a kód elég hosszú, majdnem 500 sor a teljes verzió. Egy elég alap változattal indítok és amikor már fut, ruházom fel további tulajdonságokkal. A legtöbb alkalmazást így fejlesztik.

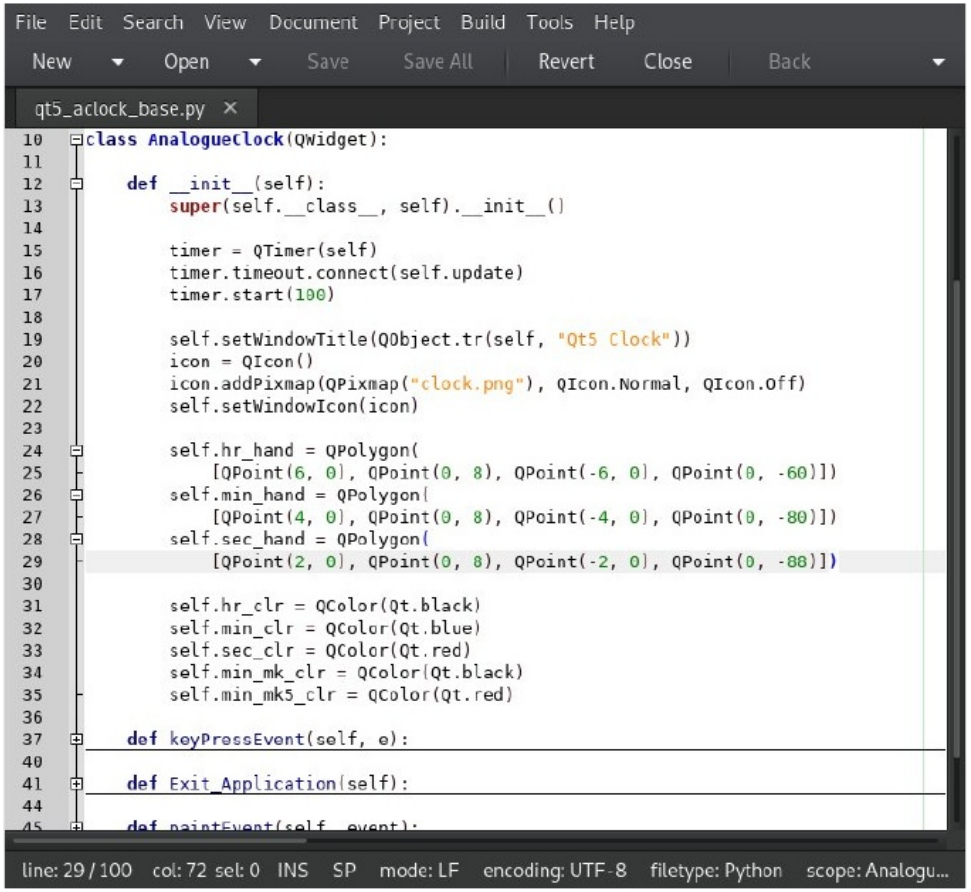
Íme a csökkentett változat kódja Geany-ben, a definíciók összecsukva (ezt a margón található kis négyzetekre kattintással teheted meg, Segít koncentrálni a kód aktuálisan foglakoztató részére). Most sokkal kevésbé ijesztően néz ki.



Ahogy azt korábban jeleztem, az órát általában úgy állítják be, hogy bejelentkezéskor induljon el. Ezért készítettem parancssori felületet hozzá, amivel az összes opcióval ellátott parancsot lehet írni hozzá, ha kell, és az kiadható automatikus indításra. Lehet még készíteni egy qt5_aclock.desktop fájlt is az opciókkal, ahogy azt az alkalmazáskereső program készítésekor tárgyaltuk. Ha egyik opcióval sem kívánsz élni, akkor futtasd simán azok nélkül. További lehetőség, opciókat alapbeállításként tartalmazó verzió készítése. Végül is az alkalmazás készítésének ez a célja – valami olyasmit kapni, ami pontosan az elvárásaidnak megfelelően működik. Egyszerűen telepíthetnéd a Cairo clock-ot, de ez a te gyermeked. Személyes dolog.

A képen látható kódot elnézve az egyetlen dolog, ami az előző példához képest változott az az osztály elnevezése és az új paintEvent eljárás. Emellett nincs felhasználói felület importálás Emellett nem történik felhasználói felület-modul importálás, mivel nem használtuk a Designer-t, hogy készítsünk egyet, Emiatt az osztály definíció csak a QWidget osztályt hívja meg a zárójelei között. Ebben a fázisban csak az adott widget attribútumaival és eljárásaival rendelkezik.

Az egyetlen változásnak valójában csak a __init__ eljárás és az új paintEvent eljárás bizonyul. KeyPressEvent és az exit_Application pontosan megegyezik a korábbi példákban találhatókkal. Még a 95. sor utáni kódban sem található meglepetés.



Íme az __init__ eljárás kibontva

A 15-17 sorok meghívnak egy QTimer (időzítő) objektumot, ami a timer változóhoz lett rendelve. A QTimer objektum folyamatosan lejár minden intervallumban és kiad egy, a QWidgets frissítő eljáráshoz kapcsolt jelzést. A 07. sor elindítja az időzítőt 100 milliszekundumos intervallummal . Ez annyit tesz, hogy az osztály másodpercenként 10-szer frissül.

19-22. sor állítja be az ablak címét és az ikont.

A 24-29. sor alakítja ki a három mutató formáját. A Qt tudja, hogyan kell három, vagy több oldalú alakzatok rajzolni a QPolygon eljárása segítségével. Az eljárás minden sarokponthoz x és y koordinátákat igényel és ezeket külön a feladatra tervezett QPoint objektumok formájában kapja meg, A QPoint két integert vár, nem lebegőpontos tizedes számokat. (A hivatalos Qt dokumentáció szerint: „A QPoint osztály egy pontot határoz meg a síkban az integerek pontosságát alkalmazva.”)

A kép jobb oldalánál láthatsz egy vékony kék függőleges vonalat. Ezt a vonalat a 80. oszlopba helyeztem el, ami a kód szövegének megegyezés szerinti maximális sorszélessége. Ez a konvenció azokra az időkre vezethető vissza, amikor a terminálok legfeljebb 80 oszlopot jeleníthettek meg. Ha megmaradunk a konvenciónál, akkor mindenki, aki olvassa a kódot elégedett lesz. Semmi rossz nem történik, ha túlléped a korlátot. Ahhoz, hogy a sorok illeszkedjenek, az első zárójelet követően felosztottam azokat. A Python elfogadja ezt sor folytatásaként, de ha ott kell megosztanod a sort, ahol nincs zárójel, akkor használj fordított törtjelet a sor utolsó karaktereként és folytasd a következő sorban. Sorok tördelésekor ügyelj az olvashatóságra. Valami ilyesmit is csinálhattam volna:

```
self.hr_hand = QPolygon([QPoint(6, 0),
                         QPoint(0, 8),
                         QPoint(-6, 0),
                         QPoint(0, -60)])
```

A Python számára ez ugyanaz.

31-35. sor határozza meg a három mutató színét, a percjelzőket és az ötperces jelzőket. Én szabványos szín elnevezéseket használtam, legtöbb konvenció elfogadott, mint a '#000ff00' a zöldhöz (az idézőjel szükséges). Ezzel az __init__ eljárás készen van.

A következő két eljárás nem tér el a korábban már alkalmazottaktól.

```
    def keyPressEvent(self, e):
        if e.key() == Qt.Key_Escape:
            self.Exit_Application()

def Exit_Application(self):
    self.close()
        sys.exit()
```

A paintEvent eljárás az, ahol a dolgok java történik.

qt5_aclock_base.py

```
44
45      def paintEvent(self, event):
46          side = min(self.width(), self.height())
47          time = QTime.currentTime()
48          qp = QPainter()
49          qp.begin(self)
50          qp.setRenderHint(QPainter.Antialiasing)
51          qp.translate(self.width() / 2, self.height() / 2)
52          qp.scale(side / 210.0, side / 210.0)
53
54          qp.setPen(QPen(Qt.lightGray, 6))  # draw the background
55          style = Qt.BrushStyle(Qt.SolidPattern)
56          qp.setBrush(QBrush(QColor(255, 255, 255), style))
57          qp.drawEllipse(QRect(-97, -97, 194, 194))
58
59          qp.setPen(Qt.NoPen)  # draw the hour hand
60          qp.setBrush(QBrush(self.hr_clr))
61          qp.save()
62          qp.rotate(30.0 * ((time.hour() + time.minute() / 60.0)))
63          qp.drawConvexPolygon(self.hr_hand)
64          qp.restore()
65
66          qp.setPen(QPen(self.min_mk5_clr))  # draw the 5 minute marks
67          qp.setBrush(QBrush(self.min_mk5_clr))
68          for i in range(0, 12):
69              qp.drawEllipse(90, 0, 4, 4)
70              qp.rotate(30.0)
71          qp.setPen(Qt.NoPen)
72          qp.setBrush(QBrush(self.min_clr))
73
74          qp.save()  # draw the minute hand
75          qp.rotate(6.0 * (time.minute() + time.second() / 60.0))
76          qp.drawConvexPolygon(self.min_hand)
77          qp.restore()
78          qp.setPen(Qt.NoPen)
79          qp.setBrush(QBrush(self.sec_clr))
80
81          qp.save()  # draw the second hand
82          qp.rotate(6.0 * (time.second()))
83          qp.drawConvexPolygon(self.sec_hand)
84          qp.restore()
85
86          qp.setBrush(QBrush(self.min_mk_clr))  # draw the minute marks
87          qp.setPen(QPen(self.min_mk_clr))
88          for j in range(0, 60):
89              if j % 5:
90                  qp.drawEllipse(90, 0, 2, 2)
91              qp.rotate(6.0)
92          qp.end()
```

Az órát a QWidget által adott formába jeleníti meg. Mivel kör alakú órát akarunk, ezért korlátoznunk kell, hogy a megjelenítő terület legrövidebb oldalához igazodjon méretben. Erre a szabványos min() Python függvényt alkalmazhatjuk, ami az adott argumentumok közül a legkisebb elemet adja vissza.

Az idő változót a QTime osztály currenttime (aktuális idő) eljárástól kapott érték adja, amit a rendszerórából nyer.

A QPainter osztály végzi az formába történő festést (igen, akár egy valódi festő = painter).

48. sor az osztály egy példányát a qp változóhoz rendeli.

Az összes rajzolást a QPainter begin és end eljárása közötti kódban történik.

A 49. és a 92. sorok hívják meg ezeket az eljárásokat.

Az 50. sor mondja meg a qp-nek, hogyan jelenítse meg a készített képeket. Az Antialiasing (élsimítás) finomabb vonalakat és sugarakat ad.

Az 51. sor igazítja (fordítja le geometriai kifejezésekre) a kép közepéhez a painter-t.

Az 52. sor skálázza a koordináta rendszert (de tényleg, csak játssz egy kicsit az értékekkel, hogy lásd).

Ahhoz, hogy a lapra rajzoljon, a qp-nek kell egy toll a körvonalakhoz és egy ecset az alakzatok kitöltéséhez. A QPen-ben van szín, stílus (vonaltípus: folyamatos, szaggatott stb.), az objektumtól függő vastagság, vég- és csatlakozási pont stílus. Ha külön körvonalra nincs szükség, akkor a Qt.NoPen különleges objektum használható. A QBrush rendelkezik stílussal, színnel, erősséggel és textúrával.

Az 54-56. sor állítja be ezeket, az alapbeállítástól eltérő attribútumokat. Próbáld meg cserélni az ecset stílusát a Qt.DiagCrossPattern-ben és nézd a hatást.

Az 57. sor rajzolja meg az ellipszist (a kör olyan ellipszis, aminek kis- és nagy tengelye egyforma) a négyszögön belül kezdve -97, 97, 194 szélestől és 194 magastól. Az értékek nem pixelek, hanem az 52. sor skálájához viszonyítottak.

Az eljárás kódjának többi része ismétlő jellegű, egy különleges tulajdonság kivételével, amit elmagyarázok.

Az 59. sor meghívja a setPen eljárást NoPen-nel. Az óra mutatója egyszerű, színezett gyémánt alak körvonal nélkül, ellentétben az óra testével, aminek a kerete, vagy foglalata világosszürke.

A 61. sor menti a qp aktuális állapotát.

A 62. sor számítja ki az óra jelzőjének szögét az aktuális időhöz és elforgatja a QPainter-t ezzel az értékkel. Például 3.05-kor a szög:

30*(3+5/60)=30*3,083=92,9, vagy csak egy kicsivel megy túl a számlapon a 3-on (12 óra a 0 fok).

A 63. sor rajzolja meg az óra mutatóját.

A 64. sor állítja vissza a painter-t az előző állásába, az elforgatást megszüntetve.

A 66.-tól 70.-ig rajzol 12 kis kört 90-es sugárral és 4-es méretben, 30 fokokat forgatva közöttük. Itt nincs szükség mentésre és helyreállításra, mivel a 360 fok teljes.

A 72-77. sor rajzolja meg a perc mutatót, kiszámolva a szöget az aktuális perc és másodperc értékből. A másodperceket az óra számításakor eldobtuk, mivel a különbség nagyon kicsi lenne.

A 79.-től 84.-ig sorok rajzolják meg a perc mutatót.

A perc jelöléseket az 5 perces jelölőkhöz hasonló módon rajzolja meg, átugorva az ötperces pozíciókat. Ennek érdekében modulo operátort használ egy if kifejezésen belül. A modulo 5 az osztás után visszaadja a maradékot.

```
1 % 5          ==> 1   draw mark
2 % 5          ==> 2   draw mark
3 % 5          ==> 3   draw mark
4 % 5          ==> 5   draw mark
5 % 5          ==> 0   skip mark
6 % 5          ==> 1   draw mark
```

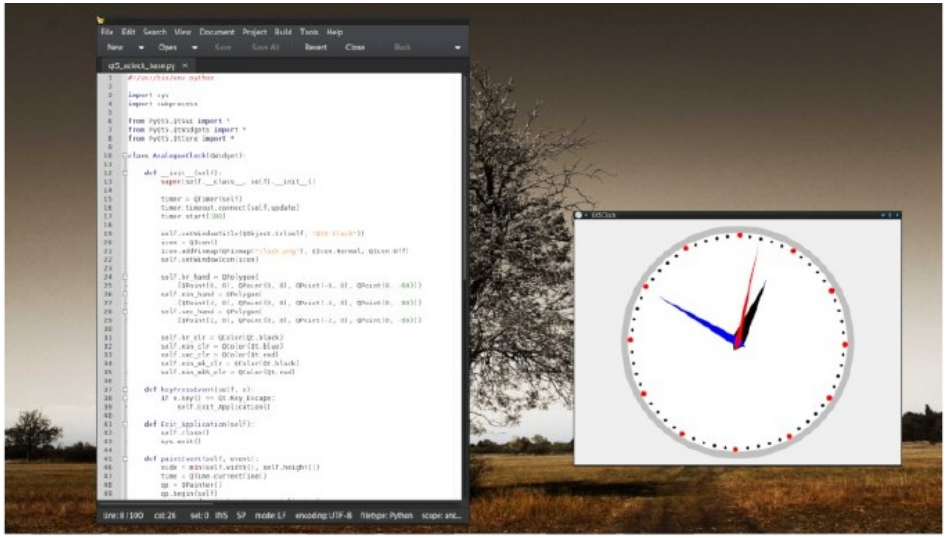
Az előbbiek ismétlődnek minden alkalommal, amikor az időzítő aktiválódik, másodpercenként tízszer. Ha az időzítőt egy másodpercre állítjuk, akkor a frissítési időt úgy egy ezredmásodperccel elkéshetjük, ami miatt a nagymutató nem mindig kerül a megfelelő értékre. Ha nem kell a nagymutató, akkor távolítsd el az azt megrajzoló kódot és az időzítő értékét a 17. sorban állítsd 1000-re. Ez csökkenti a rendszerterhelést (noha a terhelés elég kicsi még nagymutatóval is).

A kód záró része ismerős lehet:

```
if __name__ == "__main__":
        app = QApplication(sys.argv)
        clock = AnalogueClock()
        clock.show()
        sys.exit(app.exec_())
```

Ezzel kész az alap óránk.

Ha ezt a kódot futtatod, egy elég nagy méretű órát látsz, aminek az optimális helyét az ablakkezelő határozza meg.



Az irányítás visszaszerzéséhez tégy egy sort a self.setWindowTitle mögé (az __init__ … definiciónál).

self.resize(120, 120)

hogy kisebb legyen az óra.

Próbáld átméretezni az ablakot, vagy állítsd teljes képernyősre.

Kiegészítés tulajdonságokkal

Az egyes új tulajdonságokat egyenként adom hozzá és tesztelem. Ha Geany-t használsz, akkor elég könnyen megy. Módosítsd a kódot és nyomj F5-öt. Az alkalmazás remélhetően elindul és egy kis konzolablak nyílik meg. Az ablak a Python üzeneteit hivatott megjeleníteni. A Geany alapbeállítás szerinti terminálja az xterm. Ha nincs telepítve az xterm, vagy másikat akarsz használni, akkor az a Szerkesztés menüben a Beállítások → Eszközök-nél állítható be.

Például: konsole -e "/bin/sh %c"

Ha az Alkalmazás sikeresen elindul és a változások megfelelőek, egyszerűen nyomd meg az ESC gombot az az Alkalmazás bezárásához és a Return gombot a terminál ablakának bezárásához. Ha nem jó, akkor olvasd el a terminálablak üzeneteit, hogy megpróbáld kitalálni, mi romlott el. A menet: F5, ESC, Return, kódszerkesztés és a folyamat ismétlése. Az F5-öt lenyomása a kódot automatikusan menti.

Átlátszóság

Először is szabadulj meg az ablakkerettől és a keret hátteret állítsd átlátszóra. Ezt a két sort helyezd el valahol az __init__ eljárás elejénél. A pontos helye nem érdekes addig, amíg az adott eljáráson belül van.

```
        self.setWindowFlags(Qt.FramelessWindowHint)
        self.setAttribute(Qt.WA_TranslucentBackground, True)
```

Talán korábban kellett volna megemlítenem, hogy egy eljárás, vagy függvény ott végződik, ahol a behúzásnak vége. Ez a hurkokra és más vezérlő struktúrákra is igaz, és ez az, amiért a négy szóközös behúzás annyira fontos.

Most, hogy már nincs ablakkeret örülünk, hogy van keyPressEvent-ünk, mivel nincs bezáró gombunk. Nyomj Esc-et az óra bezárásához.

Ezt a kódba opcióként be is foglalhatjuk.

Változtasd meg az __init__-hez adott két sort a következőre:

```
if transparency == 'transparent':
        self.setWindowFlags(Qt.FramelessWindowHint)
        self.setAttribute(Qt.WA_TranslucentBackground, True)
```

Az „if” egy vonalban legyen a „super”-rel és a következő két sor négy szóközzel behúzott.

A vonal után:

```
if __name__ == "__main__":
```

Ezt a sort négy szóköz behúzással add hozzá:

```
    transparency = 'transparent'
```

Most, ha nem kell az átlátszatlanság, a fenti sort váltsd erre:

```
    transparency = ''
```

Hasonló módszerrel fogom a többi opciót is hozzáadni és végül parancssorból, vagy az alkalmazás indításakor kapcsolhatóvá tenni.

Pálcikák

A percek pontjait vonásokká alakításához írd ezt a sort az átlátszóság sora utánra:

```
    transparency = 'transparent'
    marks = 'batons'
```

Az 5 perces jelölések rajzolata kódját változtasd erre:

```
qp.setPen(QPen(self.min_mk5_clr))
qp.setBrush(QBrush(self.min_mk5_clr))
        for i in range(0, 12):
                if marks == 'batons':
                        qp.drawLine(85, 0, 95, 0)
                else:
                        qp.drawEllipse(90, 0, 4, 4)
                qp.rotate(30.0)
```

És a percjelölők kódját erre:

```
qp.setBrush(QBrush(self.min_mk_clr))
qp.setPen(QPen(self.min_mk_clr))
for j in range(0, 60):
        if j % 5:
                if marks == 'batons':
                        qp.drawLine(90, 0, 95, 0)
                else:
                        qp.drawEllipse(90, 0, 2, 2)
        qp.rotate(6.0)
```

Most, ha a

```
    marks = 'batons'
```

átváltod erre

```
    marks = ''
```

visszakapod a pontokat.

A nagymutató söpréséhez

írd be, hogy

```
    movement = 'sweep'
```

e mögé

```
    marks = 'batons'
```

Változtasd meg a nagymutatót rajzoló kódot így:

```
    qp.save()
    if movement == 'sweep':
            qp.rotate(6.0 * (time.second() + time.msec() / 1000))
    else:
            qp.rotate(6.0 * (time.second()))
    qp.drawConvexPolygon(self.sec_hand)
    qp.restore()
```

Ez ezredmásodpercet ad hozzá az egyenlethez, hogy a léptetés kisebb legyen. Ez nem igazán finom mozgatás, de egy pozícióban tízszer rajzolja újra másodpercenként, ami eléggé közelít egy igazi falióra nagymutatójának mozgásához. Legtöbben ezt az eljárást alkalmazzák

Feliratok

Add hozzá

```
    legend = 'Qt5 Clock'
```

ez után

```
    movement = 'sweep'
```

Add hozzá ezt

```
font = QFont()
font.setFamily("Arial") # vagy amit szeretnél / telepített
font.setPointSize(12)
font.setBold(False)
font.setItalic(True)
font.setUnderline(True)
font.setWeight(75)
qp.setFont(font)
qp.drawText(-50, 25, 100, 16, Qt.AlignCenter, legend)
```

valahová a qp.begin és a qp.en kifejezés közé új blokként. Ügyelj arr, hogy a rendszeredben telepített betűtípust használj. A többi betűjellemzőt ízlésednek megfelelően állítsd be. Ez intézi a feliratok megjelenítését.

Az induló pozíció

E mögé

```
    legend = 'Qt5 Clock'
```

írd be

```
xpos = 20
ypos = 20
```

Az ezt követető kódot

```
    clock.show()
```

változtasd meg erre:

```
    clock.show()
    if transparency == 'transparent':
            clock.move(xpos, ypos)
    else:
            clock.move(xpos, ypos + 20)
```

Vedd észre a kettőst == az if kifejezésben == 'transparent': ez nem hozzárendelés. Itt az egyenlőséget vizsgálod.

Egy kicsit több kell y irányban, hogy a címsort elhelyezd abban az estben, ha nincs átlátszóság.

Sötét téma

E mögé

```
    ypos = 20
```

írd be

```
    theme = 'light'
```

Az __init__ eljárásban keresd meg a 31.-től 35. sorig és ilyesmit keress meg:

```
        self.hr_clr = QColor(Qt.black)
        self.min_clr = QColor(Qt.blue)
        self.sec_clr = QColor(Qt.red)
        self.min_mk_clr = QColor(Qt.black)
        self.min_mk5_clr = QColor(Qt.red)
```

majd erre változtasd meg:

```
        if theme == 'light':
            self.hr_clr = QColor(Qt.black)
            self.min_clr = QColor(Qt.blue)
            self.sec_clr = QColor(Qt.red)
            self.min_mk_clr = QColor(Qt.black)
            self.min_mk5_clr = QColor(Qt.red)
        else:
            self.hr_clr = QColor(Qt.lightGray)
            self.min_clr = QColor(Qt.cyan)
            self.sec_clr = QColor(Qt.red)
            self.min_mk_clr = QColor(Qt.white)
            self.min_mk5_clr = Qt.cyan
```

Most már menjünk a paintEvent eljárásban a kódhoz, ami megfesti a hátteret és így néz ki:

```
            qp.setPen(QPen(Qt.lightGray, 6))
            style = Qt.BrushStyle(Qt.SolidPattern)
            qp.setBrush(QBrush(QColor(255, 255, 255), style))
            qp.drawEllipse(QRect(-97, -97, 194, 194))
```

majd erre változtasd meg:

```
        if theme == 'light':
            qp.setPen(QPen(Qt.lightGray, 6))
            style = Qt.BrushStyle(Qt.SolidPattern)
            qp.setBrush(QBrush(QColor(255, 255, 255), style))
```

```
else:
        qp.setPen(QPen(QColor(189, 183, 107), 4))
        style = Qt.BrushStyle(Qt.SolidPattern)
        gradient = QRadialGradient(QPoint(0, 0), 75)
        gradient.setColorAt(0, QColor(0,0,0))
        gradient.setColorAt(1, QColor(22,8,0))
        qp.setBrush(gradient)
qp.drawEllipse(QRect(-97, -97, 194, 194))
```

Próbáld ki a theme = 'light'-ot theme = 'dark'-ra váltani, lásd az alternatív stílust.

Sugárirányú elmosást adtam hozzá a sötét témához, hogy mutassam a tulajdonságot. Kis méretű órán ez nem igazán észrevehető. Próbáld ki teljes képernyőn, vagy más színekkel.

A következő részben bemutatom, hogyan helyezzük el a számokat az órán, hogyan forgassuk a római számokat az óra körívén és hogyan tegyük mindezt opcionálisan választhatóvá az alkalmazás indító parancsában.

A szerkesztő megjegyzése: az Alkalmi Python sorozat összes kódja letöltésre itt elérhető.