ANALÍZIS MODELL

Tartalomjegyzék

1. Bevezetés

2. Kezdeti osztálydiagram

2.1 Osztálydiagram

2.2 Osztályok felsorolása

2.3. Alrendszerek

3. Az első alrendszer modellje

3.1 Statikus modell

3.1.1 Kapcsolatok pontosítása

3.1.2 Attribútumok azonosítása

3.1.3 Bázisosztályok keresése

3.2. Dinamikus modell

3.3. Funkcionális modell

3.4. Operációk azonosítása

3.5 Az analízis modell osztálydiagramja.

3.6. Az analízis modell osztályainak listája.

3.6.1 Első osztály neve

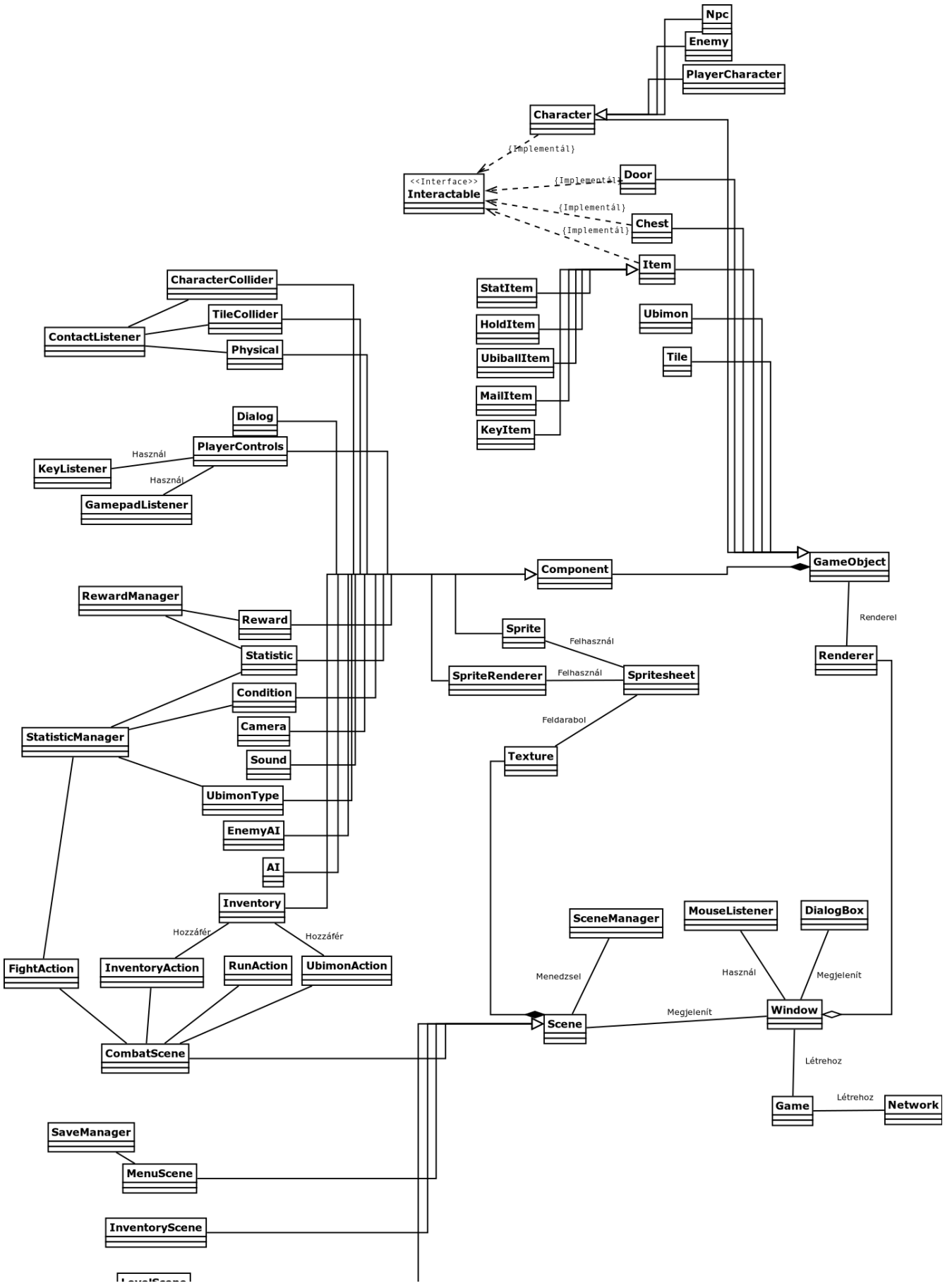
3.6.2. Második osztály neve.

X. Mellékletek

1. Bevezetés

2. Kezdeti osztálydiagram

2.1 Osztálydiagram


Npc
Enemy
PlayerCharacter
Character
{Implementál}
<<Interface>>
Interactable
Door
{Implementál}
{Implementál}
Chest
{Implementál}
Item
CharacterCollider
TileCollider
ContactListener
Physical
StatItem
HoldItem
UbiballItem
MailItem
KeyItem
Ubimon
Tile
Dialog
PlayerControls
Használ
KeyListener
Használ
GamepadListener
Component
GameObject
RewardManager
Reward
Statistic
Condition
Camera
Sound
StatisticManager
UbimonType
EnemyAI
AI
Inventory
Sprite
Felhasznál
SpriteRenderer
Felhasznál
Spritesheet
Renderel
Renderer
Feldarabol
Texture
Hozzáfér
Hozzáfér
FightAction
InventoryAction
RunAction
UbimonAction
SceneManager
MouseListener
DialogBox
Menedzsel
Használ
Megjelenít
Megjelenít
Scene
Window
CombatScene
Létrehoz
Game
Létrehoz
Network
SaveManager
MenuScene
InventoryScene

2.2 Osztályok felsorolása

Component: Absztrakt osztály, amely egy adott GameObjecthez tartozik, tartalmazza a komponensehez kapcsolódó alapvető függvényeket és adatokat.

A Component absztakt osztályt kiterjesztő osztályok:

CharacterCollider: A karakterek ütközéséért felelős osztály.

TileCollider: A "csempékkel" történő ütközésért felelős osztály. (Hogy ne lehessen rajtuk áthaladni)

Dialog: A karakterek szövegéért felelős osztály.

PlayerControls: A játékos inputjához játékban történő történést rendel hozzá.

Reward: A karakterek által harc után kapott itemekért felelős osztály.

Statistic: A karakterek szintjéért, életerejéért, sebzéséért, stb felelős osztály.

Condition: Enum osztály, amely felsorolja a harcban szerezhető kondíciókat.

Camera: A játékost követő kamera, amely a játéktér "mozgatásáért" felelős.

gameobjecthez tartozó hangokért felelős.

Sound: A gameobjecthez tartozó hangokért felelős.

UbimonType: Enum osztály, amely felsorolja az ubimon típusokat.

EnemyAI: Az (nem játékos karakter) ellenség harcban használatos logikáját határozza meg.

AI: A nem játékos karakterek viselkedéséért felelős osztály.

Inventory: A karakter által birtokolt tárgyakért és használatukért felelős osztály.

Sprite: A gameobject spritejának meghatározásáért és a hozzá tartozó függvényeket tartalmazza.

SpriteRenderer: Külön renderer a karakter gameobjecteknek, amelyeknek több spritejuk van. (animációhoz)

GameObject: Absztrakt osztály, amely a játékban található objektumokról fog információkat tárolni, mint pl.: név, komponensek listája, id.

A GameObjectet leszármaztató osztályok:

Door: Ajtó objektum, amely tartalmazza, hogy melyik kulcs item képes azt kinyitni. Implementálja az interactable interfészt. Hozzátartozó komponensek: Sprite, Sound, TileCollider.

Chest: Láda, amely itemeket tartalmaz. Implementálja az interactable interfészt. Hozzátartozó komponensek: Sprite, Sound, TileCollider.

Tile: Textúrával rendelkező "csempe" amelynek van Ütközése. Hozzátartozó komponensek: Sprite, TileCollider.

Ubimon: A harcban a karakterek által használt objektum, A játékos által használt ubimonok száma nem nagyobb, mint a játékos birtokában lévő ubimonlabdák száma. Hozzátartozó komponensek: Statistic, Condition, Sound, UbimonType.

Item: Az adatbázissal való kapcsolatért felelős (id alapján az item attributumait megszerzi). A játékos és a karakterek által birtokolt tárgyakat és a hozzájuk tartozó függvényeket leíró osztály.

StatItem: Item osztályt kiterjesztő osztály, amely olyan itemeket tartalmaz, amelyek az ubimonok statisztikáját képesek befolyásolni.

HoldItem: Item osztályt kiterjesztő osztály, amely olyan itemeket tartalmaz, amelyek a küldetésekhez kellenek.

UbiballItem: Item osztályt kiterjesztő osztály, amely ubimon labdákat ír le (különböző effektek lehetnek).

MailItem: Item osztályt kiterjesztő osztály, amely szöveget tartalmaz.

KeyItem: Item osztályt kiterjesztő osztály, amely ajtókat vagy ládákat képes kinyitni.

Character: A Gameobject leszármazottja, amely a karakter alap adatait és függvényeit tartalmazza.

PlayerCharacter: A játékos által irányított karakter. A character osztály leszármazottja. Komponensei: CharacterCollider, Physical, PlayerControls, Dialog, Sound, Camera

NPC: A nem játékos által irányított karakterekre vonatkozó adatokat és funkciókat tartalmazza. A Character osztály leszármazottja. Komponensei: CharacterCollider, Physical, Dialog, Sound, AI, Inventory.

Enemy: Az ellenséges NPC-k adatait és függvényeit leíró osztály. A character osztály leszármazottja. Komponensei: CharacterCollider, Physical, Dialog, Sound, AI, EnemyAI, Inventory.

Renderer: A GameObjectek renderelséért felelős osztály

Window: Az ablakért felelős osztály. A lwjgl függvényeket használja.

KeyListener: Olyan osztály, amely a játékos billentyűlenyomásait figyeli, lwjgl segítségével.

MouseListener: Olyan osztály, amely a játékos egér inputjait figyeli, lwjgl segítségével.

GamepadListener: Olyan osztály, amely a játékos kontroller inputjait figyeli, lwjgl segítségével.

DialogBox: A karakterek dialóg szövegének kiíratásáért felelős osztály. Képes játékostól inputot kezelni.

Texture: A textúra fájlok adatait és kezelésének a függvényeit tartalmazó osztály.

Spritesheet: A textúrát spritokká feldaraboló osztály.

Scene: Absztrakt osztály, amely rendelkezik rendererrel, kirajzolja a háttér textúrát és a scenehez tartozó objektumokat.

LevelScene: A scene osztály leszármazottja.

CombatScene: A scene osztály leszármazottja. A harcrendszer scene-je, amely tartalmazza a harchoz kapcsolódó játékos opciókat, megjeleníti az ubimonokat, azok statjait.

MenuScene: A scene osztály leszármazottja. A játék menüje.

InventoryScene:A scene osztály leszármazottja. A játékos inventoryjának az interface-e.

FightAction: A harcban történő játékos által kiadható támadásparancsok elvégzéséért felelős.

RunAction: A harcból való menekülésért felelős osztály.

InventoryAction: A harban az inventory használatáért felelős osztály.

UbimonAction: A harban történő ubimon cseréért felelős osztály.

SaveManager: A játékos által elmentett mentéseket és beállításokat betöltő, törlő és új mentéseket létrehozó osztály.

StatisticManager: A harcban, illetve item használata esetén az ubimonok statjait számítja ki, azok típusa, szintje és kondíciója alpján.

RewardManager: A harc utáni jutalomért felelős manager, amely a játékos és az ellenfél szintjét is figyelembe veszi.

Interactable: Interfész, amely az adott tile-el vagy karakterrel való interaktivitásért felelős.

ContactListener: A Physical és a tileok által meghatározott pozíció alapján az ütközésket figyelő objektum.

Networking: A hálózaton történő játékért felelős objektum

Game: A window-t és a Networking osztályokat használó (main) osztály.

2.3. Alrendszerek

Alrendszerek:

Statisztika alrendszer:

Az ubimonok statisztikáit: életpontokat (hp), sebzést(dmg), kondíciót, szintet menedzselő alrendszer. Az ubimonok statjai harc közben vagy itemek használatakor változhatnak. Az alrendszer részei: StatisticManager, FightAction, Statistic, Condition, UbimonType.

Jutalom alrendszer:

A harc utáni jutalmi itemekért felelős alrendszer. Az alrendszerben résztvevő osztályok: RewardManager, Reward, Statistic

Harc alrendszer:

A harc helyzetben a játékos által választható opciók megvalósításáért felelős alrendszer. Az alrendszerhez tartozó osztályok: FightAction, InventoryAction, RunAction, UbimonAction.

Megjelenítő alrendszer:

A játékban megjelenő elemekért felelős alrendszer. Az alrendszerhez tartozó objektumok: Sprite, SpriteRenderer, Spritesheet, Texture. Window, Scene, SceneManager, CombatScene, MenuScene, InventoryScene, LevelScene.

Ütközés alrendszer:

A karakterek pozíciója és a tile-ok pozíciója alapján ütközést figyelő alrendszer. Részei: ContactListener, CharacterCollider, TileCollider, Physical.

Network alrendszer:

Az online többjátékos módért felelős alrendszer. Része: Network.

3. A megjelenítő alrendszer modellje

3.1 Statikus modell

3.1.1 Kapcsolatok pontosítása

A megjelenítő alrendszer felel a játékban található összes grafikus elemének ábrázolására. A Renderer áll a grafikai megjelenítés feldolgozásának főpillérénél, amely közvetlen kapcsolatot ápol a GameObjecttel. A Renderer alatt lévő második ág a Window biztosítja a grafikai elemek egy nagy kereten belüli eltárolását, amely mellesleg közvetlenül elérhetővé teszi a DialogBox-ot, ami biztosítja a jelenetek egyes pontjain a rendszer válaszüzeneteit. Ez lehet a menüben lévő megerősítő kérdések, hibák visszajelzése, vagy a játék alatt lévő dialógusok megjelenítése.
A Window után elérkezünk a többértékű Scene osztályhoz is, ami az általa leszármaztatott kisebb játékjelenetekkel kapcsolatos osztályokból alkot egy csoportot, melyek a szoftver bizonyos szakaszain más jelenetek metódusaival dolgozik és állítja aktívra.
A Texture és az az alatt lévő többi alosztály fontos szerepet játszanak a grafikai elemek szabályszerű beazonosítására, kirajzolására és azok tárolására.

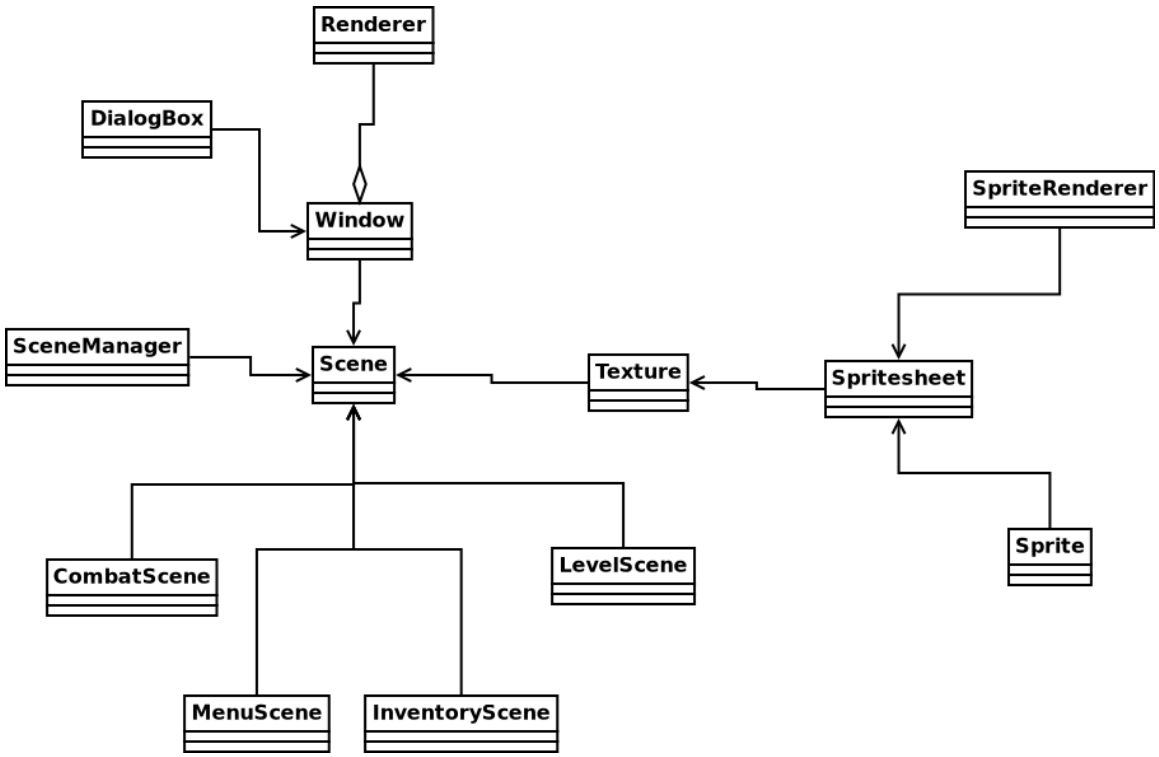
3.1.2 Attribútumok azonosítása

Az ebben az alrendszerben található osztályok többsége nem rendelkezik sok tulajdonsággal, mivel ezek feladata többnyire adott képsorozat (pálya) vagy objektum betöltése vagy rendelése. A képek célkoordinátáinak tárolására nincs szükség, mert azt mindig megkapják a kirajzolást kérő féltől. A Texture ősosztály leszármaztatottjai, azok amelyek a legtöbb aritmetikai számítást végzik el, többek között adott játék objektumok animációs fázisát a Spritesheet-ből.
A Scene osztály alatt található többi leszármazott osztály inkább csoportosítási és betöltési szerepet vál.

3.1.3 Bázisosztályok keresése

Az összes grafikai osztály jellemző tulajdonsága, hogy tárolják a képet és gondoskodnak ennek megjelenítéséről, ezenkívül léteznie kell a képek betöltését végző metódusnak is, például Sprite-ok esetén a SpriteRenderer osztályt, amelyhez adott egy draw() nevű metódus, ami x,y koordinátákba kirajzolja a képet. A Sprite osztályban található adattagok például a MapSprite esetében tiszta a helyzet, egy képet tárol, amit meg kell tudni jelenítenie, azonban ObjectSprite-nál más a helyzet, hiszen valahogy tárolnia kell egy objektum különböző nézeteit és animációs fázisait, amit a Spritesheet tesz lehetővé.

3.2. Dinamikus modell



3.3. Funkcionális modell

Egyképernyő (pálya és egységek) kirajzolásának menete a következő:

1. Az adott scene-től függően a Scene egyenként meghívja a kirajzolni kívánt mezők draw() metódusát

(pontosan közölve, hogy hova történjen a kirajzolás)

2. Miután a pálya kirajzolódott, a SceneManager üzenetet küld a látótérbe eső Objektumoknak, hogy rajzolják ki magukat

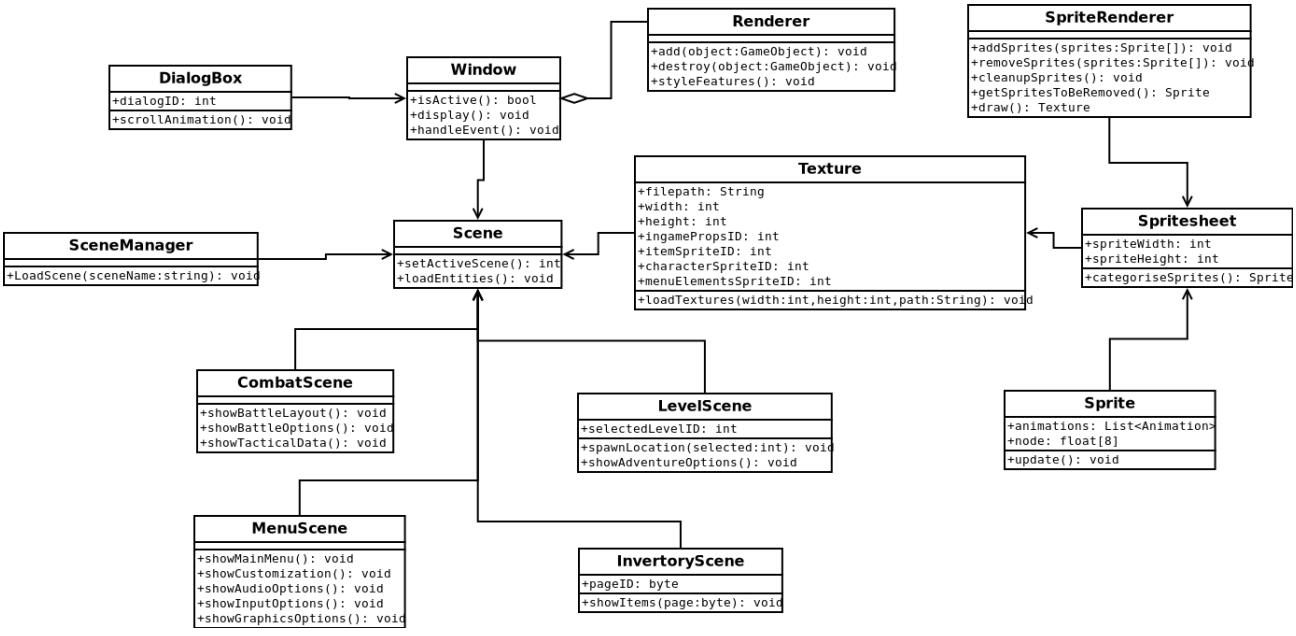
3. A SceneManager gondoskodik az animációról is, ezért bizonyos időközönként üzenetet küld a Scene-n belüli elemeknek, hogy frissítsék az animációs fázisukat.

4. Amikor a látótérbe eső egységek kirajzolására kerül a sor, a SceneManager üzenetet küld a SpriteRenderer-nek, hogy rajzolja ki a Scene-hez megfelelő Sprite-o(ka)t a Spritesheet-ből. Ekkor a draw() metódusa lefut a Renderer-ből, átadva neki a szükséges paramétereket.

3.4. Operációk azonosítása

A felhasználó a MenuScene láthatja, amikor elindítja a szoftvert egy a Window osztályon belül. A Scene-n belül lévő többi kisebb Scene osztály metódusai a játék bizonyos pontjain futnak le. A MenuScene választásait kattintható gombokkal tudja elérni, kontroller esetén a gombokkal, továbbá a menü irányítható a nyilakkal és az Enter, illetve Esc gombokkal. A játékmenet alatt szintén a nyilakkal irányítható a játszható karakter, a főkarakter további környezettel való interakciója pedig játékos által meghatározott gombbal történik főként klaviatúrán vagy kontrolleren.

3.5 Az analízis modell osztálydiagramja



3.6. Az analízis modell osztályainak listája

3.6.1 Renderer

Felelőssége, feladata: A renderer gondoskodik a játékban lévő összes objektum grafikai feldolgozásáról és azok megjelenítéséről.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
add(object:GameObject)	void	Adott játék objektumok hozzáadásáért felel.
destroy(object:GameObject)	void	Adott játék objektumok eltűntetéséért felel.
styleFeatures()	void	Csoportosított grafikai elemek stílus funkció lekérdezésére szolgál.

3.6.2. Window

Felelőssége, feladata: A játékszoftver által vetíteni kívánt értékek biztosítása egy ablakban.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
isActive()	boolean	Igaz/hamis értéket visszaadó metódus idle vagy aktív állapot lekérdezése érdekében
display()	void	Szabályszerű megjelenítésért felelős
handleEvent()	void	Bizonyos események szabályszerű lekezeléséért felelős metódus

3.6.3 DialogBox

Felelőssége, feladata: A játék bizonyos részein dialógus dobozokat jelenít meg, és ez az osztály gondoskodik az azonosításról, és a doboz animációjáról.

Attribútumok:

Név	Típus	Leírás

dialogID	int	Bizonyos dialógusokat beazonosító ID.

Operációk:

Név	Típus	Feladat
scrollAnimation()	void	Szövegek scroll animációjáért felelős metódus.

3.6.4. Scene

Felelőssége, feladata: A scene tárolja tulajdonképpen a játék minden egyes jelenetét, és a kisebb csoportokat a szoftver bizonyos pontján váltogatja és tölti be az azokhoz megfelelő egyedeket.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
setActiveScene()	int	Ez a metódus felel adott kritériumoknak megfelelő jelenetek változtatására.
loadEntities()	void	A megfelelő jelenetekhez tartozó objektumok betöltésére szolgál.

3.6.5. SceneManager

Felelőssége, feladata: A jelenetmenedzser segítségével tudunk a programban utalni a jelenetváltozásokra. Itt tudjuk kezdeményezni egy másik jelenet betöltését.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
LoadScene(sceneName:string)	void	Ezt a függvényt hívjuk meg akkor, ha az adott feltételeknek megfelelően váltani szeretnénk egy másik jelenetre.

3.6.6. CombatScene

Felelőssége, feladata: Az Ubimon játékban nagy hangsúlyt fektettünk az ellenfelek elleni megméretettés alatt. Mint minden másik RPG-ben itt is fontos, hogy legyen egy külön harci jelenet, ahol a játékos dönthet arról, hogyan arasson győzelmet. Ez az osztály biztosítja e interfész megjelenítését.

Attribútumok:

Név	Típus	Leírás
ubimonID	int	A játékos által bírtokolt ubimonID-je.
enemyUbimonID	Int	Az ellenfél által birtokolt ubimon ID-je.

Operációk:

Név	Típus	Feladat
showBattleLayout()	void	A harc jelenet alaprajzának megjelenítéséről gondoskodik.
showBattleOptions()	void	Biztosítja a felhasználó számára az elérhető lehetőségeket.
showTacticalData()	void	Taktikai/stratégiai információk elővetítésére szolgál. Sok kritérium változhat.
showUbimonStatistics(int:pu_id, int:eu_id)	void	A játékos és az ellenfél által használt ubimonok statjait megjelenítő függvény.

3.6.7. MenuScene

Felelőssége, feladata: Ez a jelenet tartalmazza azon elemek megjelenítését, melyek a játék mechanikáját megvalósítják.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
showMainMenu()	void	A játék főmenüjének a megjelenítéséről gondoskodik.
showCustomization()	void	A karakterkinézet választó jelenet megjelenítése új játék kezdése érdekében.
showAudioOptions()	void	Hang/zene beállítások megjelenítésére szolgál.
showInputOptions()	void	A bemeneti gombok beállítások megjelenítésére szolgál.
showGraphicsOptions()	void	Grafikai beállítások megjelenítésére szolgál.

3.6.8. LevelScene

Felelőssége, feladata: Mivel a játékunk RPG műfajt alkalmaz, így a harc jelenet mellett, fontos biztosítani egy nyíltvilágot is, melyen a játékos saját karakterét irányítva szabadon dönthet az útvonaláról. Ez az osztály gondoskodik adott pálya résznek megfelelő betöltéséről és kezeléséről.

Attribútumok:

Név	Típus	Leírás

selectedLevelID	int	Játék betöltése esetén szükség van adott pályarészek megfelelő azonosítására. Ezeket az azonosítókat itt tároljuk.

Operációk:

Név	Típus	Feladat
spawnLocation(selected:int)	void	Ez a metódus felel a selectedLevelID tulajdonságnak megfelelő pályarészhez való spawnolásához.
showAdventureOptions()	void	Bizonyos gomb aktiválása esetén előjön az ún. “kaland” beállítások, melyben elérhető a térkép, az invertory, a mentési opció, a statisztika, az Ubimonok stb. Ennek a menünek a megjelenítésére szolgál ez a metódus.

3.6.9. InventoryScene

Felelőssége, feladata: A hátizsákunkban lévő tárgyak megjelenétésért felel.

Attribútumok:

Név	Típus	Leírás
pageID	byte	Mivel adott tárgyak kategórizálva vannak, ezért ezeknek az oldalait külön azonosítanunk kell.

Operációk:

Név	Típus	Feladat
showItems(page:byte)	void	Bemeneti paraméternek megfelelő oldalon lévő tárgyak megjelenítése

3.6.10. Texture

Felelőssége, feladata: Itt azonosítjuk be a játék alatt előforduló összes textúrának kategóriáját, fájl elérési helyét. magasságát, szélességét, illetve azokat töltjük be a renderelés alatt.

Attribútumok:

Név	Típus	Leírás
filepath	String	A fájl elhelyezkedésének pontos címét tárolja.
width	int	Textúra szélességének értéke.
height	int	Textúra magasságának értéke.

ingamePropsID	int	A játék nyílt világ terén előforduló tereptárgyak azonosítására szolgáló érték.
itemSpriteID	int	Adott tárgyak beazonosítására szolgáló érték.
characterSpriteID	int	A karakterek spritejainak beazonosítására szolgáló érték.
menuElementsSpriteID	int	Adott menü grafikai elemek beazonosítására szolgáló érték.

Operációk:

Név	Típus	Feladat
loadTextures(width:int,height:int, path:String)	void	A jelenetnek megfelelő adott egyedekhez használandó textúrák betöltése.

3.6.11. Spritesheet

Felelőssége, feladata: A spritesheet tartalmazza egy adott csoportban található elem több kisebb képekből álló állapotát, melyek a szoftver bizonyos részein töltődnek be, ezzel érzékeltetve az egyenletes animációt.

Attribútumok:

Név	Típus	Leírás
spriteWidth	int	A sprite szélességének értéke
spriteHeight	int	A sprite magasságának értéke.

Operációk:

Név	Típus	Feladat
categoriseSprites()	Sprite	Bizonyos sprite elemek megfelelő csoportba való kategórizálásáért szolgál.

3.6.12. SpriteRenderer

Felelőssége, feladata: A SpriteRenderer felel adott Sprite-ok megjelenítésére, és irányítja hogyan jelenjenek meg vizuálisan a jelenetek alatt.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
addSprites(sprites:Sprite)	void	Sprite(ok) hozzáadása adott jelenethez a kritériumoknak

		megfelelően.
removeSprites(sprites:Sprite)	void	Sprite(ok) eltávolítása adott jelenetek alatt.
cleanupSprites()	void	Más jelenet beolvasása alatt érdemes váltás közben a többi nem aktív sprite-okat kitörölni optimalizáció érdekében.
getSpritesToBeRemoved()	Sprite	Egy olyan getter metódus, mely segít beazonosítani azon elemeket, melyekre bizonyos jelenetnél nincs szükségünk.
draw()	Texture	Ez a metódus szolgál a textúrák kirajzolására.

3.6.12. Sprite

Felelőssége, feladata: A Sprite-ok kétdimenziós grafikai objektumok statikus textúrával, melyek bizonyos változások után váltakoznak általában lassabb képkocka/másodperces változással. Kevesebb számítást vesznek igénybe, mint a bonyolultabb 3D-s modellek. Ezek csoportosításával, azonosításával és frissítésével foglalkozunk ebben az osztályban.

Attribútumok:

Név	Típus	Leírás
animations	List<Animation>	Animation típus alapú lista főként karakter sprite-ok esetén, hogy könyedén tudjuk kiolvasni adott pontokon lévő sprite cellákat.
node	float[8]	Ez a 8 elemű float tömb tárolja a sprite x y pontjait sorban.

Operációk:

Név	Típus	Feladat
update()	void	Egy folyamatos frissítésre szolgáló algoritmus, hogy minél gyorsabban tudjon a program sprite váltásokra reagálni.

4. A statisztikai alrendszer modellje

4.1 Statikus modell

4.1.1 Kapcsolatok pontosítása

A statisztikai adatok lekérdezése egy távoli szerverről és egy lokális adatbázisból történik. Az programban egy osztály felelős azért, a fontosabb statisztikai elemeket lekérdezze a játék adott pontjain, és protokoll szerint menedzselje az értékváltozásokat. Az adatbázisból lehet lekérdezni az Ubimonok tulajdonságait, kondícióit, egyes játékbeli egyed statisztikáit.

4.1.2 Attribútumok azonosítása

A statistic osztály legfontosabb attribútumai:

health: Az adott ubimon életpontja

exp: Az adott ubimon tapasztalatpontja

friendship: barátsági szint

level: Az adott ubimon szintje

Condition: A kondíciók listája

type: Az adott ubimon típusa

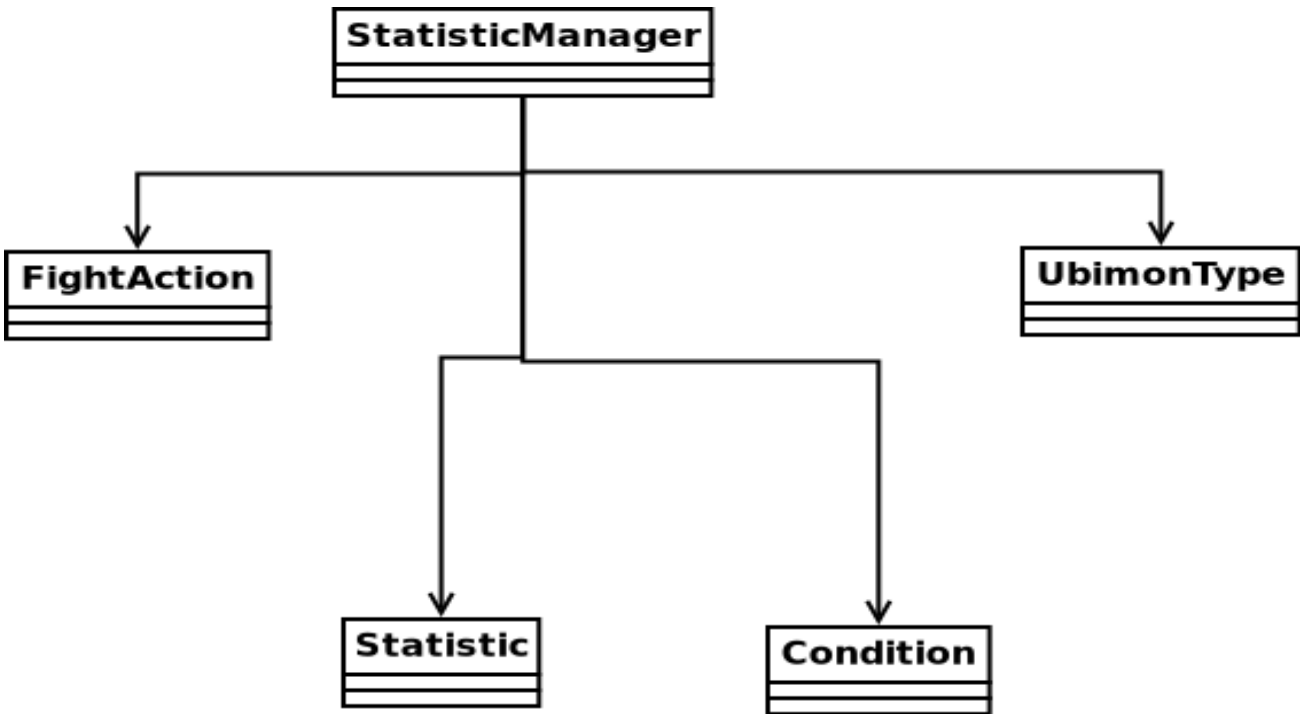
weakTypes: Azok az ubimontípusok, amelyre az adott ubimon gyenge.

strongTypes: Azok az ubimontípusok, amelyre az adott ubimon erős.

4.1.3 Bázisosztályok keresése

Nincs kifejezetten olyan osztály, amely bázisosztálynak kiemelhető lenne, hiszen az adatbázisba való írás, olvasás és lekérdezés műveleteket egy részét ez az alrendszer hajtja végre.

4.2. Dinamikus modell



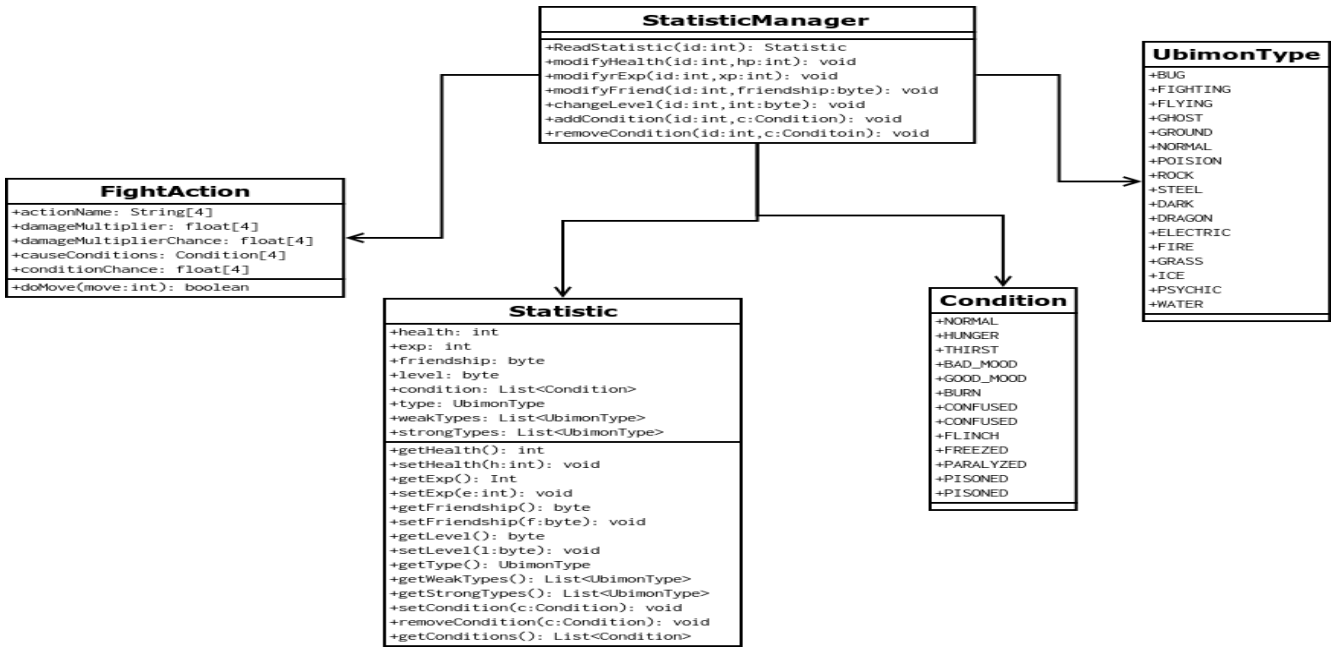
4.3. Funkcionális modell

A StatisticManager osztály ír és olvas az adatbázisból, módosítja az adatbázist a FightAction osztály üzeneteinek megfelelően, továbbá a GameObjectekhez tartozó Statistic osztály értékeit módosítja.

4.4. Operációk azonosítása

A játék alatt a statisztikai módosítások inkább a harcok alatt játszanak hatalmas szerepet (főleg ha a Tamagotchi mód ki van kapcsolva), hiszen a harcok alatt fejlődik a szörnyünk, kifáradhat, újabb mozdulatokat tanulhat meg ahogy nagyobb szinten lesz, státusz módosítokat rakhatnak rá stb.
Ugyanakkor egyes kondíció értékek főként a nyílt világi tevékenységünk alapján módosul, hogy mennyire tartjuk az Ubimonunk éhség/szomjúság mértékén magasan, illetve mennyire gondoskudunk a kedvéről.

4.5 Az analízis modell osztálydiagramja



4.6. Az analízis modell osztályainak listája

4.6.1 StasticManager

Felelőssége, feladata: A statisztikai menedzser segítségével tudunk a programban utalni a statisztikai változásokra.

Attribútumok: nincs

Operációk:

Név	Típus	Feladat
ReadStatistic(id:int)	Statistic	Ezt a függvényt hívjuk meg akkor, ha az adott feltételeknek megfelelően szeretnénk leolvasni adott Ubimon statisztikáit.
modifyHealth(id:int, hp:int)	void	A HP érték módosítása.

modifyrExp(id:int, xp:int)	void	Tapasztalatipont növelése/csökkentése.
modifyFriend(id:int, friendship:byte)	void	Barátsági pont növelése/csökkentése.
changeLevel(id:int, level:byte)	void	A szint változtatása.
addCondition(id:int, Condition:c)	void	Kondíció hozzáadása
removeCondition(id:int, Conditoin:c)	void	Kondíció eltávolítása

4.6.2. FightAction

Felelőssége, feladata: A harcban lévő Ubimonok statisztikai adatait, és cselekdeteinek metódusait tartalmazó osztály.

Attribútumok:

Név	Típus	Leírás
actionName	String[4]	A támadások nevei
damageMultiplier	float[4]	A támadások sebzés szorzói
damageMultiplierChance	float[4]	A támadás sebzés szorzóinak valószíműsége
causeConditions	Condition[4]	Az okozható kondíciók
conditionChance	float[4]	Az okozható kondíciók valószínűsége

Operációk:

Név	Típus	Feladat
doMove(move:int)	boolean	A felhasználó által kiválasztott mozdulatot végzi el az Ubimon sok feltétel teljesülése után.

4.6.3 Statistic

Felelőssége, feladata: Adott Ubimon statisztikáit tárolja, és metódusokkal módosítja.

Attribútumok:

Név	Típus	Leírás
health	int	Az Ubimon életerje.

exp	int	Az Ubimon tapasztalatpontjai.
friendship	byte	Az Ubimon és a mestere közötti barátsági pont.
level	byte	Az Ubimon szintje.
condition	List<Condition>	Az Ubimon kondíciói.
type	UbimonType	Az Ubimon típusa
weakTypes	List<UbimonType>	Azok az ubimon típusok, amelyek extra sebzést okoznak az Ubimonra
strongTypes	List<UbimonType>	Azok az ubimon típusok, amelyekre extra sebzést képes okozni az Ubimon.

Operációk:

Név	**Típus**	**Leírás**
getHealth()	int	Életpont lekérdezése
setHealth(h:int)	void	Életpont beállítása
getExp()	Int	Tapasztalatpont lekérdezése
setExp(e:int)	void	Tapasztalatpont beállítása
getFriendship()	byte	Barátsági pont lekérdezése
setFriendship(f:byte)	void	Barátsági pont beállítása
getLevel()	byte	Szint lekérdezése.
setLevel(l:byte)	void	Szint beállítása
getType()	UbimonType	Típus lekérdezése
getWeakTypes()	List<UbimonType>	Azok a típusok, amelyekre gyenge
getStrongTypes()	List<UbimonType>	Azok a típusok amelyekre erős
setCondition(c:Condition)	void	Kondíció beállítása
removeCondition(c:Condition)	void	Kondíció törlése
getConditions()	List<Condition>	Kondíciók lekérdezése

4.6.4 Condition

Feladata: Adott Ubimon kondiciójának értékeit tárolja enum osztály.

Attribútumok:

Név	Típus	Leírás
NORMAL	Condition	Egészséges státusz esetén ez az érték 0.
HUNGER		Az Ubimon éhsége (Tamagotchi mód kikapcsolása esetén ez az érték lekérdezése nem aktív).
THIRST		Az Ubimon szomjúsága (Tamagotchi mód kikapcsolása esetén ez az érték lekérdezése nem aktív).
BAD_MOOD		Az Ubimon kedve (Tamagotchi mód kikapcsolása esetén ez az érték lekérdezése nem aktív).
GOOD_MOOD		
BURN		Minden körben elvesz az ubimon életéből, lecsökkenti a sebzést
CONFUSED		1 körben 50% esély, hogy az ubimon magát sebzi.
FAINTED		Akkor érvényes, ha az ubimon élete 0
FLINCH		Nem támadhat 1 körig
FREEZED		Nem támadhat, ameddig ki nem lesz olvasztva
PARALYZED		A támadások 50% 0 sebzést okoz
PISONED		Minden körben csökken az ubimon élete, a harc után is megmarad
SLEEP		Pár körben nem támadhat.

Operációk: Nincs.

4.6.4 UbimonType

Felelőssége, feladata: Ubimon azonosításra szolgáló osztály.

Attribútumok:

Név	Típus	Leírás
BUG	UbimonType	Ubimon típusok
FIGHTING		
FLYING		
GHOST		
GROUND		
NORMAL		

POISION		
ROCK		
STEEL		
DARK		
DRAGON		
ELECTRIC		
FIRE		
GRASS		
ICE		
PSYCHIC		
WATER		

Operációk: Nincs

5. A harc alrendszere

5.1 Statikus modell

5.1.1 Kapcsolatok pontosítása

A harc alrendszer feladata, hogy meghatározza, hogy a játékosnak harchelyzetben milyen lépései lehetnek. A Scene osztályt leszármaztató CombatScene osztály fogja megjeleníteni a harcban résztvevő ubimonokat és a játékos által választható lehetőségeket, amelyek a FightAction, RunAction, InventoryAction és UbimonAction.

A FightAction a statisticManager osztállyal áll kapcsolatban, amely tartalmazza az adott ubimonhoz tartozó értékeket, kondíciót továbbá ezek típusát, amely meghatározza, hogy a harcban résztvevő pokémonok sebzési és védelmi értékei milyen szorzóval lesznek kiszámítva (2x, 1x, 0.5x, 0x). Az InventoryAction és az UbimonAciton kapcsolatban áll az Inventory osztállyal. Az inventoryAction kilistázza a játékos által bírtokolt olyan itemeket, amelyek a harcban használt ubimon statisztikáit módosíthatják (StatItem). Az UbimonAction kapcsolatban áll az Inventory osztállyal, mert az ubimonok Ubimon labdákhoz (UbiballItem) van kötve, az osztály lényege, hogy kilistázza a játékos által ubimon labdákban tárolt ubimonokat és harc közben, hogy a játékos tudjon ubimont váltani.

5.1.2 Attribútumok azonosítása

Az alrendszer legfontosabb attribútumai a FightAction osztályhoz tartoznak.

- actionName: A támadások neveit eltároló tömb.

- damageMultiplier: A sebzés szorzó értékeket eltároló tömb.

- damageMultiplierChance: A sebzés szorzó értékek valószínűségét tároló tömb.

- causeConditions: A támadás során létrehozható kondíciók listája.

- conditionChance: A támadás során kondíció létrehozásának a valószínűségét tároló tömb.

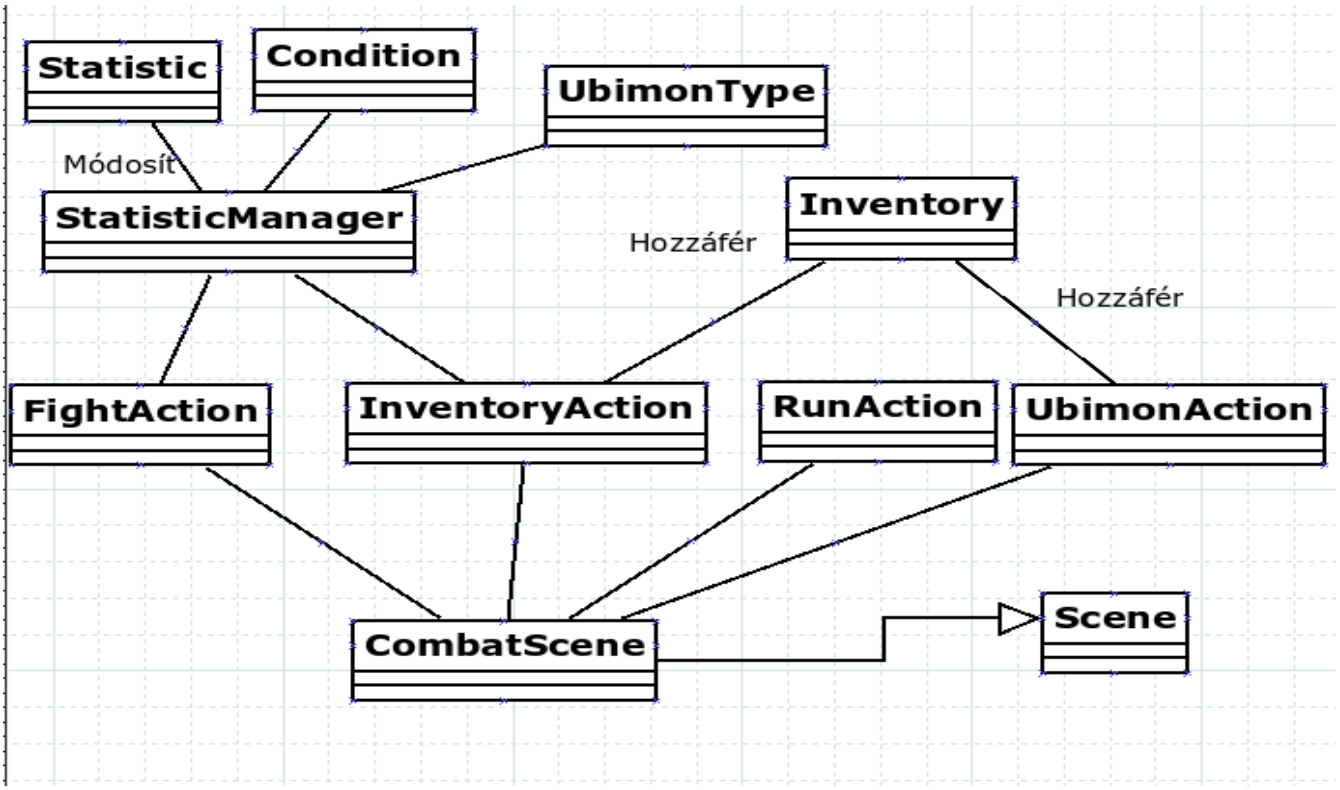
Az UbimonAction osztály attribútumja:

- currentID: A jelenlegi ubimon iD-je.

5.1.3 Bázisosztályok keresése

A harc rendszer osztályai nem rendelkeznek bázisosztályokkal (önálló osztályok).

5.2. Dinamikus modell



5.3 Funkcionális modell

Választás: A játékos választ a 4 lehetőség közül. (Fight, Run, Inventory, Ubimon)

Végrehajtás: A CombatScene osztály Inventory vagy Ubimon választás esetén megjeleníti a megfelelő menüt. Támadás kiválasztása esetén a megfelelő animáció kerül megjelenítésre. (Vizuális visszajelzések).

Jelzés a StatisticManagernek: A játékos támadás / sebződés vagy statisztikát befolyásoló tárgy használata esetén

jelez a statisticManagernek, hogy változtassa meg az adott Ubimonhoz tartozó értékeket.

Ubimon váltás: Az ubimon spritjának és a sceneben megjelenített értékek kicserélése.

5.4 Operációk azonosítása

FightAction:

- doMove(move:int) A kiválasztott támadás végrehajtása
- activateHeld(item:byte) Kézben tartott item használata.

RunAction:

- run() Harc elhagyása.

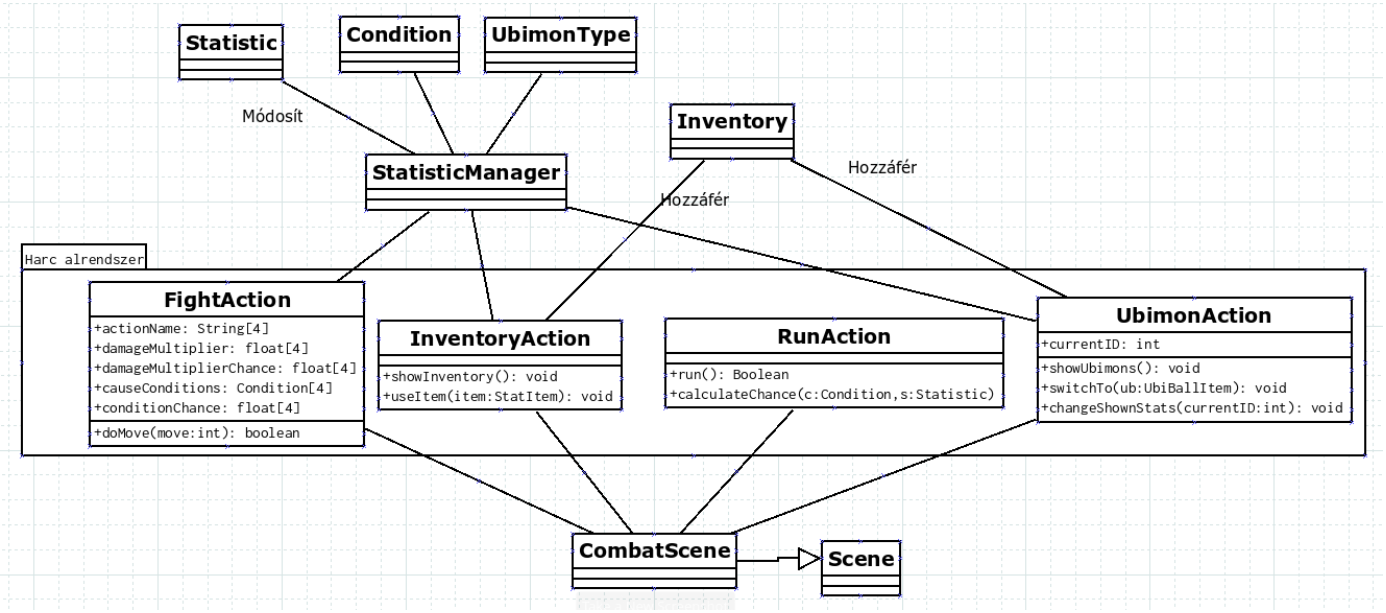
InventoryAction:

- showInventory() Inventory menü megjelenítése
- useItem(StatItem:item) Kiválasztott item aktiválása

UbimonAction:

- showUbimons() Ubimon menü megjelenítése
- switchTo(UbimonBall:ubimonball) Kiválasztott ubimonra váltás.

5.5 Az analízis modell osztálydiagramja



5.6.1. FightAction

Felelőssége, feladata: A harcban lévő Ubimonok statisztikai adatait, és cselekdeteinek metódusait tartalmazó osztály.

Attribútumok:

Név	Típus	Leírás
actionName	String[4]	A támadások nevei
damageMultiplier	float[4]	A támadások sebzés szorzói
damageMultiplierChance	float[4]	A támadás sebzés szorzóinak valószíműsége
causeConditions	Condition[4]	Az okozható kondíciók
conditionChance	float[4]	Az okozható kondíciók valószínűsége

Operációk:

Név	Típus	Feladat
doMove(move:int)	boolean	A felhasználó által kiválasztott mozdulatot végzi el az Ubimon sok feltétel teljesülése után.

5.6.2 InventoryAciton

Feladatai: A játékos által bírtokolt tárgyak kilistázása egy menüben. A játékos által kiválasztott tárgy aktiválása.

Attribútumok: Nincs

Opeációk:

Név	Típus	Feladat
ShowInventory()	void	Az inventory menü megjelenítése.
useItem(StatItem:item)	void	Ubimon statisztikát módosító tárgy használata.

5.6.3 RunAction

Feladatai: A harc elhagyása és a harc elhagyásának esélyét kiszámolni

Attibútumok: Nincs

Operációk:

Név	Típus	Feladat
run()	Boolean	A harcból való eltávozás a feltételeknek megfelelően.
CalculateChance(Condition: c, Statistic: s)	float	A menekülés esélyét kiszámító függvény.

5.6.4 UbimonAction

Feladata: Harc közben Ubimon váltás, megfelelő statisztika megjelenítése.

Attibútumok:

Név	Típus	Feladat
currentID	int	A jelenlegi ubimonID-je.

Operációk:

Név	Típus	Feladat
showUbimons()	void	Az ubimonok listája.
switchTo(UbiBallItem:u)	void	Ubimon váltás. (id kicserélés)
changeShownStats(int:currentID)	void	megfelelő statisztikák megjelenítése