from PyQt5.QtGui import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import \*

import random

import time

bomb1 = QImage("./images/bug.png")

flag1 = QImage("./images/flag.png")

start1 = QImage("./images/dog.png")

time1 = QImage("./images/clock-select.png")

colors = { 1: QColor('#ff3300'), 2: QColor('#990099'), 3: QColor('#000066'), 4: QColor('#ffcc00'), 5: QColor('#006600'), 6: QColor('#4CAF50'), 7: QColor('#E91E63'), 8: QColor('#FF9800') }

t = [(8, 10), (16, 40), (24, 99)] #3 уровня

ready = 0

run = 1

fail = 2

suc = 3

STATUS\_ICONS = {

ready: "./images/plus.png",

run: "./images/smiley.png",

fail: "./images/cross.png",

suc: "./images/smiley-dog.png",

}

class Pos(QWidget):

expandable = pyqtSignal(int, int)

clicked = pyqtSignal()

ohno = pyqtSignal()

def \_\_init\_\_(self, x, y, \*args, \*\*kwargs):

super(Pos, self).\_\_init\_\_(\*args, \*\*kwargs)

self.setFixedSize(QSize(20, 20))

self.x = x

self.y = y

def reset(self):

self.is\_start = False

self.is\_mine = False

self.adjacent\_n = 0

self.is\_revealed = False

self.is\_flagged = False

self.update()

def paintEvent(self, event):

p = QPainter(self)

p.setRenderHint(QPainter.Antialiasing)

r = event.rect()

if self.is\_revealed:

color = self.palette().color(QPalette.Background)

outer, inner = color, color

else:

outer, inner = Qt.gray, Qt.lightGray

p.fillRect(r, QBrush(inner))

pen = QPen(outer)

pen.setWidth(1)

p.setPen(pen)

p.drawRect(r)

if self.is\_revealed:

if self.is\_start:

p.drawPixmap(r, QPixmap(start1))

elif self.is\_mine:

p.drawPixmap(r, QPixmap(bomb1))

elif self.adjacent\_n > 0:

pen = QPen(colors[self.adjacent\_n])

p.setPen(pen)

f = p.font()

f.setBold(True)

p.setFont(f)

p.drawText(r, Qt.AlignHCenter | Qt.AlignVCenter, str(self.adjacent\_n))

elif self.is\_flagged:

p.drawPixmap(r, QPixmap(flag1))

def flag(self):

self.is\_flagged = True

self.update()

self.clicked.emit()

def reveal(self):

self.is\_revealed = True

self.update()

def click(self):

if not self.is\_revealed:

self.reveal()

if self.adjacent\_n == 0:

self.expandable.emit(self.x, self.y)

self.clicked.emit()

def mouseReleaseEvent(self, e):

if (e.button() == Qt.RightButton and not self.is\_revealed):

self.flag()

elif (e.button() == Qt.LeftButton):

self.click()

if self.is\_mine:

self.ohno.emit()

class MainWindow(QMainWindow):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)

self.b\_size, self.n\_mines = t[2] #уровень

w = QWidget()

hb = QHBoxLayout()

self.mines = QLabel()

self.mines.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)

self.clock = QLabel()

self.clock.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)

f = self.mines.font()

f.setPointSize(24)

f.setWeight(75)

self.mines.setFont(f)

self.clock.setFont(f)

self.\_timer = QTimer()

self.\_timer.timeout.connect(self.update\_timer)

self.\_timer.start(1000)

self.mines.setText("%03d" % self.n\_mines)

self.clock.setText("000")

self.button = QPushButton()

self.button.setFixedSize(QSize(32, 32))

self.button.setIconSize(QSize(32, 32))

self.button.setIcon(QIcon("./images/smiley.png"))

self.button.setFlat(True)

self.button.pressed.connect(self.button\_pressed)

l = QLabel()

l.setPixmap(QPixmap.fromImage(bomb1))

l.setAlignment(Qt.AlignRight | Qt.AlignVCenter)

hb.addWidget(l)

hb.addWidget(self.mines)

hb.addWidget(self.button)

hb.addWidget(self.clock)

l = QLabel()

l.setPixmap(QPixmap.fromImage(time1))

l.setAlignment(Qt.AlignLeft | Qt.AlignVCenter)

hb.addWidget(l)

vb = QVBoxLayout()

vb.addLayout(hb)

self.grid = QGridLayout()

self.grid.setSpacing(5)

vb.addLayout(self.grid)

w.setLayout(vb)

self.setCentralWidget(w)

self.init\_map()

self.update\_status(ready)

self.reset\_map()

self.update\_status(ready)

self.show()

def init\_map(self):

# Добавление позиций на карту

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = Pos(x, y)

self.grid.addWidget(w, y, x)

# сигнал для управления

w.clicked.connect(self.trigger\_start)

w.expandable.connect(self.expand\_reveal)

w.ohno.connect(self.game\_over)

def reset\_map(self):

# очистка мин

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = self.grid.itemAtPosition(y, x).widget()

w.reset()

# добавить мины

positions = []

while len(positions) < self.n\_mines:

x, y = random.randint(0, self.b\_size - 1), random.randint(0, self.b\_size - 1)

if (x, y) not in positions:

w = self.grid.itemAtPosition(y, x).widget()

w.is\_mine = True

positions.append((x, y))

def get\_adjacency\_n(x, y):

positions = self.get\_surrounding(x, y)

n\_mines = sum(1 if w.is\_mine else 0 for w in positions)

return n\_mines

# смежность к позициям

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = self.grid.itemAtPosition(y, x).widget()

w.adjacent\_n = get\_adjacency\_n(x, y)

# начало

while True:

x, y = random.randint(0, self.b\_size - 1), random.randint(0, self.b\_size - 1)

w = self.grid.itemAtPosition(y, x).widget()

# чтобы не начать с мины

if (x, y) not in positions:

w = self.grid.itemAtPosition(y, x).widget()

w.is\_start = True

# все позиции вокруг этого, если они тоже не являются минами

for w in self.get\_surrounding(x, y):

if not w.is\_mine:

w.click()

break

def get\_surrounding(self, x, y):

positions = []

for xi in range(max(0, x - 1), min(x + 2, self.b\_size)):

for yi in range(max(0, y - 1), min(y + 2, self.b\_size)):

positions.append(self.grid.itemAtPosition(yi, xi).widget())

return positions

def button\_pressed(self):

if self.status == run:

self.update\_status(fail)

self.reveal\_map()

elif self.status == fail:

self.update\_status(ready)

self.reset\_map()

def reveal\_map(self):

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = self.grid.itemAtPosition(y, x).widget()

w.reveal()

def expand\_reveal(self, x, y):

for xi in range(max(0, x - 1), min(x + 2, self.b\_size)):

for yi in range(max(0, y - 1), min(y + 2, self.b\_size)):

w = self.grid.itemAtPosition(yi, xi).widget()

if not w.is\_mine:

w.click()

def trigger\_start(self, \*args):

if self.status != run:

# первый клик.

self.update\_status(run)

# отсчёт

self.\_timer\_start\_nsecs = int(time.time())

def update\_status(self, status):

self.status = status

self.button.setIcon(QIcon(STATUS\_ICONS[self.status]))

def update\_timer(self):

if self.status == run:

n\_secs = int(time.time()) - self.\_timer\_start\_nsecs

self.clock.setText("%03d" % n\_secs)

def game\_over(self):

self.reveal\_map()

self.update\_status(fail)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

window = MainWindow()

app.exec\_()