Tartalom

[Miért érdemes játékfejlesztéssel foglalkozni? 2](#_Toc165406775)

[Játék Leírása 2](#_Toc165406776)

[Fejezetek rövid ismertetése 3](#_Toc165406777)

[Elémeti háttér 3](#_Toc165406778)

[Voxel Technológiák 3](#_Toc165406779)

[Voxel és voxeljáték definíció 3](#_Toc165406780)

[Kettő Dimenziós Voxelek (Terraria) 4](#_Toc165406781)

[Három Dimenziós Voxelek (Minecraft) 4](#_Toc165406782)

[Mesh Előállítás, Megjelenítés 5](#_Toc165406783)

[Miért a Marching Cubesra esett a választás 5](#_Toc165406784)

[Marching cubes 5](#_Toc165406785)

[Hol használatos a Marching Cubes algoritmus? 5](#_Toc165406786)

[Orvosi Tudomány 5](#_Toc165406787)

[CAD és szimulációs szoftverek 6](#_Toc165406788)

[Az algoritmus működése 7](#_Toc165406789)

[A Sokszöghalmaz Előállítása 9](#_Toc165406790)

[Lookup Table 9](#_Toc165406791)

[Edge Table 9](#_Toc165406792)

[Corner Table 9](#_Toc165406793)

[Pszeuorandom Generálás 10](#_Toc165406794)

[Perlin Noise 10](#_Toc165406795)

[Hivatkozások 11](#_Toc165406796)

# Miért érdemes játékfejlesztéssel foglalkozni?

A videójáték ipar az informatika fejlődésével együtt rohamos lépésben halad előre. Az informatika és azon belül a videójáték fejlesztés is egy nagyon összetett iparág, amiben mindenki megtalálhatja a saját stílusának, érdeklődési körének megfelelő irányt. A videójáték fejlesztés emellett egy magas profitot generáló terület, kifejezetten az elmúlt években.

A globális videojáték-piac 2022-ben 217,06 milliárd dollárra volt becsülve és 2023 és 2030 között 13,4%-os összetett éves növekedési rátára lehet számítani. 2030-ra vélhetően meg fogja haladni a 580 milliárd dollárd. A piac bővülése köszönhető az online játékok térhódításának, amelyet a széles sávszélességű internetkapcsolat széleskörű elérhetősége tett lehetővé, emellett a mobiljátékok iránti kereslet is növekszik, miközben a 3D játékokra is folyamatosan nagy a kereslet. (Grand View Research, n.d.)

A gazdaság hullámvölgyei ellenére a videojáték-ipar figyelemre méltó rugalmasságot mutatott és évről évre tovább növekszik. Amíg van igény a szórakoztatásra, addig lesznek lehetőségek a játékfejlesztésben.

A videójáték készítés szenvedély és munka is lehet egyben. Egy olyan karrier választása, amit élvezünk nagyon fontos. A videójáték egy egyedülálló termék, amely egyesíti a művészetet, a történetmesélést és a technikai megvalósítást. A játékfejlesztés általában különböző készségekkel rendelkező fejlesztőkből álló csapatokat foglal magában, köztük művészeket, programozókat, írókat és tervezőket. Az ebben az iparágban való munkavégzés lehetőséget ad kreatív önkifejezésre, legyen szó karakterek tervezéséről, a videójáték világának létrehozásáról, vagy a narratívák megalkotásáról. A videojáték-ipar karrierutak széles skáláját kínálja, a játéktervezéstől és programozástól a marketingig és a közösségi menedzsmentig.

A videojátékok globális hatókörrel rendelkeznek, és emberek millióinak életét érinthetik szerte a világon. Legyen szó szórakoztatásról, társadalmi kapcsolatok ápolásáról, vagy akár az oktatás és az empátia előmozdításáról, a videojáték-iparban végzett munka lehetővé teszi, hogy jelentős hatást érjünk el nagy léptékben.

A videójátékok alapjaiban az informatika szokásos "image"-e jelenik meg, mely a problémamegoldást, programozást és a különböző funkcionalitások megvalósítását jelenti. A videójáték fejlesztés, mint terület nagyon dinamikus, a technológia folyamatosan fejlődik, melynek eredménye képpen újabb és újabb kihívásokkal állítja szembe a fejlesztőket. Ennek eredménye képpen a videójátékok is egyre összetettebbek és realisztikusabbak. Ám nem feltétlenük kell a realizmust és a komplexitást üldözni, hiszen egyszerűbb grafikával és játékmenettel rendelkező játékok is emberek millióinak nyújtanak elfoglaltságot nap mint nap.

# Játék Leírása

**Ha már alakul az ötlet kibővíteni, átdolgozni ezt a leírást**

A játék megvalósítása Unity-ban történik. Ez több előnnyel is jár, mint például, hogy a játékmotort nem kell kifejleszteni. A játék világa Open World és egy random generált pályán játszódik.

Unity

A Unity a Unity Technologies által fejlesztett videójáték motor. A szoftverrel való videójáték készítés Windows-on, MacOS-en és Linuxon is lehetséges, a játékot futtatni pedig virtuálisan bármilyen rendszeren lehetséges, legyen az Xbox vagy PlayStation széria, Windows, Linux vagy Mac számítógép, vagy akár mobil. A Unity alkalmas 2D és 3D játékok készítésére is, valamint használják még egyéb interaktív tartalmak létrehozására, például építészeti látványtervek készítésére.

Open World

Az Open World játékok jellemzői, hogy a játékos nincs megkötve abból a szempontból, hogy a pályán hova mehet. Egy Open World játékban általában jelen van progresszió, amely végig vezeti a játékost sorban a világ különböző részein, de nincs megkötve a játékos a különböző területek közötti mozgásban. Az Open World játékok fontos eleme a fő történetszál mellett a világ szabad felfedezése, amely az irányított történet és progresszió mellett extra tartalmat szolgáltat a játékosnak és nagyon fontos a teljes játékélmény szempontjából.

Random Generálás

A random generálás a teljes pályára kiterjed. Mind maga a pálya és a pályán jelen lévő tartalom is minden új világ létrehozásakor különböző, de egy adott „Seed”-hez kötött. A Seed a játék egy olyan érték, amely a pszeudorandom algoritmusoknak egy kezdőértéket szolgáltat. Ugyan arra a Seed-re a játék ugyan azt a világot fogja visszaadni.

# Fejezetek rövid ismertetése

**ha már alakulnak a fejezetek akkor elkészíteni ezt a részt**

# Elémeti háttér

# Voxel Technológiák

## Voxel és voxeljáték definíció

A voxeljáték egy olyan videojáték, amelyben a környezet és a benne lévő tárgyak voxelekből állnak. A voxelek alapvetően térfogati pixelek, amelyek egy háromdimenziós rács értéket képviselnek. A hagyományos 2D pixelekkel ellentétben, amelyeknek csak szélessége és magassága van, a voxeleknek mélységük is van, ami lehetővé teszi teljesen háromdimenziós környezet létrehozását. A voxel és a voxeljáték definíciója elég laza és gyakran kéz a kézben jár a sandbox játékokkal. (Wikipédia, 2024)

A voxel-játékokban a játékosok gyakran úgy lépnek kapcsolatba ezekkel a környezetekkel, hogy módosítják vagy megsemmisítik a voxeleket, vagy létrehoznak új voxeleket, lehetővé téve a terep dinamikus deformációját. Mivel ez nem egy előre definiált folyamat, hanem a játékostól függ, hogy mit tesz, ezért egy ilyen játéknak egy olyan rendszert kell prezentálnia, amely előre definiált interakciókon lehetővé tesz „végtelen sok” konfigurációt. A voxeljátékok kettő jól ismert példája a Minecraft és Terraria.

## Kettő Dimenziós Voxelek (Terraria)

A Terraria egy 2011-ben kiadott 2D videójáték. A Re-Logic fejleszti, akik először Windows-ra adták ki, majd más PC- és konzolplatformokra is. A játék alapból sandbox jellegű, a felfedezésre és kreatív építésre inspirál első sorban egy procedurálisan generált pályán. A Terraria sok pozitív visszajelzést kapott a közösség felől, ennek köszönhetően 2022-ig több mint 44 millió példány kelt el belőle. A Terraria az eddigi egyik legkelendőbb videójáték. (Wikipedia, 2024)

Bár a voxel egy térfogati pixelként van értelmezve, a Terraria is a voxeljátékok közé sorolható. A Terraria egy kétdimenziós rácsot vesz alapul, amelyen belül különböző blokkok helyezhetőek le. Az előre definiált interakciók blokk-blokk és blokk-entitás (pl. játékos) szinten vannak jelen, ez teszi lehetővé a világ dinamikus változtatását és különbözteti meg a játékot más, statikus pályákkal rendelkező hasonló játékoktól.

**megvan a játék, csak kell egy screenshot megszerkesztve, hogy illusztrálja a négyzetrácsos felépítést**

## Három Dimenziós Voxelek (Minecraft)

A Minecraft egy 2011-ben kiadott sandbox játék, a legnagyobb példányszámban eladott játék a világon. A Mojang Studios által fejlesztett játékból több mint 300 millió darabot adtak el eddig. Korai tesztelési verziókat követően először 2009 májusában hozta nyilvánosságra a játékot az eredeti fejlesztője, Markus "Notch" Persson. 2011. november 18-án adták ki hivatalosan amikor Notch lemondott és Jens "Jeb" Bergensten vette át a fejlesztést. Hamar népszerűvé vált és a világ legkelendőbb videójátéka lett, közel 140 millió havi aktív játékossal. A Minecraft-ból adtak ki verziókat mobil és konzol kompatibilitással is. (Wikipedia, 2024)

A Minecraft egy 3D kockás, pixeles, procedurálisan generált világban játszódik, gyakorlatilag végtelen tereppel. A Minecraft több szempontból is egy technikai csodának számít. A világ annak ellenre, hogy véletlenszerűen generált nagyon precízen illeszti össze a sok különböző tartalmat, miközben játékbeli méretekkel számolva több mint 3.6 milliárd négyzetkilométer. Referenciának, a föld, mint bolygó felszíne félmilliárd négyzetkilométerre becsülhető.

A Minecraft-ben a voxelek blokkoként vannak jelen. Egy blokk a tér egy kockányi területét definiálja egy háromdimenziós kockaháló mentén. Ezek a blokkok interaktálnak egymással és az entitásokkal, például a játékossal.

**megvan a játék, csak kell egy screenshot megszerkesztve ami illusztrálja a kockás felépítést**

A Voxel technológia lehetővé teszi a nagyméretű környezetek rugalmas és hatékony megjelenítését, ami hozzájárult a voxel alapú játékok népszerűségéhez. Ezen túlmenően a voxeljátékokhoz gyakran társított kockás esztétika ikonikussá és felismerhetővé vált a játékközösségben.

**ha túl sok lenne a tartalom akkor vissza lehet venni a játékok „lényegtelen” történelmi hátteréből és csak belinkelni**

# Mesh (háló) Előállítás, Megjelenítés

A számítógépes grafikában használt mesh (háló) kifejezés, egy olyan adatstruktúrára utal, amely 3 dimenziós objektumok felszínét definiálja poliéder formájában. A poliéder egy olyan térbeli alakzat, amelyet pontok, élek és oldallapok definiálnak, vagy másszóval egy poliédert minden oldalról síkok határolnak. Poliéder például a kocka, de a gömb nem, hiszen a gömbnek nincs sík felszíne. A számítógépes grafikában íves felületeket is hálók segítségével jelenítenek meg, akkor azonban a felület le van egyszerűsítve síklapokra.

Egy háló 3 típusú komponensből áll. A pontokból (vertex), amelyek a sarkokat definiálják előállíthatók az élek, amelyek körbezárnak egy adott poligont. Egy háló használhat bármilyen N-csúcsú poligont a felszín definiálására, de a legelterjedtebb megoldás a háromszöget veszi alapul. Egy háromszög alapú hálóra Triangle Mesh-ként hivatkoznak.

## Unity Mesh

Egy háló definiálása többfajta adatszerkezettel is lehetséges. Az ilyen adatszerkezetek rendelik hozzá a pontokat a poligonokhoz, Unity esetében háromszögekhez. Unity-n belül egy Mesh adattípus kettő tömböt foglal magában:

Vertices

A vertices tömb egydimenziós adatszerkezet, amelyen belül 3 dimenziós vektorok (Vector3) találhatók. Ezek a háló pontjainak térbeli elhelyezkedését határozzák meg. Tulajdonképpen egy hosszú felsorolás az összes pontról. (Az ismétlődés meg van engedve)

Unity-n belül a maximális vertex szám egy hálón belül 16-bites buffer esetén 65535 vagy több mint 4 milliárd 32-bites buffer esetén. A 16-bites buffer az alapértelmezett, hiszen feleannyi memóriát használ, mint a 32-bites buffer.

Triangles

A triangles tömb szintén egydimenziós adatszerkezet, amely indexeket (Integer) tartalmaz, amelyek a vertices tömbre mutatnak. A tömb 3-as indexcsoportonkként van értelmezve. Minden hármas csoport egy háromszöghöz rendeli hozzá a háromszög csúcsainak a vertices tömbben definiált pozícióját indexelés segítségével. Ezekből következik, hogy a triangles tömb elemeinek száma mindig 3 többszöröse.

# Miért a Marching Cubesra esett a választás

A Marching Cubes algoritmus a Minecraft-al ellentétben a kockák szintje alá megy precizitás szempontjából. Míg a Minecraft-ban egy blokk vagy jelen van egy koordinátán, vagy nem, addig a Marching Cubes egy kockához 256 konfiguráció közül társít egyet. Ez lehetővé tesz a talaj lényegesen nagyobb „felbontásban” való megjelenítését.

Ez a precizitás két lépcsőben is jelen van.

Felezőpontos megoldás, Pozíció interpolációs megoldás

**illusztráció minecraft→rough terrain→smooth terrain**

**lehet egy szélesebb kép is, csak kell textúra a talajra amihez kell custom shader**

# Marching Cubes Algoritmus

A Marching Cubes algoritmus a számítógépes grafikában használt módszer, amellyel háromdimenziós skalármezőből poligon hálót lehet létrehozni. Különösen hasznos „isosurface” renderelésére. Az algoritmust William E. Lorensen és Harvey E. Cline fejlesztette ki az 1980-as években az amerikai General Electric számára folytatott kutatásuk közben. A kutatás célja az MRI és CT scan-ek effektív számítógépes megjelenítése volt. (Lorensen & Cline, 1987)

A háromdimenziós skalármező alatt egy térbeli négyzetrácsot kell elképzelni, amelynek minden pontja képvisel egy adott értéket. Ezeknek a térbeli pontoknak az értelmezésével lehet felületeket létrehozni, például sűrűségre hivatkozva. Például úgy, hogy a pontok, amelyek egy alakzat belsejében helyezkednek el magas konstans értékkel rendelkeznek, míg amik az alakzaton kívül vannak, alacsony értéket vesznek fel.

Az isosurface egy olyan felület, amely állandó értékű pontokat (pl. nyomás, hőmérséklet, sűrűség) képvisel egy térfogaton belül; más szóval, ez egy olyan folytonos függvény szinthalmaza, amelynek tartománya a 3 dimenziós tér. A pontok általában 3 dimenziós négyzetrács mentén helyezkednek el és a felvett értékeik határozzák meg a megjelenített alakzat felszínét. (Wikipedia, 2023)

# Hol használatos a Marching Cubes algoritmus?

## Orvosi Tudomány

A Marching Cubes algoritmus egyik elterjedt felhasználási területe az orvosi tudomány. Az MRI (Magnetic resonance imaging) és CT (computed tomography, vagy CAT, computed axial tomography) scan-ek a technológiájából adódóan e algoritmus használatához megfelelő adatokat generálnak.

Az ilyen scan-ek általában 2 dimenziós képekként vannak megjelenítve szeletenként, ahol egy kép egy adott pozícióban lévő metszetet szimbolizál szürkeárnyalatos módon. A különböző árnyalatok az adott metszet adott pontjainak az intenzitását mutatja. A CT és MRI szkennerek különböző szövetekről különböző intenzitású jeleket olvasnak le. Ezeknek a scan-eknek a precizitása nagyon magas, ezért könnyen különbséget tudnak tenni akár erek és izomszövet közt is. (National Institute of Biomedical Imaging and Bioengineering, 2022)

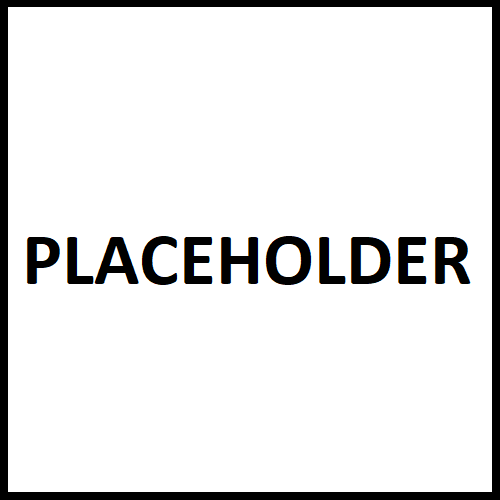
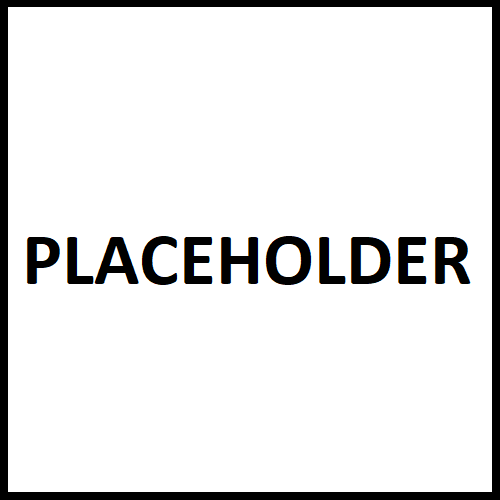
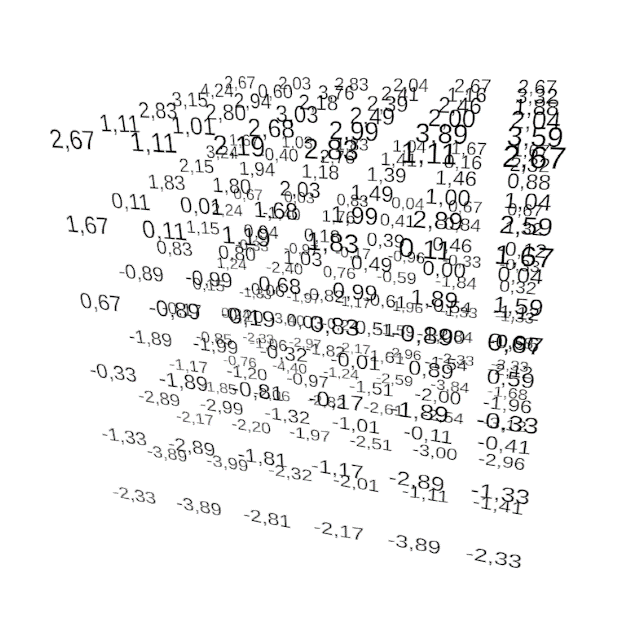
Ezeknek az értékeknek a megjelenítése lehetséges a Marching Cubes algoritmussal, úgy, hogy ha tudjuk a különböző szövetekhez tartozó jellemző konstansokat, akkor a scan értékeiből és térbeli pozíciójukból kiszámítja a különböző alakzatok felületét, például a csontoknak az alakját.

## CAD és szimulációs szoftverek

A Marching Cubes algoritmust CAD (számítógéppel segített tervezés) szoftverek is alkalmazzák, ahol a felhasználó saját maga készíti el az alakzatot, amelyet a számítógép megjelenít.

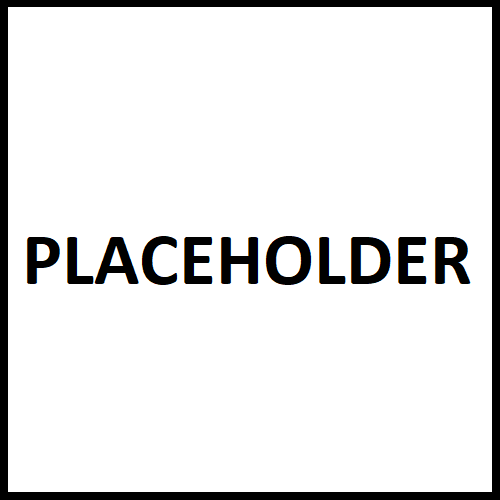
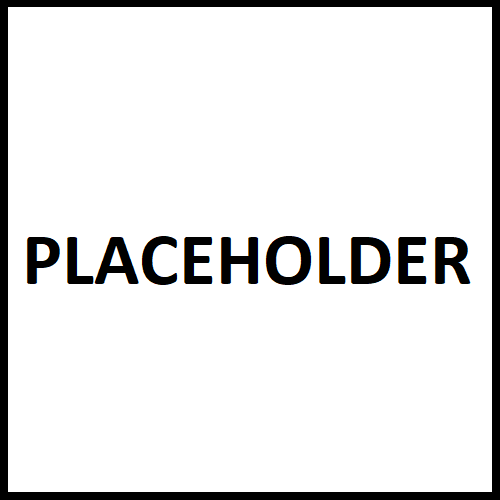
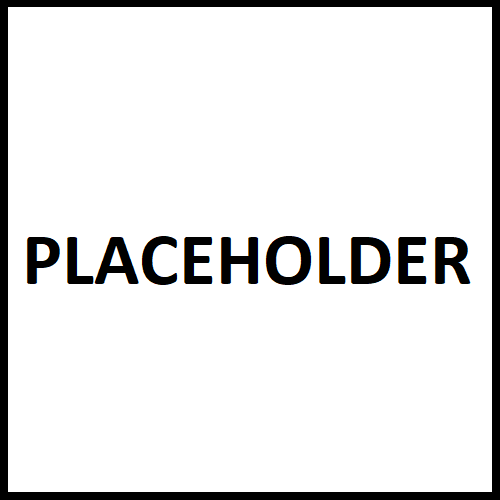
A Marching Cubes algoritmus szimulációk kiértékelésére is használatos, például folyadékdinamikai szimulációk vizualizálására. Egy Marching Cubes által megalkotott isosurface reprezentálhat repülőgép propeller által keltett lökéshullámokat.

# Az algoritmus működése

Skalármező: Az első lépés egy háromdimenziós skalármező meghatározása, ahol a tér minden pontja egy skalárértéknek felel meg. Ez a skaláris érték különféle tulajdonságokat jelenthet, például sűrűséget, hőmérsékletet vagy mesterségesen generált értékek is lehetnek.

Rácsfelosztás: A skalármezőt tartalmazó tér kockákból álló rácsra van felosztva. Minden kocka nyolc csúcsot tartalmaz, és nyolc skaláris értékkel van társítva, mindegyik csúcshoz egy. Ebből a meghatározásból fakad, hogy egy pont egyszerre nyolc kockának a csúcsa egyszerre (kivéve, ha a mező valamelyik szélén helyezkedik el). Ez a későbbiekben fontos lesz.

Felület meghatározása: Az algoritmus minden kockánál kiértékeli a skaláris értékeket a nyolc csúcsánál, és meghatározza, hogy a felület metszi-e a kockát. Ez úgy történik, hogy a skaláris értékeket egy előre meghatározott konstanssal hasonlítja össze, amely a vizsgált felületet reprezentáló állandó érték. A konstansnál kisebb értékkel rendelkező pontok például a felület alatt, míg a konstansnál nagyobb értékek a felület felett helyezkednek el. Ez egy bináris döntés a kocka mind a nyolc csúcsán. Ezen összehasonlítások alapján a kocka a 2^8 = 256 lehetséges konfiguráció egyikébe eshet.

Interpoláció: Ha a kocka konfigurációja ismert, az algoritmus meghatározza a felület metszéspontjait a kocka éleivel. A legegyszerűbb megoldás veszi a felezőpontot a két csúcs között és ott halad át a felület. A matematikailag korrekt megoldás a metszéspontokat a kocka csúcsainak skalárértékeiből interpolálja oly módon, hogy az értékek közötti különbség fogja meghatározni a felület metszéspontját és a kocka élén való elhelyezkedését.

Sokszög halmaz előállítása: Az előző lépésben kiszámított metszéspontok felhasználásával az algoritmus sokszögeket generál, hogy „megközelítse” a felület azon részét, amely metszi a kockát. A kocka minden konfigurációja egy adott sokszöghalmaznak felel meg.

Háló összeállítás: Végül az egyes kockákhoz létrehozott sokszögeket összeilleszti, hogy egy folytonos hálót képezzenek, amely a felületet képviseli. Ez a háló ezután hagyományos grafikai technikákkal megjeleníthető.

A Marching Cubes algoritmust széles körben használják olyan alkalmazásokban, mint az orvosi képalkotás, a tudományos vizualizáció és a számítógéppel segített tervezés (CAD), összetett háromdimenziós struktúrák skaláris adatokból történő megjelenítésére. Érdemes megjegyezni, hogy bár a Marching Cubes úttörő volt a bevezetésekor, az algoritmus későbbi változatai és fejlesztései történtek olyan problémák megoldására, mint a topológiai helyesség és a háló minősége.

# A Sokszöghalmaz Előállítása

Hogy a folyamat könnyebben értelmezhető legyen, az algoritmus három előre definiált segédhalmazt használ. Ezek a halmazok konstans értékeket tartalmaznak, amelyek irányítják az algoritmust, hogy elérjen egy adott konfigurációig, amelyet már használni lehet.

**+táblázatok forrása**

**+lookup table használatának technikai előnyei**

## Lookup Table

A Marching Cubes algoritmusban a kockák limitálva vannak 256 különböző konfigurációra oly értelemben, hogy mely éleken helyezkednek el a háromszögek csúcsai, amelyek az adott konckán belüli felületet reprezentálják. Ezek a konfigurációk előre ismertek, ezért fölösleges lenne őket újra kiszámolni minden egyes alkalommal. Erre az algoritmus egy „Lookup Table”-t használ, amely 256 sorral rendelkezik.

public static readonly int[,] TriangleTable = new int[,] {

{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{0, 8, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{0, 1, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},  
…

Minden sorban, felsorolás szerűen jelen vannak az éleknek az indexei, amelyeken egy háromszög csúcs jelen van. Egy kockán belül maximum 5 háromszög lehet jelen, ezért a „Lookup Table” egy 256 x 15 -ös kétdimenziós tömbként értelmezhető. A sorok indexei jelentik az adott konfigurációt, a sorokon belül pedig hármas csoportokban jelen vannak a háromszögek csúcsaihoz tartozó indexek. A kocka éleinek a pozíciója, vagyis a kezdő és végpontja ismert, tehát ha tudjuk melyik élen van az adott pont, akkor az él két csúcsa között csak interpolálni kell a pontnak a pozícióját.

## Edge Table

Ez a táblázat mindössze a kocka éleit definiálja úgy, hogy az algoritmust átirányítja a kettő megfelelő sarokhoz, amelyek az él kezdő és végpontjai. Az átirányítás a Corner Table-ben történik indexelés segítségével.

public static readonly int[,] EdgeTable = new int[12, 2] {

{0, 1},

{1, 2},

{3, 2},  
…

## Corner Table

Ebben a táblázatban a kocka csúcsai vannak jelen 3 dimenziós vektorok formájában. Ezeket a vektorokat hozzáadva a kockának a (0, 0, 0) csúcsához mind a nyolc csúcsot el lehet érni.

public static readonly Vector3Int[] CornerTable = new Vector3Int[8] {

new Vector3Int(0, 0, 0),

new Vector3Int(1, 0, 0),

new Vector3Int(1, 1, 0),

new Vector3Int(0, 1, 0),

new Vector3Int(0, 0, 1),

new Vector3Int(1, 0, 1),

new Vector3Int(1, 1, 1),

new Vector3Int(0, 1, 1)

};

Az Edge Table és a Corner Table használata nélkül is meg lehetne írni az algoritmust, ám mivel ezek az értékek konstansok és fontos, hogy mindig ugyan olyan sorrendbe következzenek a konfiguráció kiértékelése során, ezért egyértelmű, hogy ha kiemeljük őket konstans tömbökbe, az lecsökkenti a hibalehetőségek számát és kompaktabbá, jobban olvashatóvá teszi a kódot. Ennek a folyamatnak elhanyagolható műveletigénye van ahhoz képest, hogy mennyivel olvashatóbbá teszi a kódot.

Egy példa:

Az alább látható FOR ciklusban a kocka sarkai vannak kiértékelve. Ha a tömbön végigmegyünk a 0-ás indextől kezdve, akkor mindig ugyan abban a sorrendben fognak következni a csúcsok, amelyeket egy indexel elérünk, nem kell minden csúcsnak a háromdimenziós eltolását kiírni a (0, 0, 0) csúcstól.

int GetCubeCongif(float[] cube)

{

int configIndex = 0;

for (int i = 0; i < 8; i++)

if (cube[i] > terrainHeight)

configIndex |= 1 << i; //bitshift

return configIndex;

}

Ezeknek a tömböknek a „konstanssága” nagyon fontos, hiszen az algoritmus mindig ugyan abban a sorrendben kell, hogy kiértékelje az adott konfigurációt, főleg, hogy a kód szimplifikálása érdekében a konfiguráció meghatározásakor BITSHIFT-et használ. A bitshift csak akkor fogja a megfelelő konfigurációt visszaadni, hogyha a konfigurációk is sorban vannak és a csúcsok kiértékelése is minden egyes alkalommal ugyan abban a sorrendben történik.

# Pszeuorandom Generálás

A talaj megjelenítése után a következő lépés a talajnak a megalkotása. Erre sok különböző pszeudorandom algoritmus áll rendelkezésre. A választás a Perlin Noise algoritmusra esett egy nagyon egyszerű indok miatt. A Perlin Noise a Unity Mathf könyvtárának egy beépített funkciója és tökéletesen alkalmas talajmagasság meghatározására.

# Perlin Noise

**perlin noise**

**heightmap**

**marching cubes optimalizázás magyrázat kezdete**

# Hivatkozások

Grand View Research. (dátum nélk.). *Video Game Market*. Forrás: Grand View Research: https://www.grandviewresearch.com/industry-analysis/video-game-market#:~:text=b.-,The%20global%20video%20game%20market%20size%20was%20estimated%20at%20USD,USD%20583.69%20billion%20by%202030

National Institute of Biomedical Imaging and Bioengineering. (2022. Június). *Computed Tomography (CT)*. Forrás: Nibih: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct

Wikipedia. (2023. Szeptember 2). *Isosurface*. Forrás: Wikipedia: https://en.wikipedia.org/wiki/Isosurface

Wikipedia. (2024. April 24). *Minecraft*. Forrás: Wikipedia: https://en.wikipedia.org/wiki/Minecraft

Wikipedia. (2024. April 8). *Terraria*. Forrás: Wikipedia: https://en.wikipedia.org/wiki/Terraria

Wikipédia. (2024. Február 5). *Voxel*. Forrás: Wikipedia: https://en.wikipedia.org/wiki/Voxel