Szakdolgozat

Miskolci Egyetem

Karaktermozgatás állapotgépekkel a Godot játékmotorban

Készítette:
Halász Máté Sándor
Programtervező informatikus

Témavezető:
Dr. Hornyák Olivér

Miskolc, 2025

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Halász Máté Sándor (T1TNWL) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: állapotgép, videójáték

A szakdolgozat címe: Karaktermozgatás állapotgépekkel a Godot játékmotorban

A feladat részletezése:

A szakdolgozat célja egy játékoskarakter mozgásának és interakcióinak megvalósítása állapotgépek alkalmazásával. A munka középpontjában egy olyan szoftverarchitektúra kialakítása áll, amely a játékos aktuális állapotát állapotgéppel reprezentálja, és ezen keresztül biztosítja a különböző mozdulatok, akciók és interakciók szabályozását.
A megvalósítás során minden mozdulat meghatározott előfeltételekhez lesz kötve, amelyek az állapotgépben egyértelműen modellezhetők. Például az ugrás végrehajtásának feltétele a "Grounded" (földön tartózkodó) állapot, míg a mozgás történhet földön és levegőben egyaránt. Az állapotgépek hierarchikus és egymásra épülő struktúrája lehetővé teszi az állapotok közötti átmenetek (pl. "Grounded" → "Jumping", "Grounded"→ "GroundedMoving", "Jumping" → "InAirMoving") pontos és bővíthető kezelését.
A téma jelentősége abban rejlik, hogy az állapotgépek használatával a játéklogika átláthatóbbá, modulárisabbá és könnyebben karbantarthatóvá válik. Az így létrehozott rendszer rugalmasan bővíthető új mozdulatokkal és állapotokkal, miközben a kód tisztább és strukturáltabb marad.

Témavezető: Dr. Hornyák Olivér (egyetemi docens)

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Halász Máté Sándor**; Neptun-kód: `T1TNWL` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Karaktermozgatás állapotgépekkel a Godot játékmotorban* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....
Hallgató

1.

A szakdolgozat feladat módosítása

szükséges (módosítás külön lapon)

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

A szakdolgozat bírálatra

bocsátható

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Contents

1 Bevezetés **1**

2 Koncepció **2**

2.1 Játékmenet 2
2.2 Használt technológiák 3
2.3 Fejlesztési Környezet Választása 4
2.4 Játékmotor Választása 4
2.5 Vizuális Világ Megtervezése 5
2.6 Mozgásrendszer megtervezése 8
2.7 Állapotok 9

1 Bevezetés

A videójátékok az elmúlt években egy dollármilliárdos iparággá nőtték ki magukat. Nem csak a rohamosan fejlődő technológiának, de a szoftveres ágon tett előre lepéseknek is hála. A 2000-es évek elején a videójáték gyártás mindössze a programozók (matematikusok) számára egy hobbi volt, ami az akkori hardverből a lehető legtöbbet való kihozásról szólt. Manapság a játékmotoroknak hála bárki elkezdhet játékokat gyártani, viszont ez nem azt jelenti, hogy sokkal egyszerűbb lenne a videójáték gyártók dolga. A rohamosan fejlődő ipar és a játékfejlesztés elérhetőségének a növekedése az elvárásokat is megnövelte a játékok iránt.
Mivel a számítástechnika a matematikából nőtte ki magát, főleg a kezdetekben, ezért sok gyakorlatot alkalmazunk belőle. Egyike ezeknek a gyakorlatoknak az állapotgépek alkalmazása egyes esetekben. Név szerint, azon esetekben ahol fontos a könnyű bővíthetőség és az, hogy ne kelljen újabb változókat létrehozni annak érdekében, hogy kiküszöböljünk egyes eseteket. A legnépszerűbb példa egy állapotgép használatára az a videójátékokhoz kapcsolódik. A kezdetekben ez nem volt olyan fontos, mivel maguk a játékok egyszerűbbek voltak, lásd: Doom, Wolfenstein, Elite (1984) ahol a legfontosabb a grafika megoldása és helyes kirajzolása volt. Manapság, ahol ún. "Omni-Movement" van a játékokban, ami egy olyan mozgásfajta, ahol a játékos teste a fejétől függetlenül mozog, tehát a test és a fej (kamera) mutathat két különböző irányba. Ezen esetben a játékosoknak közelről kell figyelni a mozgásukat és cselekedeteiket, és jól definiált struktúrákat kell létrehozni egyes mozgássorozatoknak. A játékosok képesek különböző mozdulatok végrehajtására, többek között: futás, ugrás, csúszás és vetődés. Ezek mind precíz beviteleket követelnek meg a játékostól és az állapotok pontos nyilvántartását. Ennek a kivitelezéséhez az állapotgépek használata elengedhetetlen, mind a mozgásrendszer kivitelezéséhez, mind az animációk helyes lejátszásához.
A témámat, viszont nem az Omni-Movement inspirálta, az én projektemhez csak példaképpen kapcsolódik, hogy demonstráljam egy sokkal bonyolultabb koncepción is a témámat és megmutassam, hogy az iparban is használatos. Sokkal inkább a [Shibuya-punk esztétika](Shibuya-punk) királya és a mai napig a Sega egyik legkevésbé elismert kiadása a: Jet Set Radio (2000), és az az által inspirált Team Reptile játék a: Bomb Rush Cyberfunk (2023). Ezekről sokan nem hallottak, viszont, ha azt mondom, hogy "Tony Hawk Pro Skater", akkor az már több embernek fog ismerősen hangzani. A koncepció ugyan az mind a kettőnél: gurulj és csinálj menő trükköket (miközben próbálod magad nem összetörni, természetesen). Ha a felszínt nézzük is már eléggé bonyolultnak néz ki a helyzet, mind mozgás, mind pontszám számítás szempontjából; és elkezd járni az agyunk azon, hogy mégis mennyi állapotot

kellet nyomon követniük a fejlesztőknek, hogy ezeket megvalósítsák. Annak érdekében, hogy egy kicsit mélyebbre ássak és megválaszoljam ezt a kérdést úgy döntöttem, hogy magam is nekiállok és létrehozok egy mozgásrendszert, amiben tesztelhetjük, hogy mennyire is érné meg állapotgépeket használni, miért és mennyivel eredményezne szebb, jobban strukturált, átláthatóbb, könnyebben bővíthető kódot, és gyorsabb programot, mint egy hagyományos if-and megoldás.
A szakdolgozat végére szeretnék egy teljesen működőképes és játszható mozgásrendszer-prototípust elkészíteni, amely bemutatja az állapotgépek fontosságát a mozgásrendszerek kialakításában és a játékfejlesztés más terein. Mind ezt a Shibuya-punk esztétikában többnyire magam által elkészített modellekből.

2 Koncepció

Az évek alatt sok olyan játékkal játszottam, amelyek mélyen kidolgozott mozgásrendszerekre vannak építve, ilyen műfajok például a: Movement-Shooter, bizonyos Sport játékok többnyire azok, amik gördeszkákra, görkorcsolyákra és a hasonlókra épülnek. Fontosnak tartottam itt megemlíteni a Movement-Shooter műfajt, mivel az ebbe tartozó játékok szerettették meg igazán velem a szofisztikált mozgásrendszereket és belőlük tanultam sokat. Ezen játékok tökéletesen összehangolják a finom mozdulatokat az aréna-lövöldék gyors döntéshozatalával és precíz célzás igényével. Erre már az eredeti Quake játék többjátékos módját is fel tudnám hozni, de manapság inkább az "Ultrakill" nevű játékot szokták felhozni, mint ékes példája. A projekt maga viszont egy játszható prototípus, amely egyben egy hordozható rendszer is, így a terv szerint nem lesz műfajhoz és programhoz kötve, hanem mint egy Lego darab, kivehető és átrakható egy másik programba.

2.1 Játékmenet

Játékról, mint úgy nem lehet beszélni, viszont hordozható rendszerről már inkább. A mozgásrendszert úgy terveztem meg, hogy akármilyen olyan környezetben használható legyen, amelyben a játékos karakteren, vagy esetleg nem-játékos karakteren (NPC-n) valamilyen gurulásra vagy csúszásra alkalmas eszköz van (pl.: gördeszka, görkorcsolya, síléc, jégkorcsolya). Ehhez a Tony Hawk's Proskater játéksorozatot és Bomb Rush Cyberfunk / Jet Set Radio játékmenetét és mozgásrendszerét vettem alapul. Azzal az eltéréssel, hogy a prototípusban, megnyerési pont nincs, mivel ez túlmutatott a prototípusnak megszabott feladaton.

2.2 Használt technológiák

A használt technológiákat két részre osztanám fel: Szoftver és Hardver. Most egyenlőre, csak egy rövidebb felsorolást tennék a technológiákról és a későbbiekben, amikor a megfelelő részekhez érek, bővebben beszámolok róluk.
Szoftver szempontjából:

- Blender-t használtam a 3Ds modellek elkészítéséhez.
- Krita-t használtam a Modellek textúráinak megrajzolásához és SLK_img2pixel-t használtam a színek állításához.
- A hangok megvágásához Audacity-t és LMSS-t használtam.
- A diagramok megszerkesztéséhez Draw.io-t használtam.
- A verziókövetéshez Git-et használtam.

Hardver szempontjából a szoftvert két különböző rendszeren teszteltem és fejlesztettem:

- Asztali Számítógép:
  - Windows 10
  - Processzor: AMD Ryzen 5 7600X 6-Core
  - AMD Radeon RX 6600
  - Memória (RAM): 32GB
- LENOVO IdeaPad Slim 3 15AMN8:
  - Kubuntu 25.10
  - Processzor: 8 x AMD Ryzen 5 7520U with Radeon Graphics
  - Videókártya: AMD Radeon 610M
  - Memória (RAM): 16GB

Megjegyezném, hogy ez nem jelenti azt, hogy csak ezen gépigényeket elérő rendszereken működne a rendszer, egyszerűen csak ezeken tudtam kipróbálni. Valamint a digitális koncepció rajzok elkészítéséhez és a textúrák megalkotásához az: XP-Pen Deco 02-őt használtam.

2.3 Fejlesztési Környezet Választása

A fejlesztési környezet választásakor több szempontot is figyelembe kellett vennem:

- Elérhetőség: Mivel a fejlesztést két különböző rendszeren, két különböző operációs rendszer alatt végeztem fontos volt számomra, hogy a fejlesztési környezet elérhető legyen mind a két operációs rendszeren.
- Programozási Nyelv Támogatása: A Godot-n belül van számos programozási nyelvhez támogatás, ezek a:
  - GDScript a Godot saját script nyelve,
  - C# a Godot Mono támogatásával,
  - Valamint C és C++ számos kiegészítő támogatásával.

Fontos volt, hogy ezek közül az egyikhez legalább támogatást nyújtson a fejlesztési környezet.

- Kényelem: A fejlesztési környezet által nyújtott kiegészítési, valamint emlékeztető segédletek fontosak voltak, mivel a rendszer sok hasonló kódrészt tartalmaz (pl. Dictionaries, Leképzések, stb.).
- **Integráció a motorral:** Ez egy bónusz pont a listán, de sokat segít, ha a fejlesztési környezet integrálja a játékmotort az egyszerű dokumentáció olvasás és metódus / függvény hívás érdekében.

2.4 Játékmotor Választása

A játékmotor megválasztása során fontos volt számomra az elérhetőség, kompatibilitás és a hordozhatóság.

- Elérhetőség: Mivel a szoftvert két különböző operációs rendszeren fejlesztettem ezért az elérhetőségi szempont elengedhetetlen volt. A játékmotorok mindig szóba jön a Unity és az Unreal Engine, viszont ezeknél az opcióknál a licensz kérdése is szóba jön. Nem egy utolsó szempont az se, hogy ezzel a rendszerrel én majd a jövőben tovább dolgozzak akármilyen fennakadás vagy hátráltatás nélkül.
- Verziókezelés támogatása: Akármilyen rendszer fejlesztése során fontos a megbízható verziókezelés. Háromdimenziós terekben
- Tapasztalat: Végül, de nem utoljára fontos volt a meglévő tapasztalat a választott motorral, hogy a lehető legjobbat tudjam kihozni a választott projektből reális időn belül.

Így mindezeket a szempontokat összevetve a választásaim végül a: Godot motorra, C# programozási nyelvre és a Visual Studio Code fejlesztőkörnyezetre esett.

2.5 Vizuális Világ Megtervezése

A vizuális világ, amint már említettem a Shibuya punk esztétikára alapszik. 1 Agresszív vonalak, elmosott, neon színek erőteljes használata és dinamikus kompozíciók definiálják. Műfajalkotó játéknak számít a Jet Set Radio (amire mostantól **JSR**-ként fogok hivatkozni).

Figure 1: Demonstráció az agresszív vonalakra

A JSR 2000-ben jött ki a DreamCast konzolra. És korszakalkotó volt a játék vizuális világa, 2 játékmenete és mozgásrendszere. Százezreket fogott meg és évekig utána nem volt semmi olyan játék, ami el tudta volna csípni, hogy mi is tette a játékot annyira ikonikussá. A játékmenetében is megtartja a dinamikusságot és az agresszív vonalakat, ami a játékosban azt az érzést kelti, hogy gyorsabb és pontosabb mint valójában.

Figure 2: Rövid gif a JSR játékmenetéről

Ahhoz, hogy pontosan replikálni tudjam ezeket a tulajdonságokat mélyen bele kellett magam ásnom a játék világába, de mivel egy eléggé régi játékról van szó, nem könnyen hozzáférhető. Szerencsére erre a problémára megoldást találtam egy 2023-mas ún. "Szerelmi Vallomásban" (Love Letter), a Bomb Rush Cyberfunk-ban 3 (amire mostatntól **BRCF**-ként fogok hivatkozni).

Figure 3: Reklámkép a Bomb Rush Cyberfunk-ról

A játék teljes mértékben a JSR-t veszi alapul és megpróbálja replikálni a játékot annyira, amennyire csak tudja 4, miközben újít és iterál a meglévő koncepciókon. A játékmenet kifinomultabb, mint a JSR-nak, de képes megtartani azt az érzést, amit annak idején az elődje keltet.

Figure 4: Játékbeli kép a Bomb Rush Cyberfunk-ról

Látszatra a két játék ugyan azt a stílust tartja, azzal a különbséggel, hogy míg a JSR egy földhöz-ragadtabb játékmenetet és világot ad át, mint ha egy, a 20-as éveiben lévő lázadó punk tini lennél, addig a BRCF egy sokkal sci-fi-sebb világba dob minket, ahol mindenkinek lehet egy jetpack a hátán, ami eszméletlen sebességekre gyorsít fel minket és rásegít a trükkjeinkre A vizuális világ megtervezésében tehát ezeket a szempontokat vettem alapul és ezek szerint alakítottam a világot és a karaktert.

2.6 Mozgásrendszer megtervezése

Most, hogy a vizuális világ megbeszélésre került, ideje áttérni arra, ami igazán fontos: a mozgásrendszer.
Ahhoz, hogy egy minél folyékonyabb mozgásrendszert tudjak alkotni sok tervezés és előre gondolás kellet. A véges automaták egyik előnye az, hogy nagyon könnyen bővíthetőek, így ha el is rontottam volna valamit, vagy esetleg kellett volna még egy állapot ahhoz, hogy az animáció vagy a mozgás jól jöjjön ki, akkor azt könnyen megtehettem volna. Ennek a tulajdonságnak köszönhetően volt egy kis mozgásterem a kísérletezésre, ami újoncként nagy előny.

2.7 Állapotok

Át kellet gondolnom, hogy hogyan is szeretném, hogy kinézzen egyes mozdulat és, hogy egyes, a játékos által megadott bemenetre, milyen mozdulat lehet végrehajtható, bizonyos időkben. Egyszóval, a mozgás-láncokat meg kellett alkotnom. Viszont, nem csak a játékos kezdeményezhet állapotváltozásokat, hanem a program maga is. Előfordulhatnak események, amelyek befolyásolják a játékos mozgását és egyben az állapotát is. Ahhoz, hogy megkülönböztessem a program eseményeit a játékos bementeitől elneveztem azokat az eseményeket, amik a játékos irányítása fölött vannak "E"-nek, mint "Event" és a játékos bemeneteit "I"-nak, mint "Input". Most már tárgyalásra kerülhet az állapot diagram.
Minden a belépéssel vagy leidézéssel (spawn) kezdődik. A felhasználó itt kapja meg az irányítást a karaktere fölött. 6

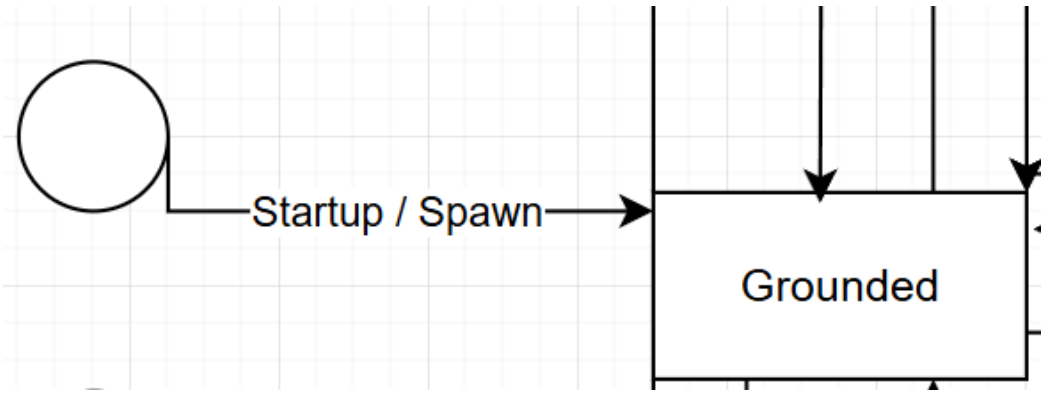

Figure 5: A program elindítása / a karakter újraéledése utáni állapot

A karakter minding egy szilárd, állható talaj fölött éled le, vagy éled újra, ezzel garantálva az állapotok közötti folytonosságot. Az alap állapot minden esetben az ún. "Idle", avagy nyugalmi állapot lesz, ahol a karakter egy állható, szilárd talajon van és játékos nem ad meg bemenetet.

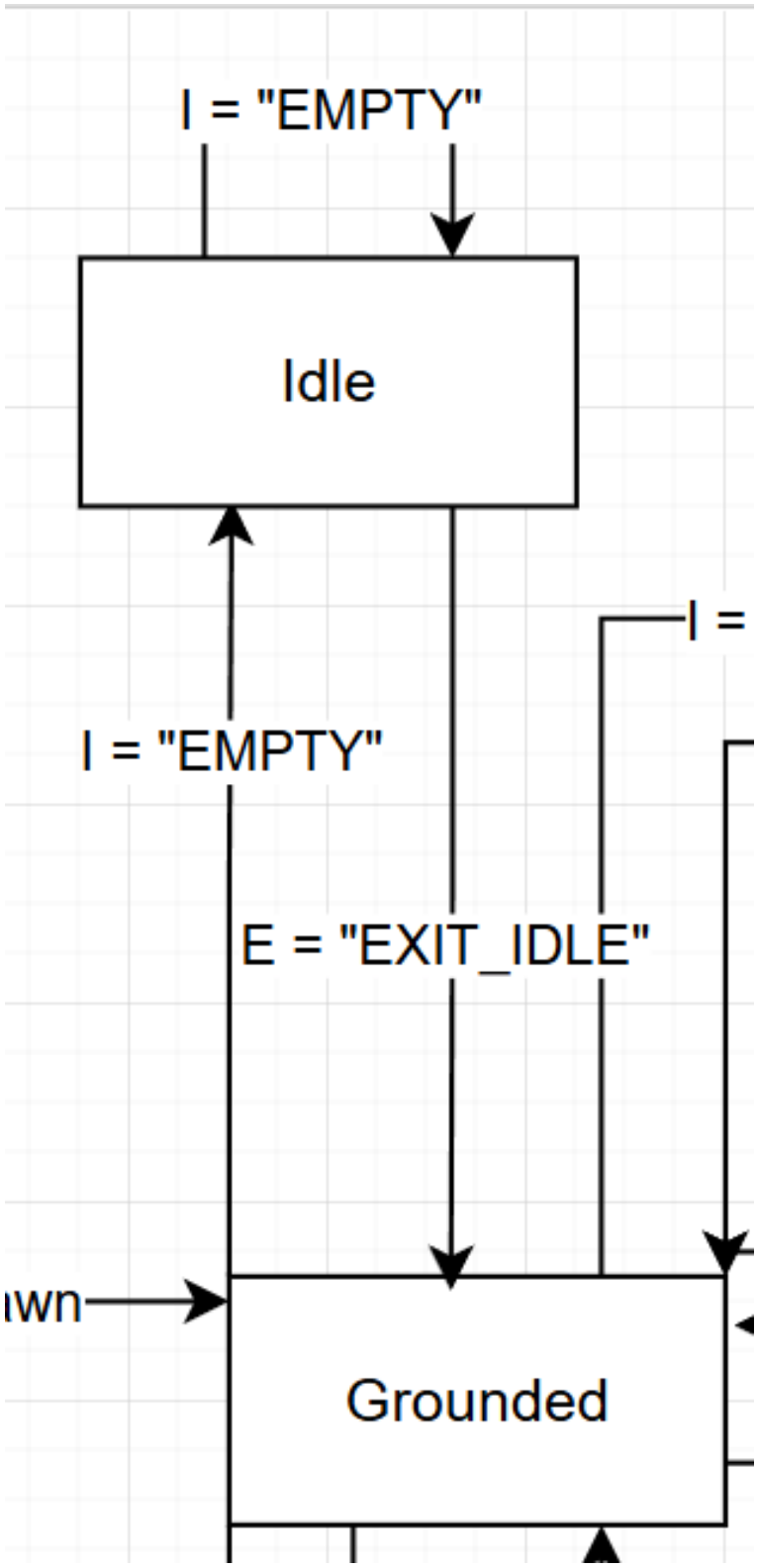

Figure 6: A program elindítása / a karakter újraéledése utáni állapot