Sisukord

Kursuse "Teeme ise arvutimänge - algus"

6. RAAMAT

VEEL KASULIKKU JA HELI

Tiina Kull

Tartu Ülikool

2012

Heli

Pole midagi parata, aga tummad mängud ei köida. Nii olen ka mina juba oma näitena toodud Ussimängus heli kasutanud. Heli muudab mängu põnevamaks, nauditavamaks, ootamatumaks.

Heli saab olla nii sisendiks kui väljundiks. Sisendina võetud heli ei ole mitte midagi muud kui mikrofoniga salvestamine. Sellist lahendust arvutimängudes leidub aga üsna vähesel määral. Väljundina kasutatavat heli leidub aga igal pool. Väjundheliga me selles peatükis ka tegelema hakkame. Vaatame täpsemalt lühikesi heliefekte ja pidevalt taustana mängivat muusikat.

Heli nimelt on järjekordne keeruline valdkond, mis on üsna kapriisne ja ei taha mitte erinevate arvutisüsteemidega ühte moodi käituda. Täpselt sama moodi nagu ka graafika. Tuleb arvestada paljude erinevate helikaartide ja operatsioonisüsteemide eripäradega. Kuid nagu ma olen korduvalt öelnud, siis meie eest on väga suur töö ära tehtud Pygames. Meie asi on vaid seda oskuslikult kasutama õppida. Abimeheks on seekord **pygame.mixer**.

Helifailid

Enne kui programmis saab heli kasutama hakata, tuleb see heli saada oma arvutisse faili kujul. Kas salvestad ise, tõmbad internetist või kasutad mõne plaadi peal olevaid helifaile. Helifailide formaate on väga palju erinevaid. Kõige tuntumad on järgmised:

- *Wave* failid - failinimi lõpeb lühendiga **.wav**
- *Mp3* failid - failinimi lõpeb lühendiga **.mp3**
- *WMA* (*Windows Media Audio*) failid - failinimi lõpeb lühendiga **.wma**
- *Ogg Vorbis* failid - failinimi lõpeb lühendiga **.ogg**

Kõige sagedamini kasutatakse **wav** ja **mp3** faile ja seda soovitan teha ka sul:)

Helifailide kasutamiseks peavad need failid asuma kas samas kataloogis kus sinu programmgi asub, sellisel juhul saad programmis viidata vaid selle faili nimele. Kuid fail võib asuda ka mujal arvutis, kuid siis tuleb faili kätte saamiseks programmile ette anda täispikk teekond vajaliku failini. Näiteks nii:

```
taustamuusika = "h:\Program Files\Arvutimängud\muusika\taustaheli.wav"
```

Pygame.mixer

Kõigepealt, et programmis üldse saaks heli mängima panna, peame panema käima ***pygame.mixer***'i. Selleks tuleb programmi alguses, täpselt sama moodi nagu graafika kasutamise korralgi öelda, et nüüd ole valmis minu programmis peale joonistamise ka häält tegema. Selleks peab kirjutama need kolm rida:

```
import pygame

pygame.init()

pygame.mixer.init()
```

Nüüd oleme kõik ettevalmistused teinud ja võime hakata oma programmis heli mängima.

Nagu juba sissejuhatuses rääkisin, siis on võimalik teha kahte erinevat tüüpi heli. Nö lühikesed piiksatused, mida nimetatakse heliefektideks ja pikemad muusikapalad, mida saab panna taustana mängima. *Pygame.mixer*'s töötavad need kaks asja erinevate käskudega.

Lühikesed palad

Lühikesed efektid on tavaliselt .wav formaadis ja nende kautamiseks tuleb *pygame*'s kasutada **Sound** käsku. Sound tõmbab terve helifaili korraga arvutimällu ja hakkab sealt heli mängima:

```
heli_soodud = pygame.mixer.Sound('soodud.wav')

heli_soodud.play()
```

Pikad muusikapalad

Pikemad palad on aga tavaliselt mp3 formaadis. Näiteks kui sa peaksid tahtma taustaks lasta Beethoveni sümnfooniat nr5, mis on ligi 10 min pikk, siis ei ole otstarbekas tervet faili korraga arvutimällu tõmmata. Sellisel juhul tuleks kasutada otse failist mängimise võimalust ja seda tehakse *pygame*'s **music.load()**-ga:

```
pygame.mixer.music.load("Bhnr5.mp3")

pygame.mixer.music.play()
```

Näide:

Kui programm peab ainult heli mängima, siis on mõistlik enne heli käimapanemist veidi oodata, sest sageli võtab programmil *mixer*'i üles seadmine veidi aega. Näites kasutatava helifaili võid võtta siit: http://math.ut.ee/~kull/mustrastas.ogg


```
import pygame, sys

pygame.init()

pygame.mixer.init()
aken = pygame.display.set_mode([640, 480])
pygame.time.delay(1000)

heli_linnud = pygame.mixer.Sound('mustrastas.ogg')

heli_linnud.play()

while True:
    for i in pygame.event.get():
        if i.type==pygame.QUIT:
            pygame.quit()
            sys.exit()
```

Helitugevuse kontroll

Kui helifail on programmis käima pandud, siis oleks ju väga mõnus, kui saaks ise kontrollida, kui vaikselt või kui tugevalt seda heli mängitakse. *Pygame.mixer*'s on sellise kontrolli jaoks eraldi meetod **set_volume()**. Sellele meetodile tuleb anda argumendiks arv 0 ja 1 vahel, näiteks 0.4 või 0.8. Mida väiksema arvu ma sulgudesse kirjutan, seda vaiksemalt faili mängitakse, mida suurem arv kirjutada, seda kõvemini see kostub mängimise ajal ka meieni.

Järgmises koodis on heli tugevuseks määratud 0.1, kuid katseta erinevate arvudega, et teada saada, kui tugev on tugev heli ja kui nõrk on nõrk heli.


```
import pygame, sys

pygame.init()

pygame.mixer.init()
aken = pygame.display.set_mode([640, 480])
pygame.time.delay(1000)

heli_linnud = pygame.mixer.Sound('mustrastas.ogg')
heli_linnud.set_volume(0.1)
heli_linnud.play()

while True:
    for i in pygame.event.get():
        if i.type==pygame.QUIT:
            pygame.quit()
            sys.exit()
```

Taustamuusika

Taustamuusikaga on see lugu, et ükskõik kui pika muusikapala ka ei valiks, siis ma ei tea ju tegelikult kunagi ette, kui pikalt mängija minu mängu mängib. Muusika aga võiks mängida kogu aeg, sõltumata programmi kasutuse pikkusest.

Selle probleemi saab aga pygame's lahendada väga kergelt, ainult ühe arvu lisamisega. Nimelt tuleb **play()** funktsiooni argumendiks panna mistahes negatiivne arv - tavaliselt -1.

Näiteks:

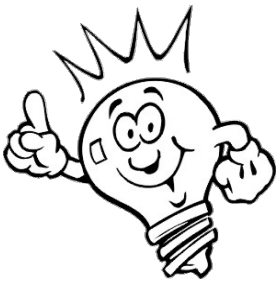
```
heli_linnud.play(-1)
```

Kui aga tahta mingit pala korrata kindel arv korda, näiteks koputuse heli kolm korda, siis tuleb funktsioon play() sisse kirjutada lihtsalt see arv, mitu korda tahetakse helifaili kordamist kuulda.

Näiteks:

```
heli_linnud.play(3)
```

Veel kasulikku



Järgnevate peatükkide alla ma koondasin materjali, mis võiksid olla veel kasulikud sinu mängu jaoks. Kõiksugused tavalised põrkamised ja ajaloendajad ja eludeloendajad oleme juba läbi vaadanud, nendega said tutvuda kursuse jooksul tasapisi valminud ussimängu erinevates etappides. Siin aga annan sulle veel ülevaate sellest, kuidas luua edetabel mängijate tulemustest ja samuti, kuidas näiteks graafilises mängus küsida kasutajalt andmeid.

Huvitavaid avastusi!

Edetabel - faili kirjutamine

Järgnev programm genereerib kasutajale juhusliku punktisumma ja laseb seejärel kasutajal tulemus edetabelisse panna. Tabelisse lähevad kirja ainult kolm kõige paremat tulemust. Kuidas see käib?

Avatakse *edetabel.pickle*-nimeline fail, kus on kirjas kõik eelnevad tulemused, kontrollitakse, kas uus tulemus on parem kui need, mis seal enne olid ning lisatakse tulemus. Õige koha leidmine käib listi kaudu - seal saab sorteerida vastavalt punktide järgi. Kui list on sorteeritud, siis kuvatakse ekraanile järjest kõik tulemused. Kui aga enne ei ole ühtegi korda edetabelit kasutatud, siis selline fail alguses puudub. Sellisel juhul programm ise loob edetabeli-nimelise tavalise faili ja hakkab sinna andmeid lisama.

```
# -*- coding: utf-8 -*-

from random import randint
import pickle

edetabeli_suurus = 3
koht_edetabelis = None
try:
        fail = open('edetabel.pickle','rb')
        edetabel = pickle.load(fail)
        fail.close()
except:
        edetabel = []

def tryki_edetabel():
        print('Koht\tPunkte\tNimi')
        for i in range(len(edetabel)):
                (punkte,nimi) = edetabel[i]
                print(str(i+1)+'\t'+str(punkte)+'\t'+nimi)

tryki_edetabel()

punkte = randint(0,100)
print('Said '+str(punkte)+' punkti')

edetabel.append((punkte,''))
edetabel.sort(reverse=True)
edetabel = edetabel[:edetabeli_suurus]
try:
        koht_edetabelis = edetabel.index((punkte,''))
        print('Said edetabelisse kohale '+str(koht_edetabelis+1))
        print('Sisesta oma nimi:')
        nimi = input()  #PYTHON 2-s  raw_input()
        edetabel[koht_edetabelis] = (punkte,nimi)
        print('Uus edetabel:')
        tryki_edetabel()
except:
        koht_edetabelis = None
        print('Kahjuks ei saanud edetabelisse')

fail = open('edetabel.pickle','wb')
pickle.dump(edetabel,fail)
fail.close()
```

Selle sama faili .py versiooni saad kätte ka siit: http://math.ut.ee/~kull/edetabel.py

Graafilises keskkonnas nime küsimine

See programm siin on aga näidis sellest, kuidas pygames saab küsida kasutajalt nime ja hiljem sisestatud nimi ka ekraanile trükkida.

```
from pygame import *

init()
aken = display.set_mode([640,480])

varv_taust = (255,255,255)
varv_kast = (255,0,0)
varv_tekst = (0,0,255)
kast = Rect((0,0),(360,120))
kast.center = aken.get_rect().center
try:
        fondi_asukoht = font.match_font('chalkboard')
except:
        fondi_asukoht = None
font_nimi = font.Font(fondi_asukoht,100)
font_sisesta = font.Font(fondi_asukoht,50)
font_tere = font.Font(fondi_asukoht,50)
pilt_sisesta = font_sisesta.render('Sisesta siia oma nimi:',1,varv_tekst)
pilt_sisesta_raam = pilt_sisesta.get_rect()
pilt_sisesta_raam.midbottom = kast.midtop

def joonista():
        aken.fill(varv_taust)
        aken.blit(pilt_sisesta,pilt_sisesta_raam)
        draw.rect(aken,varv_kast,kast,3)
        pilt_nimi = font_nimi.render(nimi,1,varv_tekst)
        pilt_nimi_raam = pilt_nimi.get_rect()
        pilt_nimi_raam.center = kast.center
        aken.blit(pilt_nimi,pilt_nimi_raam)
        display.flip()

nimi = ''
joonista()
while True:
        e = event.wait()
        if e.type == KEYDOWN:
                if e.key == K_BACKSPACE:
                        nimi = nimi[:-1]
                elif e.key == K_RETURN:
                        break
                elif e.key <= 127:
                        if key.get_mods() & KMOD_SHIFT:
                                nimi += chr(e.key).upper()
```

```
                        else:
                                nimi += chr(e.key)
                pilt_nimi = font_nimi.render(nimi,1,varv_tekst)
                if pilt_nimi.get_rect().width > kast.width:
                        nimi = nimi[:-1]
                joonista()
        if e.type == QUIT:
                quit()
                exit()

pilt_tere = font_tere.render('Tere, '+nimi+'!',1,varv_tekst)
pilt_tere_raam = pilt_tere.get_rect()
pilt_tere_raam.midtop = kast.midbottom
aken.blit(pilt_tere,pilt_tere_raam)
display.flip()

while not event.get(QUIT):    # kuni kasutaja pole sulgenud akent
        time.delay(100)
quit()
```

Selle sama faili .py versiooni saad kätte ka siit: http://math.ut.ee/~kull/sisestus.py

Kaubanduslik ussimäng

 Ei saanud ikka kiusatusele vastu panna ja koos teiega ka oma ussimängu ikka veel ja veel tuunida. Nüüdses versioonis on kasutatud taustana fotot ja eelnevates peatükkides kirjeldatud edetabelit ja nime sisestamise võimalust. Samuti mängib terve aeg mahe taustamuusika. Kuna programm on nüüdseks juba väga pikaks veninud, siis annan teile lingi koodile, et saaksite ise katsetada, ideid ammutada ja koodi näppida. Küsimuste korral kirjuta foorumisse!!!

Põnevat mängimist!

http://math.ut.ee/~kull/uss10mets/

Kuidas arendada ennast iseseisvalt edasi?

Selle peatükiga lõpeb selle kursuse materjal ära ja päris kurb oleks kui ka sinu siiani väga tubli areng koos kursuse lõpuga otsa leiaks. Seepärast toon sellel lehel sulle välja päris mitmeid võimalusi, kuidas selles vallas sammuke jällegi edasi minna.

- Mängude teemaga saab edasi tegeleda veebis tasuta saadava raamatu kaudu (mis nagu kõik meie ümber, on inglise keeles). See raamat on päris hästi kirjutatud ja pikalt lahti seletatud näidetega, seega iseseisvaks õppimiseks päris hea. Võrreldes kursusega ei pea seal aga kahjuks tegema kodutöid ja ei saa ka foorumitest küsida. Raamat asub siin: http://inventwithpython.com/chapters/

- Mängudega saab edasi tegeleda ka osaledes mängude tegemise võistlustel. Sul peaks praeguseks juba piisavalt algteadmisi olema, et sellisest üritusest osa võtta, pealegi on ettevalmistuda aega veel vähemalt pool aastat. Võistlus toimub igal aastal (see aasta aga on kahjuks juba läbi, kuid sügisest algab jälle) ja auhinnad on seal päris meeldivad:). Täpsema info leiad siit: http://www.playtech.ee/?nav=news&news=264

- Võid iseseisvalt tutvuda Tartu Ülikooli informaatikaõppe esimese kursuse (ja miks mitte ka teiste kursuste) materjalidega. Näiteks programmeerimise kursus esmakursuslastele (neile, kes pole veel programmeerimist õppinud): http://courses.cs.ut.ee/2011/programmeerimine/Main/HomePage ja ka teisi materjale: http://courses.cs.ut.ee/

- Kui sa oled 12. klassi õpilane, võid kaaluda sügisel informaatikaõppimist süvendada näiteks Tartu Ülikoolis arvutiteaduse instituudis: http://www.cs.ut.ee/. Hästi õppides tagad endale tulevikus vägagi tasuva töö ja pealegi huvitava töö.