

**Moonsweeper**

Chet elliklarga yaqinlashmasdan Q'teening sirli oyini o'rganing!

[Martin Fitspatrik](https://www.pythonguis.com/authors/martin-fitzpatrick/) tomonidan  Oxirgi yangilangan2023 yil 13 sentyabr PyQt5  [Misol ilovalar](https://www.pythonguis.com/examples/)

[Manba kodi](https://www.pythonguis.com/d/minesweeper.zip) [Ubuntu o'rnatuvchi](https://www.pythonguis.com/d/Moonsweeper.deb) [Windows o'rnatuvchisi](https://www.pythonguis.com/d/MoonsweeperSetup.exe) [Mac disk tasviri](https://www.pythonguis.com/d/Moonsweeper.dmg)

Moonsweeper - bu bitta o'yinchili boshqotirma video o'yini. O'yinning maqsadi halokatli B'ug musofirlariga yaqinlashmasdan, qo'ngan kosmik raketangiz atrofidagi hududni o'rganishdir. Sizning ishonchli tricounteringiz sizga yaqin atrofdagi B'ugs sonini aytib beradi.

*Bu Minesweeper* -da modellashtirilgan oddiy bitta o'yinchili kashfiyot o'yini , unda siz yashirin minalarga tegmasdan barcha plitkalarni ochishingiz kerak. Ushbu amalga oshirishda QWidgeto'z holatini minalar, holati va qo'shni minalar soni sifatida saqlaydigan plitkalar uchun maxsus ob'ektlardan foydalaniladi. Ushbu versiyada minalar begona xatolar (B'ug) bilan almashtiriladi, ammo ular boshqa har qanday narsa bo'lishi mumkin.

Windows, Linux va Mac uchun o'rnatuvchilarni to'liq manba kodi bilan birga yuqorida yuklab olish mumkin.

**Manbadan yugurish**

[Manba arxivini yuklab oling yoki bu yerda](https://github.com/mfitzp/15-minute-apps/tree/master/minesweeper) Github-dan tekshiring . Quyidagilardan foydalanib o'rnatish talablari:

piton

pip3 install -r requirements.txt

*Keyin Moonsweeper dasturini* quyidagi bilan ishlatishingiz mumkin :

piton

python3 minesweeper.py

Kod qanday ishlashi haqida batafsil ma'lumot olish uchun o'qing. Kod PyQt5 yoki PySide2 (Python uchun Qt) bilan mos keladi, faqat import va signal imzosi o'zgaradi (keyinroq ko'ring).

* PyQt5

* PySide2

piton

from PyQt5.QtGui import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import \*

**O'yin maydoni**

Moonsweeper o'yin maydoni NxN tarmog'i bo'lib, unda ma'lum miqdordagi minalar mavjud. Biz foydalanadigan o'lchamlar va minalar soni Minesweeper dasturining Windows versiyasi uchun standart qiymatlardan olingan. Foydalanilgan qiymatlar quyidagi jadvalda keltirilgan:

*Martin Fitspatrik tomonidan*[**PyInstaller bilan Python ilovalarini qadoqlash**](https://www.pythonguis.com/packaging-book/) - Ushbu bosqichma-bosqich qo'llanma sizga oddiy misollardan tortib to to'liq o'rnatuvchilar va imzolangan bajariladigan fayllargacha shaxsiy Python ilovalaringizni qadoqlash orqali o'rgatadi.

[Batafsil ma'lumot](https://www.pythonguis.com/packaging-book/) [Kitobni oling](https://secure.pythonguis.com/01hf77hrbf5v8z5kjtwbhmbwjz/?couponCode=WE5GY0)

| **Daraja** | **O'lchamlari** | **Minalar soni** |
| --- | --- | --- |
| Oson | 8 x 8 | 10 |
| O'rta | 16 x 16 | 40 |
| Qattiq | 24 x 24 | 99 |

LEVELSBiz bu qiymatlarni faylning yuqori qismida belgilangan doimiy sifatida saqlaymiz . Barcha o'yin maydonlari kvadrat bo'lgani uchun biz qiymatni faqat bir marta saqlashimiz kerak (8, 16 yoki 24).

piton

LEVELS = [

("Easy", 8, 10),

("Medium", 16, 40),

("Hard", 24, 99)

]

O'yin to'ri bir qancha usullarda ko'rsatilishi mumkin, masalan, o'yin pozitsiyalarining turli holatlarini ifodalovchi 2D "ro'yxatlar ro'yxati" (mening, ochilgan, bayroqlangan).

Biroq, bu amalga oshirish ob'ektga yo'naltirilgan yondashuvdan foydalanadi. Xaritadagi alohida kvadratlar ularning hozirgi holati haqidagi barcha tegishli ma'lumotlarni o'z ichiga oladi va shuningdek, o'zlarini chizish uchun javobgardir. Qt-da biz buni QWidgetoddiy bo'yoq funktsiyasidan pastki sinflarga ajratish va keyin amalga oshirish orqali amalga oshirishimiz mumkin.

Bizning plitka ob'yektlarimiz quyi sinflarga bo'linganligi sababli, QWidgetbiz ularni boshqa har qanday vidjet kabi joylashtirishimiz mumkin. Biz buni o'rnatish orqali qilamiz QGridLayout.

piton

self.grid = QGridLayout()

self.grid.setSpacing(5)

self.grid.setSizeConstraint(QLayout.SetFixedSize)

Biz o'z pozitsiyamizdagi plitkalar vidjetlarini yaratib, ularni tarmoqqa qo'shish orqali o'yinni sozlashimiz mumkin. Darajani dastlabki sozlash LEVELSoynadan o'qiydi va bir qator o'zgaruvchilarni tayinlaydi. Oyna sarlavhasi va mina hisoblagichi yangilanadi, so'ngra tarmoqni sozlash boshlanadi.

piton

def set\_level(self, level):

self.level\_name, self.b\_size, self.n\_mines = LEVELS[level]

self.setWindowTitle("Moonsweeper - %s" % (self.level\_name))

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

self.clear\_map()

self.init\_map()

self.reset\_map()

O'rnatish funktsiyalari keyinroq ko'rib chiqiladi.

Sinf Posplitkani ifodalaydi va uning xaritadagi tegishli pozitsiyasi uchun barcha tegishli ma'lumotlarni o'z ichiga oladi, jumladan, masalan, mina bo'ladimi, aniqlanganmi, bayroqlanganmi va yaqin atrofdagi minalar soni.

Har bir ob'ektda shuningdek, *bosilgan* , *ochiladigan* va *kengaytiriladigan*Pos 3 ta maxsus signal mavjud bo'lib , biz ularni maxsus slotlarga ulaymiz. Nihoyat, biz oyna hajmini yangi tarkibga moslashtirish uchun o'lchamini o'zgartirishni chaqiramiz. Bu, aslida, faqat deraza *qisqarganida* kerak bo'ladi - Qt uni avtomatik ravishda o'stiradi.

piton

def init\_map(self):

*# Add positions to the map*

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)

*# Connect signal to handle expansion.*

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

w.revealed.connect(self.on\_reveal)

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

*# Place resize on the event queue, giving control back to Qt before.*

QTimer.singleShot(0, lambda: self.resize(1,1)) *# <1>*

1. Taymer singleShotbiz voqea tsikliga qaytganimizdan so'ng va Qt yangi tarkibdan xabardor bo'lganimizdan keyin o'lchamni o'zgartirishni ta'minlash uchun talab qilinadi.

Endi bizda pozitsion plitka ob'ektlari to'plami mavjud, biz o'yin taxtasining dastlabki sharoitlarini yaratishni boshlashimiz mumkin. Bu bir qator funktsiyalarga bo'lingan. Biz ularni nomlaymiz \_reset(etakchi pastki chiziq tashqi foydalanish uchun mo'ljallanmagan shaxsiy funktsiyani ko'rsatadigan konventsiyadir). Asosiy funksiya reset\_mapuni sozlash uchun navbatma-navbat bu funksiyalarni chaqiradi.

Jarayon quyidagicha -

1. Maydondan barcha minalarni olib tashlang (va ma'lumotlarni qayta o'rnating).
2. Maydonga yangi minalar qo'shing.
3. Har bir pozitsiyaga ulashgan minalar sonini hisoblang.
4. Boshlang'ich belgisini (raketa) qo'shing va dastlabki tadqiqotni ishga tushiring.
5. Taymerni qayta o'rnating.

Buni amalga oshirish uchun kod:

piton

def reset\_map(self):

self.\_reset\_position\_data()

self.\_reset\_add\_mines()

self.\_reset\_calculate\_adjacency()

self.\_reset\_add\_starting\_marker()

self.update\_timer()

1 dan 5 gacha bo'lgan alohida bosqichlar quyida har bir qadam uchun kod bilan batafsil tavsiflanadi.

Birinchi qadam xaritadagi har bir pozitsiya uchun ma'lumotlarni qayta o'rnatishdir. Biz doskadagi har bir pozitsiyani takrorlaymiz, .reset()har bir nuqtada vidjetni chaqiramiz. Funktsiya kodi .reset() bizning maxsus sinfimizda belgilangan Pos, biz keyinroq batafsil ko'rib chiqamiz. Hozircha u minalarni, bayroqlarni tozalab, o‘z o‘rnini oshkor etilmagan holga keltirishini bilish kifoya.

piton

def \_reset\_position\_data(self):

*# Clear all mine positions*

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()

Endi barcha pozitsiyalar bo'sh, biz xaritaga minalar qo'shish jarayonini boshlashimiz mumkin. Minalar maksimal soni n\_minesyuqorida tavsiflangan daraja sozlamalari bilan belgilanadi.

piton

def \_reset\_add\_mines(self):

*# Add mine positions*

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))

*# Calculate end-game condition*

self.end\_game\_n = (self.b\_size \* self.b\_size) - (self.n\_mines + 1)

return positions

Minalar joyida bo'lsa, endi biz har bir pozitsiya uchun "qo'shni" raqamini hisoblashimiz mumkin - berilgan nuqta atrofida 3x3 o'lchamdagi to'rdan foydalanib, yaqin atrofdagi minalar sonini. Maxsus funksiya get\_surroundingshunchaki ma'lum xva yjoylashuv atrofidagi pozitsiyalarni qaytaradi. Biz kon va do'kon bo'lgan bularning sonini hisoblaymiz is\_mine == True.

Bu erda qo'shni hisoblarni oldindan hisoblash keyinchalik ochish mantiqini soddalashtirishga yordam beradi. Ammo bu shuni anglatadiki, biz foydalanuvchiga o'zining dastlabki harakatini tanlashiga ruxsat bera olmaymiz - biz buni "raketa atrofidagi dastlabki tadqiqot" deb tushuntirib, uni butunlay oqilona qilishimiz mumkin.

Hisobni kechiktirish orqali buni o'zingiz hal qilishga harakat qiling!

piton

def \_reset\_calculate\_adjacency(self):

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

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

return sum(1 for w in positions if w.is\_mine)

*# Add adjacencies to the positions*

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)

*Birinchi harakat har doim* haqiqiy bo'lishini ta'minlash uchun boshlang'ich belgisi ishlatiladi . Bu biz mina bo'lmagan pozitsiyani topmagunimizcha, tasodifiy pozitsiyalarni samarali sinab ko'rish orqali tarmoq bo'ylab *qo'pol* qidiruv sifatida amalga oshiriladi . Buning uchun qancha urinishlar ketishini bilmasligimiz sababli, biz uni uzluksiz tsiklga o'rashimiz kerak.

Bu joy topilgach, biz uni boshlang'ich joy sifatida belgilaymiz va keyin atrofdagi barcha pozitsiyalarni o'rganishni ishga tushiramiz. Biz tsikldan chiqib, tayyor holatni tiklaymiz.

piton

def \_reset\_add\_starting\_marker(self):

*# Place starting marker.*

*# Set initial status (needed for .click to function)*

self.update\_status(STATUS\_READY)

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()

*# We don't want to start on a mine.*

if not w.is\_mine:

w.is\_start = True

w.is\_revealed = True

w.update()

*# Reveal all positions around this, if they are not mines either.*

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

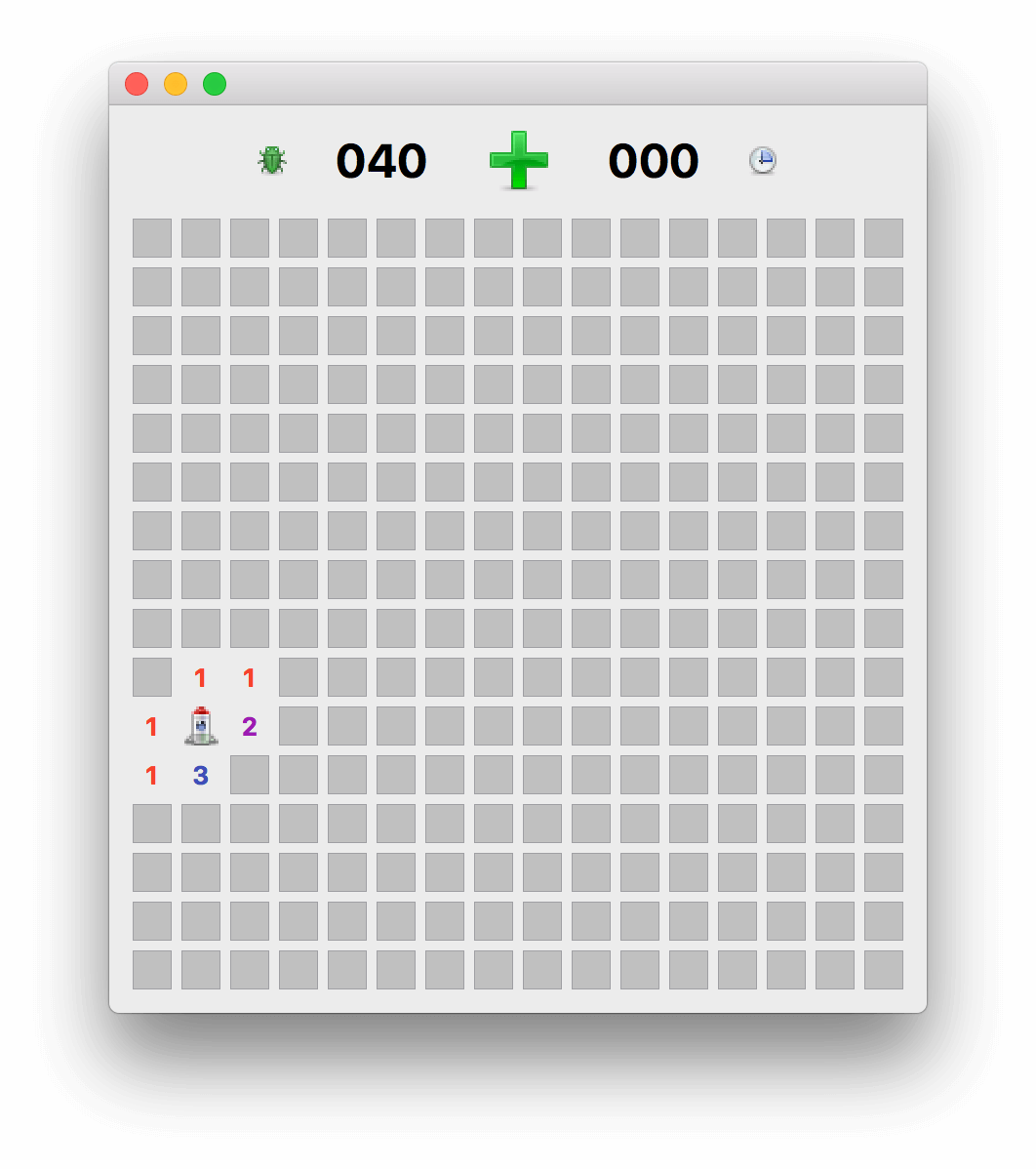
if not w.is\_mine:

w.click()

break

*# Reset status to ready following initial clicks.*

self.update\_status(STATUS\_READY)

Moonsweeper uchun boshlang'ich holati

**Plitkalarni joylashtirish**

O'yin strukturaviy o'yin bo'lib, individual plitka pozitsiyalari o'z holati haqida ma'lumotga ega bo'ladi. Bu shuni anglatadiki, Posplitkalar o'zlarining o'yin mantiqlarini boshqarishga qodir.

Sinf nisbatan murakkab bo'lganligi sababli Pos, u bu erda bir necha qismlarga bo'linadi va ular o'z navbatida muhokama qilinadi. Dastlabki o'rnatish \_\_init\_\_bloki oddiy bo'lib, xva ypozitsiyasini qabul qiladi va uni ob'ektda saqlaydi. Pospozitsiyalar yaratilgandan keyin hech qachon o'zgarmaydi.

O'rnatishni yakunlash uchun .reset()barcha ob'ekt atributlarini sukut bo'yicha, nol qiymatlariga qaytaradigan funksiya chaqiriladi. Bu konni *boshlang'ich pozitsiyasi emas* , *mina emas* , *oshkor qilinmagan* va *belgilanmagan* deb belgilaydi . Biz qo'shni hisobni ham tiklaymiz.

* PyQt5

* PySide2

piton

class Pos(QWidget):

expandable = pyqtSignal(int,int)

revealed = pyqtSignal(object)

clicked = 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

self.reset()

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()

O'yin o'yin maydonidagi plitkalar bilan sichqonchaning o'zaro ta'siriga qaratilgan, shuning uchun sichqonchani bosishlarini aniqlash va ularga munosabat bildirish asosiy hisoblanadi. Qt da biz sichqonchani bosishlarini aniqlash orqali ushlaymiz mouseReleaseEvent. Buni bizning shaxsiy vidjetimiz uchun qilish uchun Posbiz sinfda ishlov beruvchini aniqlaymiz. Bu QMouseEventsodir bo'lgan voqealarni o'z ichiga olgan ma'lumot bilan qabul qilinadi. Bunday holda, biz faqat sichqonchaning chap tugmasi yoki o'ng tugmasi bo'shatilganligi bilan qiziqamiz.

Sichqonchaning chap tugmachasini bosish uchun biz kafelning bayroqlanganligini yoki allaqachon ochilganligini tekshiramiz. Agar shunday bo'lsa, biz chertishni e'tiborsiz qoldiramiz - bayroqchali plitalarni "xavfsiz" qilib, tasodifan bosish mumkin emas. Agar kafel belgilanmagan bo'lsa, biz shunchaki .click()usulni ishga tushiramiz (keyinroq ko'ring).

Sichqonchaning o'ng tugmasi bilan ko'rsatilmagan plitkalarni bosish uchun *biz*.toggle\_flag() bayroqni yoqish va o'chirish usulini chaqiramiz .

piton

def mouseReleaseEvent(self, e):

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

self.toggle\_flag()

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

*# Block clicking on flagged mines.*

if not self.is\_flagged and not self.is\_revealed:

self.click()

Ishlovchi tomonidan chaqiriladigan usullar mouseReleaseEventquyida tavsiflanadi.

Ishlovchi .toggle\_flagshunchaki .is\_flaggedo'ziga teskarisini o'rnatadi ( Truebo'ladi False, Falsebo'ladi True) uni yoqish va o'chirish effektiga ega. E'tibor bering, biz .update()holatni o'zgartirib, qayta chizishga majburlash uchun qo'ng'iroq qilishimiz kerak . Shuningdek, biz .clickedtaymerni ishga tushirish uchun ishlatiladigan maxsus signalimizni chiqaramiz - chunki bayroqni qo'yish kvadratni ochish uchun emas, balki boshlang'ich sifatida ham hisoblanishi kerak.

Usul .click()sichqonchaning chap tugmachasini bosadi va o'z navbatida kvadratni ochishni boshlaydi. Agar bunga qo'shni minalar soni Posnolga teng bo'lsa, biz .expandableo'rganilayotgan hududni avtomatik ravishda kengaytirish jarayonini boshlash uchun signalni ishga tushiramiz (keyinroq ko'ring). Nihoyat, biz yana .clickedo'yin boshlanishini bildirish uchun chiqaramiz.

Nihoyat, .reveal()usul kafel allaqachon aniqlanganligini tekshiradi va agar .is\_revealedbo'lmasa True. .update()Vidjetni qayta bo'yashni boshlash uchun yana qo'ng'iroq qilamiz .

Signalning ixtiyoriy emissiyasi .revealedfaqat o'yin oxiri to'liq xaritasini ochish uchun ishlatiladi. Har bir ochilish qaysi plitkalar ham aniqlanishi mumkinligini aniqlash uchun qo'shimcha qidiruvni ishga tushirganligi sababli, butun xaritani ochish juda ko'p ortiqcha qo'ng'iroqlarni keltirib chiqaradi. Bu erda signalni bostirish orqali biz bundan qochamiz.

piton

def toggle\_flag(self):

self.is\_flagged = not self.is\_flagged

self.update()

self.clicked.emit()

def click(self):

self.reveal()

if self.adjacent\_n == 0:

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

self.clicked.emit()

def reveal(self, emit=True):

if not self.is\_revealed:

self.is\_revealed = True

self.update()

if emit:

self.revealed.emit(self)

paintEventVa nihoyat, biz vidjetimiz uchun Posjoriy joylashuv holatini ko'rsatishni boshqarish uchun maxsus usulni aniqlaymiz . Vidjet tuvalini maxsus bo'yashni amalga oshirish uchun biz chizishimiz kerak bo'lgan chegaralarni - bu holda vidjetning tashqi chegarasini ko'rsatadigan belgini QPainterolamiz .event.rect()Pos

*Ochilgan plitkalar, kafelning boshlang'ich pozitsiyasi* , *bomba* yoki *bo'sh joy* bo'lishiga qarab turlicha chiziladi . Birinchi ikkitasi mos ravishda raketa va bomba piktogrammalari bilan ifodalanadi. Ular QRectyordamida plitka ichiga tortiladi .drawPixmap. E'tibor bering, biz o'tish orqali QImageo'tish orqali konstantalarni piksel xaritalariga aylantirishimiz kerak QPixmap.

Siz shunday deb o'ylashingiz mumkin: "Nega ularni faqat ob'ektlar sifatida saqlamasligimiz kerak, chunki biz aynan shu narsadan foydalanamiz? Afsuski, siz o'zingiz ishga tushmasdan oldin ob'ektlarni QPixmapyarata olmaysiz .QPixmapQApplication

Bo'sh pozitsiyalar uchun (raketalar emas, bombalar emas) biz ixtiyoriy ravishda qo'shni raqamni ko'rsatamiz, agar u noldan katta bo'lsa. Matnga matn chizish uchun QPainterbiz , hizalama bayroqlari va qator sifatida chizish uchun raqamni .drawText()o'tkazishdan foydalanamiz. Foydalanish qulayligi uchun QRectbiz har bir raqam uchun standart rangni (da saqlangan NUM\_COLORS) belgilab oldik.

*Ochilmagan* plitkalar uchun biz to'rtburchakni ochiq kul rang bilan to'ldirib, plitka chizamiz va quyuqroq kul rangning 1 pikselli chegarasini chizamiz. Agar o'rnatilgan bo'lsa, biz va plitkadan .is\_flaggedfoydalanib, kafelning yuqori qismiga bayroq belgisini ham chizamiz .drawPixmapQRect

piton

def paintEvent(self, event):

p = QPainter(self)

p.setRenderHint(QPainter.Antialiasing)

r = event.rect()

if self.is\_revealed:

if self.is\_start:

p.drawPixmap(r, QPixmap(IMG\_START))

elif self.is\_mine:

p.drawPixmap(r, QPixmap(IMG\_BOMB))

elif self.adjacent\_n > 0:

pen = QPen(NUM\_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))

else:

p.fillRect(r, QBrush(Qt.lightGray))

pen = QPen(Qt.gray)

pen.setWidth(1)

p.setPen(pen)

p.drawRect(r)

if self.is\_flagged:

p.drawPixmap(r, QPixmap(IMG\_FLAG))

**Mexanika**

Biz odatda ma'lum bir nuqta atrofidagi barcha plitkalarni olishimiz kerak, shuning uchun bizda bu maqsad uchun maxsus funktsiya mavjud. U nuqta atrofida 3x3 o'lchamdagi to'r bo'ylab oddiy takrorlanadi va biz to'r chetlarida chegaradan chiqmasligimizni tekshirib ko'ring ( 0 ≥ x ≤ self.b\_size). Qaytarilgan ro'yxatda Poshar bir atrofdagi joydan vidjet mavjud.

piton

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)):

if not (xi == x and yi == y):

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

return positions

Usul expand\_revealnolga qo'shni minalar bo'lgan kafelni bosishga javoban ishga tushiriladi. Bunday holda, biz bosish atrofidagi maydonni nolga qo'shni minalar bo'lgan har qanday bo'shliqqa kengaytirishni xohlaymiz, shuningdek, bu kengaytirilgan maydonning chegarasi atrofidagi har qanday kvadratlarni (bu minalar emas) aniqlamoqchimiz.

to\_expandBiz keyingi iteratsiyada tekshiriladigan pozitsiyalarni o'z ichiga olgan ro'yxat , to\_revealko'rsatiladigan plitka vidjetlarini o'z ichiga olgan ro'yxat va any\_addedtsikldan qachon chiqishni aniqlash uchun bayroqdan boshlaymiz. Birinchi marta yangi vidjetlar qo'shilmasa, tsikl to'xtaydi to\_reveal.

Loop ichida biz qayta o'rnatamiz any\_addedva Falsero'yxatni bo'shatamiz , takrorlash uchun to\_expandvaqtinchalik do'konni saqlab qo'yamiz .l

xHar bir joy va joy uchun ybiz 8 ta vidjetni olamiz. Agar ushbu vidjetlardan biri meniki bo'lmasa va ro'yxatda bo'lmasa, to\_revealbiz uni qo'shamiz. Bu kengaytirilgan maydonning barcha qirralari ochilishini ta'minlaydi. Agar pozitsiyada qo'shni minalar bo'lmasa, biz to\_expandkeyingi iteratsiyada tekshiriladigan koordinatalarni qo'shamiz .

-ga har qanday nodavlat plitkalarni qo'shish to\_revealva faqat hali mavjud bo'lmagan plitkalarni kengaytirish orqali to\_revealbiz kafelga bir necha marta tashrif buyurmasligimizga ishonch hosil qilamiz.

piton

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

"""

Iterate outwards from the initial point, adding new locations to the

queue. This allows us to expand all in a single go, rather than

relying on multiple callbacks.

"""

to\_expand = [(x,y)]

to\_reveal = []

any\_added = True

while any\_added:

any\_added = False

to\_expand, l = [], to\_expand

for x, y in l:

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

for w in positions:

if not w.is\_mine and w not in to\_reveal:

to\_reveal.append(w)

if w.adjacent\_n == 0:

to\_expand.append((w.x,w.y))

any\_added = True

*# Iterate an reveal all the positions we have found.*

for w in to\_reveal:

w.reveal()

**Yakuniy o'yinlar**

O'yin oxiri holati sarlavha ustiga bosilgandan so'ng, oshkor qilish jarayonida aniqlanadi. Ikkita mumkin bo'lgan natija bor -

1. Plitka - bu kon, o'yin tugadi.
2. Plitka kon emas, uni kamaytiring self.end\_game\_n.

Bu nolga yetguncha davom etadi , bu esa yoki ga self.end\_game\_nqo'ng'iroq qilib g'alaba qozonish jarayonini boshlaydi . Muvaffaqiyat/muvaffaqiyatsizlik har ikki holatda ham xaritani ochish va tegishli holatni o'rnatish orqali boshlanadi.game\_overgame\_won

piton

def on\_reveal(self, w):

if w.is\_mine:

self.game\_over()

else:

self.end\_game\_n -= 1 *# decrement remaining empty spaces*

if self.end\_game\_n == 0:

self.game\_won()

def game\_over(self):

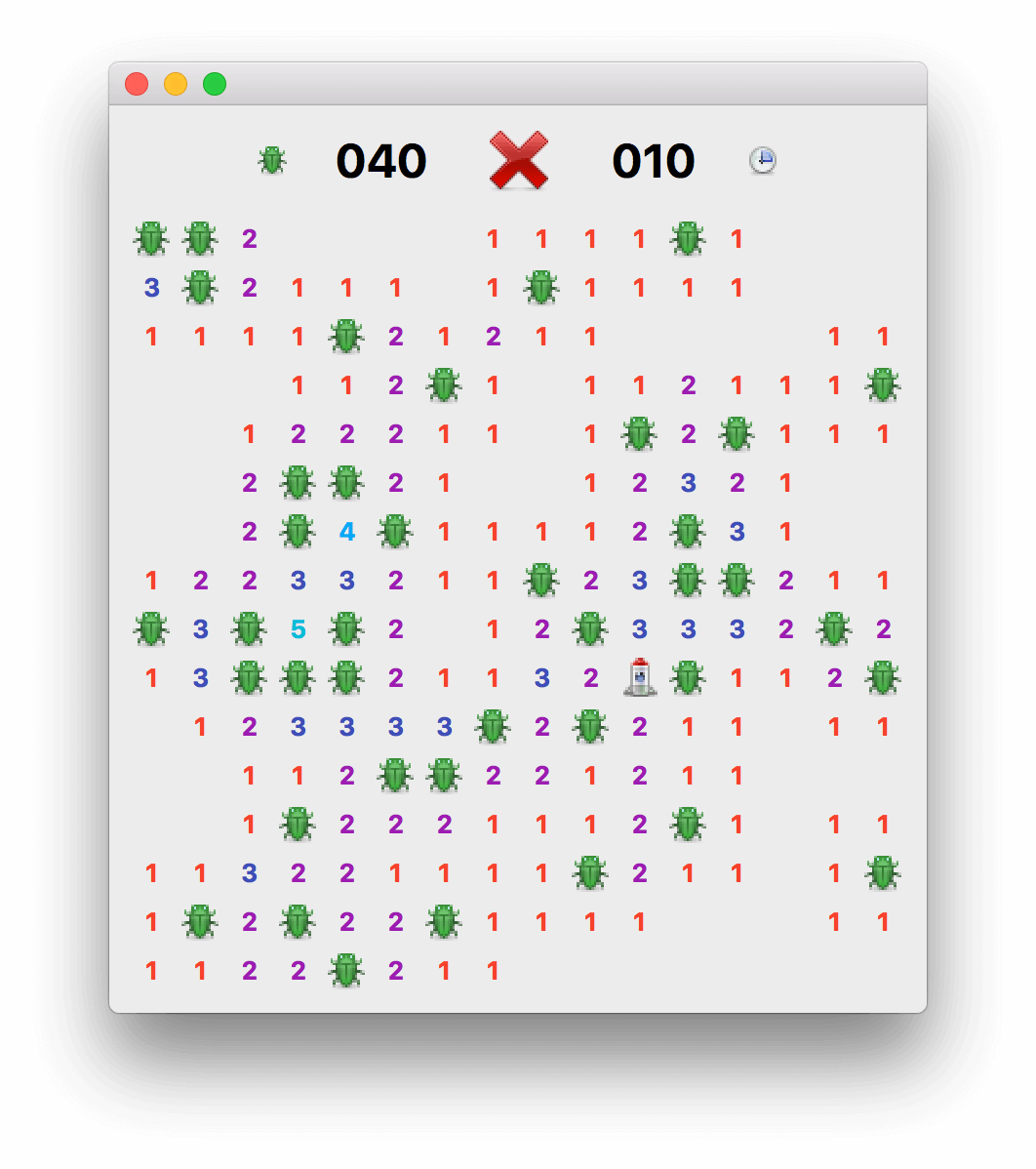
self.reveal\_map()

self.update\_status(STATUS\_FAILED)

def game\_won(self):

self.reveal\_map()

self.update\_status(STATUS\_SUCCESS)

Yo'q. B'ug tomonidan egan.

**Qo'shimcha fikrlar**

*Agar siz Moonsweeper-ni* kengaytirishni xohlasangiz , bu erda bir nechta g'oyalar mavjud -

1. O'yinchiga birinchi navbatda o'zini tutishiga ruxsat bering. Mina pozitsiyalarini hisoblashni foydalanuvchi birinchi marta bosgandan so'ng kechiktirishga harakat qiling va keyin o'tkazib yubormaguningizcha pozitsiyalarni yarating.
2. Kengashning ma'lum bir qismini avtomatik ravishda ochish uchun skaner kabi quvvatlarni qo'shing.
3. Yashirin B'ug'lar har bir burilish orasida harakatlansin. Bepul oshkor etilmagan lavozimlar ro'yxatini saqlang va B'ug'larga ularga o'tishga ruxsat bering. Har bir bosishdan keyin qo'shni joylarni qayta hisoblashingiz kerak bo'ladi.

Agar siz ko'proq ilhom olishni istasangiz, [Keyt Xollning ushbu PR-ni ko'ring](https://github.com/mfitzp/15-minute-apps/pull/11) , u boshqa narsalar qatorida startapni tanlash imkoniyatini o'zgartiradi!

Toʻliq manbani Windows, Mac va Linux uchun oʻrnatuvchilar bilan birga quyida yuklab olish mumkin.

