Matematikai és Informatikai Intézet

Játékfejlesztés Unity keretrendszerben

Készítette
Rokob Attila Adrián
Programtervező Informatikus

Témavezető
Troll Ede
tanársegéd

Eger, 2025

Tartalomjegyzék

Bevezetés

Még annak idején, éppen hogy óvodából kikerülve volt az első alkalom, hogy játszottam bármi féle számítógépes játékkal. Még olvasni alig tudtam, első betű amit megjegyeztem, az a "C" volt, hogy el tudjak jutni a Cartoon Network, saját, Flash játék oldalára, tele volt végtelen sok, alacsony költségvetésű játékkal, minden féle Cartoon Network IP alapján. Egy Mario Maker szerű Ben 10 Platformer, egy astroboy shoot em up, Egy Johnny test top-down shooter. Közel már ekkor eldöntöttem, hogy én, ha nagy leszek, játékokat fogok fejleszteni. Amint el végeztem az általános iskolát, emiatt is mentem egyenesen tovább egy informatikus középiskolába. Ebben a középiskolában íram meg az első tényleges sor kódomat, egy c# tömb kiírót, GOTO kifejezéssel, mivel még nem tudtam róla, hogy hogyan működnek a ciklusok. Ezen felüll ebben az iskolában jöttem rá, mennyire élvezhető is maga a kód írás, mennyi kreativitást lehet felhasználni, egy ennyire strukturált médiumban is. Miután végeztem közép iskolával, illetve egy covid alatti szoftverfejlesztő képzéssel, természetesen egyből jött is az egyetem, akkor már inkább egy stabil, viszonylag stresszmentes karrier által motiválva. Itt egyetemen készült el az első, befejezett (illetve annak mondott) Unity játékom, egy, a konzulensem Troll Ede tanárúr által rendezett Game Jam-re készített kalózos, társasjáték. Ennek a projektnek a készítése során Tanultam meg a szakdolgozatom alapját is adó Unity játékmotor alapjait. Ezek után már eléggé egyértelművé vált számomra, hogy a szakdolgozatomat is szeretném, hogy egy játék legyen, és mivel Unity-vel már volt tapasztalatom, illetve nem mellékesen továbbra is vezeti a piacot kissebb költségvetésű játékoknál. Magának a projetktnek, a tematikája eredetileg még a Leányka Bisztró egyik kanapéján dőlt el, hogy legyen fél úton egy "kalózos For The King", és egy "cyberpunk Nuclear Throne" között. Mivel eredetileg Gyökösi kata feladata volt az univerzum, illetve az egész játék kinézetének létrehozása, ezért maradtunk a kalóz tematikánál. Mivel a játékmenet létrehozása az én feladatom volt, emiatt maradtunk Nuclear Throne szerű felül nézetes, lövöldözős játékmenetnél, illetve idő hiánya miatt a Rogue Like elemek hozzáadásánál, mint például a véletlen generált világ, ellenfelek, és szint végén kapható tárgyak.

1. fejezet

Technológiai Bevezetés

1.1. Godot bemutatása

1.1.1. Főbb jellemzők

A godot Engine egy Multi platform, ingyenes, nyílt forráskódú játékmotor. Életét egy zárt forráskódú, "Larvotor" nevű motorként kezdte, argentin játékfejlesztő cégek számára, 2001-ben[2]. Első nyilvános verzióját 2014-ben adták ki GitHub-ra, egy MIT licensz[4] alatt. 2016-ban 20000$ támogatást nyert el a Mozilla Open Source Support "Mission Partners" Programjának keretében, Web Sockets, WebAssembly, és WebGL 2.0 támogatás hozzáadása céljából.[6]

Nevét a Samuel Beckett, francia "Godotra várva"[1] könyvről kapta. A könyvben kettő karakter a címzetes Godotra várnak, aki viszont nem érkezik meg. Ezt a játékmotor eredeti készítői, Juan Linietsky és Ariel Manzur hasonlítják ahhoz, ahogyan szoftver fejlesztők keresik a tökéletes megoldást, a tökéletes kódot, ami ahogy a könyvben, úgy a szoftverfejlesztésnél sem érkezik sosem.[2] A godot 3.0 egy közel teljes refaktorálást igényelt, hogy lehetővé tudja tenni a rendering-pipeline újradolgozását.[5] Nem sokkal ez után a verzió után létre hoztak egy Patreon oldalt, amely lehetővé tette hogy a kettő eredeti fejlesztő teljes munkaidőben tudjon dolgozni a motor fejlesztésén. 2019-ben a fejlesztői csoport két részre bomlott. Míg Linietsky csapata a jövő béli 4.0 ággal foglalkozott, Verschelde csapata a 3.0 branch fenntartását kezelte. A 4.0 verzió ismételten újra írta a motor magjának jelentős részét, hogy frissebb hardver lehetőségeket ki tudjon használni, mint például a több szálon futó kódot.[8]

A motor 4.0 verziója 2023-ban lett kiadva. Ebben a verzióban adtak hozzá támogatást a Vulkan rendering API-hoz, illetve sokkal optimalizáltabbá tették a GDScript-et, és a beépített renderer-t is.[7] Íráskori legfrissebb verziója a Godot 4.4, amely sok régóta várt funkciót hozzáadott a motorhoz, például fizikai interpolációt, egy fejlettebb beépített fizikai motort, és játékon belüli szerkesztést.[9]

A Godot-ban minden játék egy Node[10] alapú fa hierarchia, ahol minden vissza

vezet az úgy nevezett root node-ra[10]. Ez a node tartalmazza magában beágyazva a játék összes részét. Több node gyűjteménye hoz létre egy Scene-t[10], ami egyfajta tároló al-Node-oknak, újrafelhasználhatóság érdekében. Például egy Player Scene valószínőleg tartalmazni fog magában egy Sprite-ot, egy Collidert, és egy Script-et, ami mozgatja a karaktert. Ezen felül minden node közti kommunikáció Signal-ok[10] formájában történik. Ezek egy szintnyi absztrakciót adnak hard kódolt eventek felett. Például, hogyha meg nyomunk egy gombot, ami megnyit egy menüt, se a menünek nem kell tudnia, hogy melyik Node nyitotta meg, se a gombnak, hogy mit csinál az a signal, amit meg hív. Ezen felül számos beépített Signal is van, például, hogy kettő collider ütközött, vagy egy node-ot megsemmisítettek. A signal rednszer a Godot vezriója egy Observer Pattern-nek.[12] A Godot-ban minden kód* egy saját fejlesztésű nyelven, a GDScript-ben[11] íródik. Ez a nyelv a godot-hoz hasonlóan objektum orientált, illetve imperatív, tehát azt írja le hogy *hogyan* érjük el a célunkat, ahelyett, hogy *micsoda* a cél. A nyelv fokozatosan típusos[3], ami ezt jelenti, hogy valahol a gyengén, és erősen típusos nyelvek között helyezkedik el. Minden értéknek kell, hogy legyen egy típusa, viszont van egy beépített "dynamic" típus is, ami lehtővé teszi hogy az érték típusa csak futási időben legyen meghatározva. A Godot motor főbb előnyei Unityhez képset, sokkal magasabb szintű modularitás, a Node rendszernek köszönhetően, illetve az MIT licensznek köszönhetően a nyílt szabad felhasználás. Főbb hátrányai Unity-hez képest, egy sokkal kissebb, kevésbé tapasztalt közösség, illetve még mái napig fejlesztés alatt álló 3D képességek.

1.1.2. Unity keretrendszerhez képest

1.2. Unreal bemutatása

1.2.1. Főbb jellemzők

1.2.2. Unity keretrendszerhez képest

1.3. Unity bemutatása

1.3.1. Főbb jellemzők

1.3.2. Miért a Unity-t választottam

1.3.3. Használt Unity komponensek bemutatása

2. fejezet

Rendszerterv

3. fejezet

Saját projekt fejlesztése

4. fejezet

Tesztelés

5. fejezet

Összegzés

5.1. Az eredeti tervből mi valósult meg

5.2. Jövő béli fejlesztési lehetőségek

5.3. Mit csinálnék máshogy?

6. fejezet

Ábrák jegyzéke

Irodalomjegyzék

[1] Godotra Várva könyv: *Kathleen Kuiper* https://www.britannica.com/topic/Waiting-for-Godot, 2011

[2] Első publikus képek a Godotról: *Juan Linietsky*, https://godotengine.org/article/godot-history-images

[3] What is Gradual Typing: *Jeremy Siek*, https://jsiek.github.io/home/WhatIsGradualTyping.

[4] MIT licensz: https://opensource.org/license/mit

[5] Godot 3.0 release notes: *Juan Linietsky*: https://godotengine.org/article/godot-3-0-released/

[6] Mozilla Awards $385,000 to Open Source Projects as part of MOSS "Mission Partners" Program: *Mozilla*, https://blog.mozilla.org/en/mozilla/mozilla-awards-385000-to-open-source-projects-as-part-of-moss-mission-partners-program/

[7] Godot 4.0 sets sail: All aboard for new horizons: *2000+ Godot contributors*, https://godotengine.org/article/godot-4-0-sets-sail/, 2023

[8] 2022: A Retrospective: *Juan Linietsky*, https://godotengine.org/article/2022-retrospective/, 2022

[9] A unified experience: *Godot Contributors*, https://godotengine.org/releases/4.4/, 2025

[10] Overview of Godot's key concepts: *Godot Contributors*, https://docs.godotengine.org/en/stable/getting_started/introduction/key_concepts_overview.h

[11] : GDScript: *Godot Contributors*, https://docs.godotengine.org/en/stable/tutorials/scripting/g

[12] Observer: https://refactoring.guru/design-patterns/observer

[13] Tómács Tibor: *A valószínűségszámítás alapjai*, Líceum Kiadó, Eger, 2005.

7. fejezet

Fejezet címe

7.1. Szakasz címe

7.1.1. Alszakasz címe

Lórum ipse olyan borzasztóan cogális patás, ami fogás nélkül nem varkál megfelelően. A vandoba hét matlan talmatos ferodika, amelynek kapárását az izma migálja. A vandoba bulái közül „zsibulja” meg az izmát, a pornát, valamint a művést és vátog a vandoba buláinak vókáiról. Vókája a raktil prozása két emen között. Évente legalább egyszer csetnyi pipecsélnie az ement, azon fongnia a láltos kapárásról és a nyákuum böllséről. [1, 102. oldal]

A vandoba ninti és az emen elé redőzi a szamlan radalmakan érvést. Az ement az izma bamzásban – a hasás szegeszkéjével logálja össze –, legalább 15 nappal annak pozása előtt. Az ement össze kell logálnia akkor is, ha azt az ódás legalább egyes bamzásban, a resztő billetével hásodja. [1, 2]

7.1. Tétel. *Tétel szövege.*

Bizonyítás. Bizonyítás szövege. □

7.2. Definíció. Definíció szövege.

7.3. *Megjegyzés.* Megjegyzés szövege.

Összegzés

Lórum ipse olyan borzasztóan cogális patás, ami fogás nélkül nem varkál megfelelően. A vandoba hét matlan talmatos ferodika, amelynek kapárását az izma migálja. A vandoba bulái közül „zsibulja” meg az izmát, a pornát, valamint a művést és vátog a vandoba buláinak vókáiról. Vókája a raktil prozása két emen között. Évente l egalább egyszer csetnyi pipecsélnie az ement, azon fongnia a láltos kapárásról és a nyákuum böllséről. A vandoba ninti és az emen elé redőzi a szamlan radalmakan érvést. Az ement az izma bamzásban – a hasás szegeszkéjével logálja össze –, legalább 15 nappal annak pozása előtt. Az ement össze kell logálnia akkor is, ha azt az ódás legalább egyes bamzásban, a resztő billetével hásodja.

Irodalomjegyzék

[1] Fazekas István: *Valószínűségszámítás*, Debreceni Egyetem, Debrecen, 2004.

[2] Tómács Tibor: *A valószínűségszámítás alapjai*, Líceum Kiadó, Eger, 2005.

Nyilatkozat

Alulírott ....., büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, ... című szakdolgozat önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Aláírásommal igazolom, hogy az elektronikusan feltöltött és a papíralapú szakdolgozatom formai és tartalmi szempontból mindenben megegyezik.

Eger, 2024. december 18.

aláírás

A szakdolgozat megírása után ezt a nyilatkozatot kell a végéhez csatolni a következő módon:

1. A `nyilatkozat` mappában a `nyilatkozat.tex` fájlt töltse ki és fordítsa le pdf-be!
2. A `nyilatkozat.pdf` fájlt nyomtassa ki, majd írja alá!
3. Szkennelje be pdf formátumba, majd ezzel írja felül a `nyilatkozat` mappában a `nyilatkozat.pdf` fájlt!
4. A szakdolgozat forrásfájljában legyen betöltve a `pdfpages` csomag. Az utolsó két sor legyen az alábbi:

   ```
   \includepdf{nyilatkozat/nyilatkozat.pdf}
   \end{document}
   ```

5. Ezután fordítsa le a szakdolgozatot pdf-be!