Házi feladat

Programozás alapjai 2.
Tervezés

Heltai Kilián

2025. 04. 07.

A főprogram

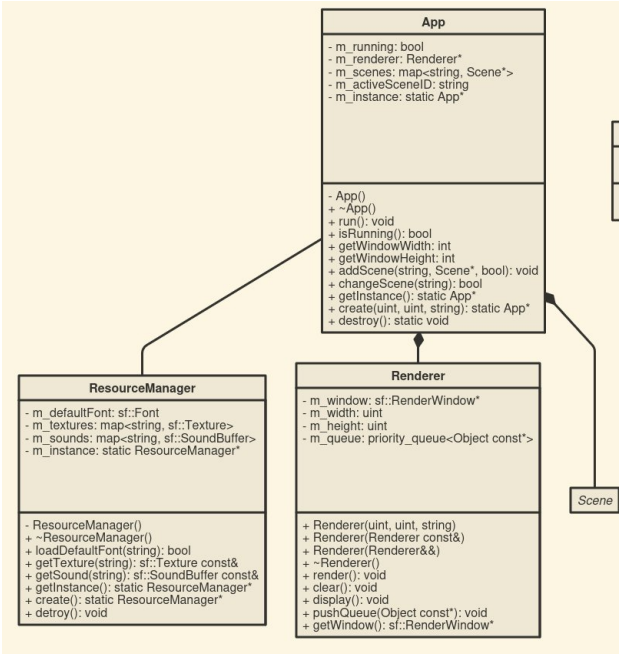
Egy *App* nevű singleton osztály felel a legmagasabb szinten mindenért. A *main()* ennek az osztálynak a *run* metódusát hívja meg.

Az *App* felelős a renderer és erőforrás-kezelő létrehozásért. Emellett a különböző nézeteketért is felelős, kezeli a nézetváltásokat, s az események kezelését nekik delegálja. Ezeket az alap nézeteket a *main()* adja hozzá, így bárki más számára is egyértelmű hogy hogyan kell ezeket létrehozni. A program ezzel a filozófiával lesz létrehozva: egyszerűen módosítható, s új dolgokat (nézetek, entitások, GUI elemek stb.) egyértelműen lehet kreálni.

A renderer-t kívülről nem lehet elérni, s az *App*on kívül nem ajánlott még egyet példányosítani. Nagyon szorosan az *App* része. A renderer egy FIFO listát tart számon a megjeleníteni kívánt objektumokról. Ezeknek az objektumoknak az élettartamát a megfelelő nézet kezeli. (*Ennek a diagramja a következő nagy részben lesz.*)

Az ablak, ami renderer-el, fix méretű: 16:9-es képarány.

Hibakezelés a főprogramban: ha egy olyan nézetet próbálunk hozzáadni a listához, aminek az azonosítója létezik, akkor egy egyedi *SceneError*-t dob megfelelő hibaüzenettel.



Erőforrás-kezelő

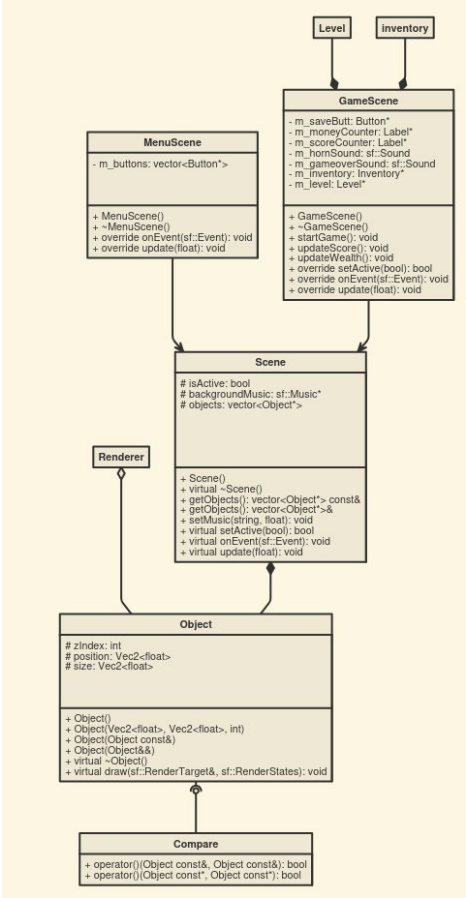
Szintén egy singleton osztály. Ez az osztály felel minden külső helyről betöltött forrásért, a szöveges konfiguráció és mentésfájlokat leszámítva. Biztosítja, hogy ugyanazt a képet/hangfájlt ne kelljen többször betölteni, ha több osztálypéldány is használná ugyanazt.

A textúrák és hang buffer-ök asszociatív tárolókban lesznek elmentve, referálni rájuk az elérési útvonalukkal lehet. Ha már létezik, akkor szimplán visszatér egy referenciával, ha még nem, akkor először megpróbálja betölteni.

Ha véletlen nem találja a keresett erőforrást, vagy pedig betöltés közben akad hiba, akkor az SFML által dobott hibát továbbítja.

Nézetek és objektumok

A főprogram nézetekre van felbontva, méghozzá kettőre: egy menüre és magára a játékra. A nézetek felelősek a megjelenítendő objektumokért illetve az események kezeléséért. Ilyen esemény például egy kattintás az egérrel.



Menü

A menü a logóból, háttérképből és két gombból áll. Az egyik gombbal egy új játékot lehet kezdeni, a másikkal pedig egy mentett állást lehet betölteni (már ha létezik).

A gombokat egy külön listában is tárolja, mert ezeket később is el kell tudnia érni keresés nélkül. Viszont az *m_buttons* tároló nem felelős az élettartamukért, az még mindig az *m_objects* dolga!

Játék

A játéknézet direkt módon nem tartalmazza a játékállást és nem kezeli azt. Ahogy a menü, ez a nézet is elsősorban a vizuális elemeket és a gombokat kezeli.

A játéklogika és az entitások kezelésére egy *Level* nevű osztály van kitalálva, amelynek működését később taglalom. Hasonlóan, a letehető tornyokat tartalmazó „eszköztárat” egy *Inventory* nevű osztály tartalmazza.

Kérdéses, hogy az *Inventory* valójában a nézethez vagy a *Level*hez tartozik hozzá inkább. Úgy döntöttem hogy inkább a nézethez, az *Inventory* független magától a pályától, mert mindig ugyanazon tornyokat tartalmazza. Viszont az is igaz, hogy pályából csak egyféle van (maga a *Level* példánya), így talán mindkét megoldás helyes.

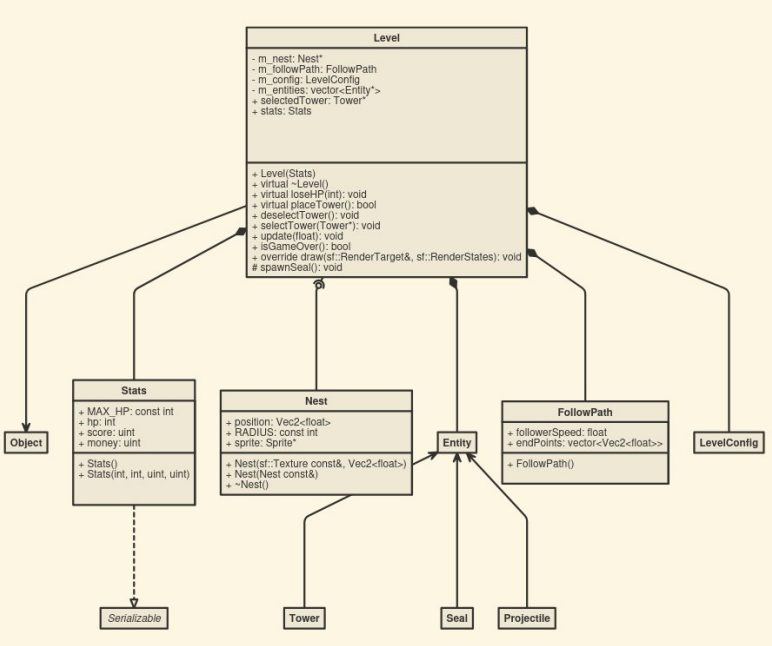
Pálya

A pálya (*Level*) is egy *Object* lesz, mivel szükséges, hogy implementálja a *draw* metódust.

Kettő belső osztálya van: *Nest* és *Stats*. A *Nest* a fészekért felel, ezt kell elérnie az ellenfeleknek. A *Stats* pedig általános, menthető statisztikát (játékállást) foglal magában, mint pont, HP és pénz.

Fontos része még a *FollowPath*. Ez tartalmazza azt az utat, amin keresztül a fókák a tojáshoz jutnak. Ezt a követési utat minden példányosított fóka megkapja referenciaként. Az út első pontja a képernyőn kívülre kell, hogy essen, hogy a játékos ne lássa a spawnolást.

A fészek pozíciója és a követési útvonal pontjai egy *level.conf* fájlba vannak eltárolva. Ennek szintaxisát később részletezem.

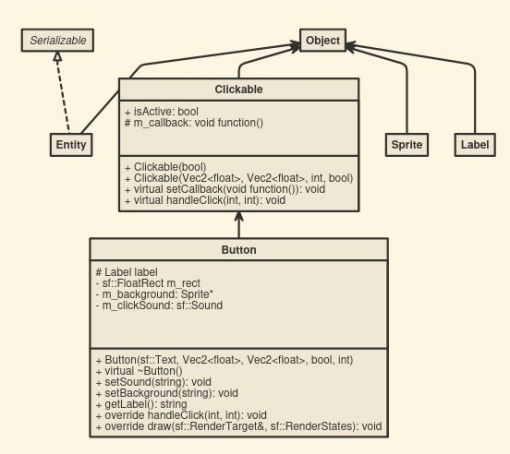


Objektum fajták

Léteznek képek (*Sprite*), címkék (*Label*), gombok (*Button*), illetve entitások (*Entity*).

Többek közt a gombokhoz lesz létrehozva egy speciális objektum alosztály, a *Clickable*. Ez ellenőrzi és kezeli azt az esetet, ha az objektumra rákattint a felhasználó. Az egér pozíciója egészen az *App run* metódusából lesz lepasszolva az objektumok szintjére.

A GUI elemek elhelyezésének segítésében az *App*-tól le lehet kérdezni az ablak méreteit (még ha az fix is).



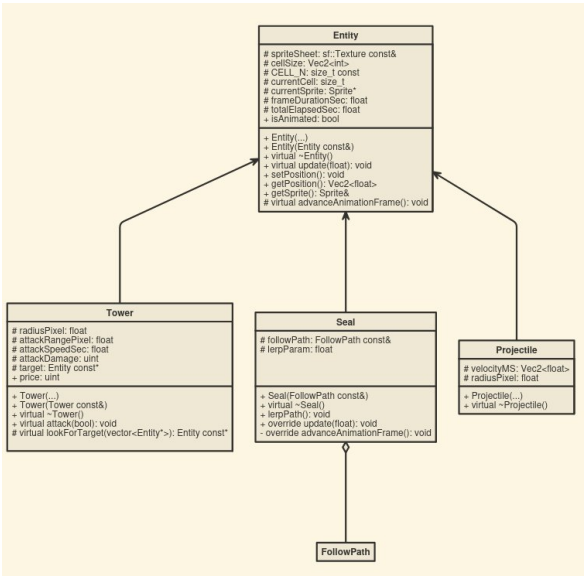
Entitások

Az *Entity* osztály őse minden olyan másik osztálynak, amelynek példánya egy torony, ellenfél vagy éppen lövedék. A lövedékek kivételével ezek a dolgok mind elmenthetők.

Minden entitásnak lehet animált sprite-ja. De például a fókáknak nincs. A *Seal* osztály így írja felül: nekik nem a sprite-ja változik, hanem a sprite magassága lesz „meganimálva;” egy kicsit összenyomódik, majd megnyúlik (szinuszosan).

Tornyok

Egy tornyot a játékos az *Inventory*-ból tud kiválasztani, ha van elég pénze rá. A tornyokat csak a pálya egy kijelölt részére lehetséges letenni, és akkor sem lehetnek egymáshoz túl közel.



Csak akkor kezdenek támadni (és kezdik meg az animációt), amikor egy megfelelő ellenséget találnak a környezetükben.

Két fajta torony (pingvin) létezik: hógolyó dobáló és egy közelharci jégcsap kardos. A hógolyó dobálónak sem végtelen a range-je, de jóval nagyobb mint a másiknak.

Ha a hógolyós talál egy megfelelő célpontot, akkor az egy konstans sebességgel mozgó hógolyót indít meg felé. A jégcsap kardos pedig instant lesebez egyet a célpontjából.

Fókák

Egy fóka spawnolásának minden frame-ben van egy fix százalékos esélye. A létrehozandó fóka típusát véletlenszerűen választja ki a *Level* osztály *spawnSeal* metódusa.

Útjukat a *FollowPath*-juk első pontján kezdik meg, és lineárisan interpolálnak végig, majd pedig a végén (miután egy tojást megettek) visszafordulnak.

Mentés

Minden osztály, amely implementálja a *Serializable* interfészt, menthető. A megfelelő adatik egy *save.dat* fájlba kerülnek.

Csak a „Mentés & kilépés” gomb megnyomásával következik be! A mentett állás törlődik, ha a játékos veszít.

Configok

A pálya alapkonfigurációjának és a mentett játék betöltésére lesz létrehozva egy-egy egyedi fájlbeolvasó. Ezek a fájlok egyedi szintaxissal rendelkeznek.

Mindkettőben el lehet helyezni kommenteket: ha egy sor „*#*”-kal kezdődik, akkor az utána levő szöveg a sor végéig ignorálva lesz.

level.conf: Név-érték párokat lehet eltárolni, amelyeket „:” választ el. Az értékek csak egész szám párok lehetnek, vagy egy lista amely ilyen párokat tartalmaz. Két attribútum nevet fog felismerni: „*nestPosition*” és „*followPath*.” A „*nestPosition*” egy párt vár el, míg a „*followPath*” egy listát.

- Első sor (fix szöveg): „**levelconf**”
- Többi sor: <***komment*** | [**attribName** szöveg] „:” [[**x** egész szám] [**y** egész szám] | „**[**” [**x** egész szám] [**y** egész szám] **...** „**]**”]>

save.dat: *A specifikációban van taglalva.*

Tesztelés megkönnyítése

Az SFML által biztosított *Vector* és *Rect* típusok helyett saját, sablonosított típusokat hozok létre. Így ezeket a tesztkörnyezetben fel tudom használni.