Farmos játék programozói dokumentáció

Felépítés

A program több modulból áll. Minden funkcionalitás saját modullal rendelkezik. A dokumentáció tartalmazza az alábbi modulokat, és funkcióikat:

- main.py
- button.py
- nevbeker.py
- gameM.py
- feladat.py
- load.py
- plantloader.py
- eszkozok.py
- scoreboardM.py
- fajlkezelo.py

Főmodul: main.py

A main.py a program főprogramja. Itt fut a main függvény, illetve itt rajzolódik ki a menürendszer.

A program a Pygame package-et használja a megjelenítéshez.

A főmodulban futnak össze a modulok, így az összes helyi meghatározás itt van deklarálva az importer függvénybe foglalva, ami még a main függvény előtt meghívódik, hogy hozzáadja a felhasznált mappákat a főprogramhoz a sys modul segítségével, így a modulok sikeresen betöltődnek a programba.

A játékot egy játékciklus működteti, ami a kilépés parancsig, folyamatosan ellenőrzi a bemenetet, illetve rajzolja ki a kimenetet az ablakra.

Ebben a modulban fut le az egyik legfontosabb parancs: a pygame.init(), ami működésbe hozza a pygame csomagot.

A főmodul főleg a menürendszerért, illetve a fájlmentésért felel.

Kirajzolja a hátteret, illetve a gombokat, amiket a button.py modul segítségével működtet.

A főmodul hívja meg a két legfontosabb modul, a gameM.py illetve a scoreboardM.py, main függvényét. Ezeknél a hívásoknál továbbadja 2 fontos benne definiált változót, a displayt és a clock-ot.

Gombok kezelése

A main függvény a felhasználó interakcióit egy listában kapja meg a pygame modultól. Ebben a listában vannak a leütött billentyűk, avagy gombok. Az egér helyzetét is a pygame modul szolgáltatja. Ezeket az adatokat feltételekkel tesztelve tesz változtatásokat a button.MenuButton típusú gombok esetén.

set():
A set függvény bekér egy pygame sprite groupot, egy gomot illetve egy állapotot, majd megváltoztatja a gomb kinéueti állapotát. A menügombok esetében 0 és 1 es állapot áll rendelkezésre 0 -> normális, 1 -> amikor az egér felette van, színesedő állapot.

Ez felhsaználja a button modulban levő MenuButton osztály set() függvényét is.

Fájlkezelés:
A fájlkezelést a főmodul különböző almodulok közt osztja szét a gameM.py modul szolgáltatja a játékos pontszámát, míg a nevbekermodul.py egy külön oldalt nyit, ahol a játékos megadhatja a nevét. A fajlkezelo.py modul Toplista osztályában történik a fájlkezelés. A main.py modul összehangolva ezeket kiosztja az egyes feladatokat a kisebb moduloknak.

Gombkezelő: button.py
A gombkezelő modul 2 osztályt tartalmaz

Egy Button osztályt, ami egy sprite, vagy a gomb egy lehetséges állapota, illetve egy MenuButton osztályt, ami több ilyen Button osztályt tartalmaz magában a sokszínű gombok eléréséhez.

MenuButton.__init__():
A menubutton konstruktorával hozza létre a gombokat. Inputként bekér 2 listát, ami a 2 állapot létrehozásához szükséges adatokat tartalmazza. Létrehoz egy gombot 2 állapottal.

self.get():
A MenuButtonon belüli act változó alapján visszaküldi az aktuális állapotot

self.set():
Bekér egy számot, amivel az act változót átállítja az osztályon belül

self.move():
Bekér egy pozíciót, majd áthelyezi a gombot oda.

self.hover():
Bekéri egy objektum (általában egér) helyét, visszatér egy logikai értékkel, ami megállapítja, hogy az objektum a gomb felett helyezkedik-e el.

Button:
Hagyományos pygame sprite, felülettel és rectangular-al, nincs egyéb tagfüggvénye, hiszen csak egy állapotot jelképez.

Névbekérő: nevbeker.py
A játék után ez a modul kéri be a játékos nevét.

Alap
Az Alap() pygame sprite osztály egy egyszerű kék téglalapot hoz létre, ami megjelöli a játékos nevének helyét az ablakon.

Ez a modul a gépeléssel van elfoglalva. A felhasználó itt gépeléssel vihet be adatot, amivel majd visszatér a main.py modulba

Felhasználja ehhez a load modult.

Játékmodul: gameM.py

Ez a modul foglalkozik a játékkal magával. Ez a modul összefűz néhány fontosabb játékbetöltő modult.

- load
- plantloader
- eszkozok
- feladat

A benne levő main függvény átveszi a főprogram kimenetét, illetve az óráját, így nem kell újat létrehozni.

Itt is játékciklus van használatban, ami kilépésig, vagy a játék végéig fut, teszteli a bemenetet, és az alapján változtatja a kimenetet a vászonra.

Két osztály van benne definiálva:

Timer:

Ez az osztály felel a játékon belüli időzítésért.

Ez az időzítő csak játékos hátralevő idejét méri.

Konstruktor:

Létrehoz 3 változós objektumot:

Az első változó helyére kerül a jelenlegi idő

A második helyére a jelenlegi + a hátralevő idő

A harmadik helyére pedig egy load.Text objektum, ami a szövegért felel.

self.get():

Visszaadja másodpercben, hogy menni idő van hátra.

self.setOut():

Frissíti a szövegkimenetét az időzítőnek

self.move()

Elhelyezi az időzítő kimenetét a vásznon.

self.timerSet()

Bekéri a jelenlegi időt, illetve a hátralevő időt, majd beállítja az időzítőt

self.runTimer()

Bekéri a jelenlegi időt. Visszatér igaz értékkel, ha letelt a hátrelevő idő

self.add():

Bekér egy időtartamot másodpercben, majd hozzáadja a hátralevő időhöz.

Pontszam:

Ez az osztály felel a játékos pontjaiért.

A megjelenítése megegyezik a Timer-ével, ugyanúgy a load.Text-osztályt használja.

Konstruktor:
Létrehoz egy kimeneti változót a load.Text segítségével, illetve egy másik változót, ami a játékos pontjait tárolja.

self.get():
Visszaadja a játékos jelenlegi pontjait.

self.add()
Bekér egy egész számot, majd hozzáadja a pontok számához.

self.move()
Áthelyezi a hozzá tartozó szöveget a vásznon.

Függvényei

mezokeres():
Bekér egy pozíciót, és egy kert objektumot, és megnézi, hogy a pozíció alatt melyik parcella van, amit vissza is ad.

buy():
Bekéri a parcella koordinátáit, egy növénytípust, illetve a kertet amiben a parcellát keresni kell, majd a növénytípust beleülteti a kertbe

novenyNoves():
Ez egy frissitő függvény, ami minden ciklus végén meg van hívva. A növényekbe beépített időzítőt frissíti ezáltal elindítva vagy megszakítva a növény növést.

idoIgeny():
Ez a függvény tárolja a növények időigényeit. Bekér egy objektum típust, és visszaadja a jelenlegi növés / elrohadási időt.

vizIgeny():
Ez a függvény tárolja a növény vízigényét logikai formátumban. Hívás esetén kér egy növény objektumot, és visszaadja, hogy kell-e locsolni vagy nem.

Vizcsepp():
Bekér egy leültetett növény objektumot és egy megjelenítő pygame sprite groupot, és kirajzol egy vízcseppet a növény felé

Locsol():
Bekér egy koordinátát, illetve egy kertet, és a növény locsolt állapotának logikai értékét igazra állítja

Kapal():
Bekér egy koordinátát, illetve egy kertet, és kiüríti a mezőt.

Arat():
Bekér egy koordinátát, egy kertet, egy feladat objektumot, és egy pontszámot.

A megfelelő pontszámot hozzáadja a pontokhoz, teljesíti a feladatot, majd a kapal függvényt használva kiüríti a mezőt.

Aratojel():
Bekér egy növény típust és egy pygame sprite groupot, és megjelenít egy aratójelet a növény felett.

Betöltő modul: load.py
A betöltőmodul foglalkozik a legfontosabb grafikai elemek betöltésével. Ez rajzolja ki a kertet, sok helyen ez ír ki szöveget, továbbá minden nem növény panel és elem itt kapott helyet.

Feladatpanel, Eszkozpanel, Boltpanel, Boltbutton, Text és Background: mind olyan osztály, amik egyszerű megjelenítést végeznek. A Pygame könyvtár alapfüggvényeit használva.

Parcella:
Olyan pygame sprite, ami rendelkezik egy ultet() tagfüggvénnyel. Ez a tagfüggvény képes az osztályon belülre elhelyezni egy másik osztályt amit a self.noveny helyen tárol el.

Kert:
A kert a parcella feletti egység. Tagfüggvényivel lehet elérni a kertben levő parcellák adatait. Konstruktorja tetszőleges méretű, felbontású 2 dimenziós mátrixot tud létrehozni vele.

Self.update():
Bekéri a kert növényeinek megjelenítő pygame sprite group-ját, majd minden elemet felfrissít benne: eltávolítja az összes elemet majd a self.getnoveny() függvénnyel visszateszi az összes aktív növény állapotát.

Self.get():
Kikeresi a kert adott mezőjét Bekéri hozzá a mező koordinátáit.

Self.getnoveny():
Bekéri a parcella koordinátáit, és visszaadja a rajta levő növény referenciáját.

Self.getAllNoveny():
Listába téve visszaadja a kert összes növényét.

Self.cellakeres():
Bekér egy helyzetet, és visszatér az ott levő parcella koordinátáival.

Self.draw():
Bekéri a kert helyét a vásznon, illetve 2 parcella közti térközt, majd elrendezi a parcellákat az adatoknak megfelelően.

Self.hover():
Bekér egy pozíciót, illetve egy parcella koordinátáit, majd visszatér egy logikai értékkel, hogy a parcella tartalmazza-e a koordinátát magában.

A növénybetöltő: Plantloader.py
A növénybetöltő modul foglalkozik a növények betöltésével. Az összes növényt innen kéri le a program. Benne szerepelnek a különböző típus osztályok répától paradicsomig, amik egy listában tárolják a növények állapotainak a képeinek a helyét. Ezeknek a szülőosztályuk a Novenyinit osztály, ami sok Allapot osztályt hoz létre a gyermekosztálytól kapott lista alapján. Betölti a képeket az állapotokba és tagfüggvényeket biztosít ezeknek változtatásához. Egyik legfontosabb tagfüggvénye a

self.get() mivel az aktuális állapotot ez közvetíti a pygame sprite group-nak, ami megjeleníti a növényt.

Az eszközbetöltő: eszkozok.py

Az eszközbetöltő modul foglalkozik a játékban megjelenő eszközökkel: kanna, kapa, sarló, illetve a jelzőkkel: vízcsepp, aratójel. Az eszközök mind pygame sprite-ok jellegzetességük, hogy rendelkeznek egy helymentes() és egy visszateres() függvénnyel. Ezek miatt tudnak szabadon mozogni, majd visszatérni az eredeti helyükre. A jelzők nagyon primitív sprite-ok ők csak egy megjelenítést végeznek, nem mozognak.

A feladat modul: feladat.py

A játék egyik legfontosabb eleme a feladat. A Feladat osztály a játékban egy listában tárol 2 elemű tuple értékekben egy számot és egy terménynevet, amiből létrehoz magában egy self.disply nevű listát, amiben a FeladatIkon típusú pygame spriteokat tárol, illetve rendez el folyamatosan. Ha elkészülnek a feladatok a játékos által akkor a listából kikerülen az elvárások, a kirajzolás pedig frissül., így folyamatosan a feladatpanel tetejéhez tartanak a feladatikonok.

A feladatmodulba tartozik a feladat generátor függvény, ami egy számot kér be nehézségként, majd egy Feladatot ad vissza. A feladat a feladatparaméterek súlyozásából áll össze. Addig pörög a ciklus a random.randint() függvény felhasználásával, amíg nem készül egy olyan feladat, amin ha összeadjuk a paraméterekhez járó súlyokat akkor a nehézség 2 es körzetébe kerüljön a végeredmény. A súlyokat a függvény tárolja.

A Toplista: scoreboardM.py

A toplista modul nem végez közvetlenül fájlkezelést, viszont a főmodultól átvesz egy bizonyos rendszer változót, aminek a típusa fajlkezelo.Toplista. Ebben már a fájlkezelések el vannak végezve, ezért ez a modul már csak a kiírással foglalkozik. Felhasználja a load modul Text osztályát, hogy 10 sornyi szöveget generáljon, és a rendszer változó listaki() függvényével egy listába megkapja a Toplista legjobb 10 eredményét, amit egy ciklussal betölt a helyükre, majd a dx, dy változók segítségével elrendezi őket.

A fájlkezelő: fajlkezelo.py

A fájlkezelő modul végzi az összes fájlkezelést. A Toplista osztály van definiálva benne, ami rendelkezik az alap fájlkezelésre alkalmas eszközökkel, tagfüggvényekkel. Az olvas függvény ellenőrzi az adatbázist, javítja a hibás sorokat, illetve létrehozza a fájlt, ha nem létezik.

A listaki() függvényre van szükség a toplistára való kiíráshoz.

Illetve tartalmazza a rendez(), ment(), beszur() tagfüggvényeket amiket kombinálva, hatékonyan lehet a fájlban módosításokat végrehajtani.