Tartalomjegyzék

Bevezetés

Napjainkban elterjed grafikai alkalmazások szinte kivétel nélkül egy fajta rács típust használnak a grafikai adatok megjelenítéséhez. Ez a rács a pixelrács, mely tulajdonképpen egy négyzethálónak fogható fel. A kétdimenziós rajzoló algoritmusok zöme is e rácshoz tartozóan született meg. Pedig a pixelrács nem a legideálisabb ezekre a célokra.

Ideális rács természetesen nem létezik, de jobb tulajdonságúak igen. Viszont e rács típusok alkalmazásának terjedését nagyban gátolja a szoftver és hardver gyártók bejáratott pixelrácsos megoldásai és algoritmusai.

Az egyik ilyen jobb tulajdonságokkal rendelkező rács típus a háromszögrács. Ezen dolgozat keretei között ismertetésre kerülnek a rács legfontosabb tulajdonságai, sajátosságai, illetve összehasonlítása a pixelráccsal.

Az általános ismertetés után, megvizsgáljuk a szakasz és körrajzoló algoritmusokat, ahol látni fogjuk, hogy nagyon egyszerű algoritmus segítségével ugyanolyan egyszerűen rajzolhatunk egyenest, mint a pixelrácson.

A következő fejezetben a terület kitöltő rutinokat vesszük közelebbről szemügyre, itt látni fogjuk, hogy ugyanazon algoritmusok kisebb módosításokkal vagy specializálással minként használható a háromszögrácson úgy, hogy az eredeti algoritmus alapötlete változatlan maradjon.

Megmutatjuk, hogy a háromszögrács a pixelráccsal szemben jobb telítettségi mutatókkal rendelkezik. Ezen tulajdonság ez egyik komoly fegyvertény lehet ezen rácstípus mellet.

Összehasonlítjuk a két fajta rács képtárolási tulajdonságait, ahol látni fogjuk, hogy a háromszögrács specialitása miatt esetlegesen sokkal alkalmasabb digitalizált képek tárolásához.

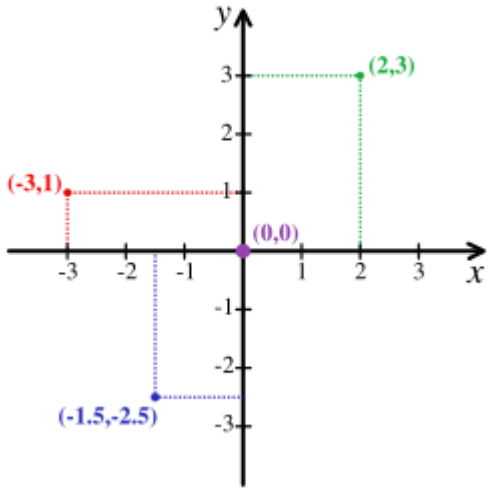
Az utolsó fejezetben megvizsgáljuk, melyik rács valójában mennyire is őrzi meg egy digitalizált képen lévő körvonalakat.

A dolgozat témakörei próbálják a lehető legtöbb alkalmazási területet lefedni, és minden területen megmutatni az alkalmazási lehetőségeket.

A pixelrács felépítése és tulajdonságai

Felépítése

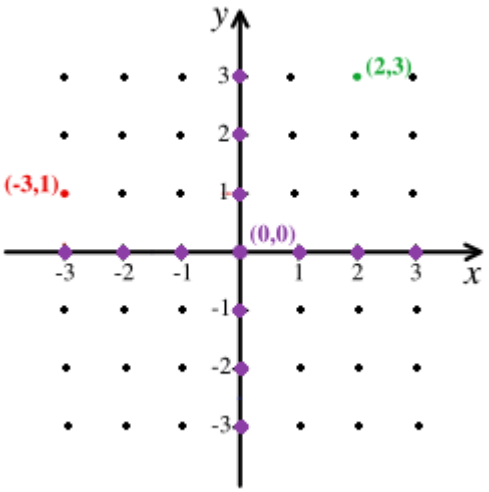
A pixelrácsot a derékszögű Descartes koordináta rendszerből fogjuk származtatni.



Az 1. ábrán látható tengely keresztből kivesszük azokat a pontokat melyek mindkét koordinátája egész szám, az így kapott szám kettesek fogják alkotni a pixel rács egyes rácspontjait, megfigyelhető, hogy az így kapott pontok tartalmazzák a koordináta rendszer két alap vektorát, u(1, 0) és v(0, 1).

Az u,v vektorok, és két n,m ∈ **Z** segítségével a pixelrács bármely pontja származtatható, generálható, például a (2,3) pont:

n*u + m*v = (n*1+m*0, n*0+ m*1) = (n, m) =(2,3), ha n = 2 és m = 3.



Kitüntet pontja az origó (0,0), amely a rendszer középpontja. Minden rácspont egy origóból induló nullvektort takar.

Egy P(x, y) pont origótól mért távolságán a hozzátartozó vektor hosszát értjük, tehát

$$|P|=\sqrt{x^2+y^2}$$

Műveletek

Legyenek $p_0(x_0, y_0)$ és $p_1(x_1, y_1)$ pontjai a rácsnak. A rács pontjain értelmezzük az összeadás, kivonás, és egy λ konstanssal való szorzás műveleteket:

$$p_0 – p_1 = (x_0-x_1, y_0-y_1)$$

$$p_0 + p_1 = (x_0+x_1, y_0+y_1)$$

$$\lambda p_0 = (\lambda x_0, \lambda y_0)$$

Ha $p_0 – p_1 = (0, 0)$, akkor a két vektor egyenlő, $x_0 = x_1$, $y_0 = y_1$ és $p_0 = p_1$.
Egy P(x, y) rácspont abszolút vektorán a P koordinátainak abszolút értékeiből elő álló vektort értjük, $P_{abs} = (|x|, |y|)$.

Távolság fogalom

Egy P(x, y) rácspont tengelyektől vet távolságán az abszolút vektorát értjük, oly módon, hogy az első, x tengelytől vet távolságot fogja a második, y koordináta jelenti, és fordítva.

Szomszédosság

Két pontra azt mondjuk, hogy közvetlen szomszédok, ha a két pont abszolút vektorának különbsége egyenlő az u, x tengely mentén szomszédosak, vagy a v, y tengely mentén szomszédosak, vektorral.
Ha a két pont két pont abszolút vektorának különbsége egyenlő az u+v ponttal, akkor a két pont átlósan szomszédos egymással,

Ha két rácspont közvetlen szomszédok, akkor nevezzük őket egy-szomszédos pontoknak, egyetlen koordinátában érnek el, ha átlósan szomszédosok, akkor legyenek két-szomszédosak, mindkét koordinátában eltérnek.

A valamelyik tengely mentén szomszédos rácspontok egymástól egyenlő, egységnyi távolságra helyezkednek el.
Az átlósan szomszédos pontok távolsága a Pitagorasz tételből következően $\sqrt{2}$.

Szimmetria

A pixelrács szimmetrikus, origó körüli $n*(\pi/2)$, $n \in \mathbf{Z}$ forgatással a rácspontok egymásba transzformálhatóak.
Legyenek $p_0(x_0, y_0)$ és $p_1(x_1, y_1)$ pontjai a rácsnak, ha p_0-t $(4n+1)*(\pi/2)$, $n \in \mathbf{Z}$,

órajárásával megegyező irányú origó körüli forgatással megkapjuk p_1-t, akkor igaz a következő összefüggés $x_0 = x_1$ és $y_0 = -y_1$, ha a forgatás szöge $(4n+2)*(\pi/2)$, akkor $x_0 = -x_1$ és $y_0 = -y_1$, ha a szög $(4n+3)*(\pi/2)$, akkor $x_0 = -x_1$ és $y_0 = y_1$, és végül ha a szög $(4n)*(\pi/2)$, akkor $x_0 = x_1$ és $y_0 = y_1$.

A $n*(\pi/2)$, $n \in \mathbf{Z}$ forgatások, végeredményei képen elő álló rácspontok, az eredeti pont tükrözéseiként előálló pontok, ahol a tükrözés tengelyei a koordináta rendszer alaptengelyei.

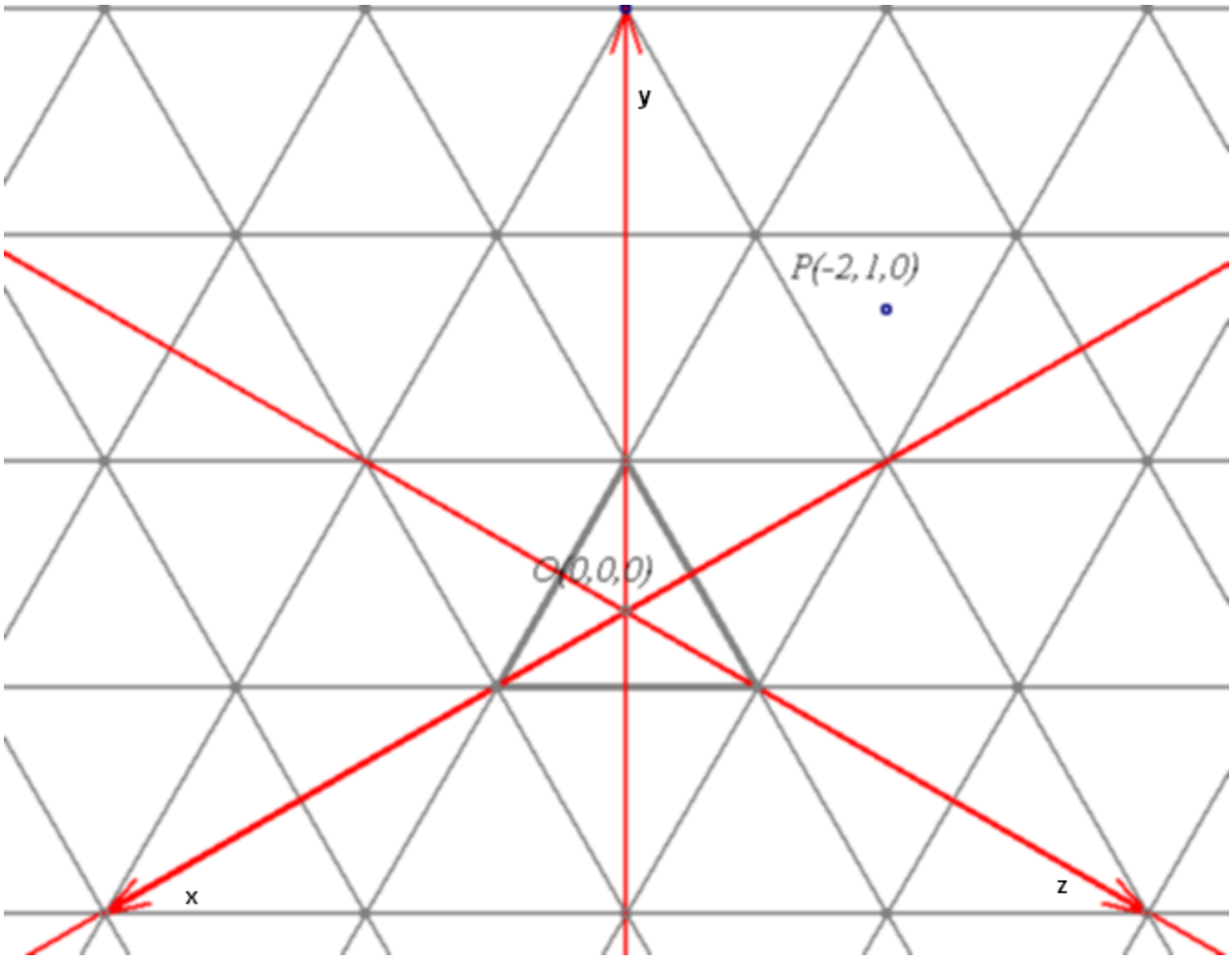
Mint látszik a rács önmagába transzformálható $4n*(\pi/2)$, $n \in \mathbf{Z}$ forgatással bármelyik irányban.

Minden egy-szomszédos rácspontot összekötve egy szabályos négyzethálót kapunk.

A háromszögrács felépítése és tulajdonságai

Felépítés

A háromszögrácsot egy háromtengelyű koordináta rendszerből fogjuk származtatni, melyben a tengelyek egymással páronként 120° zárnak be.



Mint az ábrán látható a tengelyek irányai speciálisak, ennek még nagyon fontos következményei lesznek.

A rácspontokat az alábbi módon képezhetjük:

- Legyen az origó a 3 tengely metszés pontja
- az origóból kiindulva mind 3 tengely mentén vegyünk fel pontokat egymástól $\frac{\sqrt{3}}{2}$ távolságra
- az x tengelyen fekvő pontokba állítsunk x tengellyel merőleges egyeneseket

- ismételjük meg az y és z tengelyeken fekvő pontokra is

A rendszer tengelyeire merőleges egyenesek metszés pontjai szabályos háromszögeket jelölnek ki, a rács pontjaink ezen háromszögek súlypontjai lesznek, például az ábrán a P pont. Ezek a pontok lesznek a rendszer egész koordinátás pontjai.
Tehát minden pont a rendszerben 3 koordinátával írható le, legyenek ezek sorrendben x, y, z, tekintsük ezeket az origóból induló (x, y, z) koordinátájú vektoroknak, amelyek koordinátái rendre a megfelelő tengelyen felvett értéket jelentik.

A rendszer középpontja itt is az origó O(0, 0, 0), viszont itt 3 egység vektor kell a pontok előállításához, legyenek ezek u(1, 0, 0), v(0, 1, 0), w(0, 0, 1), szükség van még 3 egész számra legyenek ezek rendre n, m, l. Megfigyelhető, hogy a rács bármely pontjára igaz, hogy a pont koordinátáinak összege vagy 0, vagy 1, ez egy jól definiált részhalmazt jelöl ki a szám hármasok halmazából, ennek értelmében igaznak kell lennie a következő össze függésnek: $n+m+l\in\{0,1\}$, ha ez nem teljesül olyan pontot kapunk, amely nem része a rácsnak.

Páros és páratlan rácspontok

Nevezzük párosnak a rácspontot, ha a koordináták összege 0, és páratlannak ha az összeg 1.

Ha adott a három egész szám, amely megfelel az összegükre vonatkozó feltételnek, n,m,l $\in$ **Z**, akkor ezek segítségével a rács bármely pontja származtatható, generálható, például a P(4, 5, -9) pont az alábbiak szerint bontható fel:

n*u + m*v + l*w = (n*1 + m*0 + l*0, n*0 + m*1 + l*0, n*0 + m*0 + l*1) = (n, m, l)

Legyenek $p_0(x_0, y_0, z_0)$ és $p_1(x_1, y_1, z_1)$ pontjai a rácsnak. A rács pontjain értelmezzük az összeadás és a kivonás műveleteket:

$$p_0 - p_1 = (x_0-x_1, y_0-y_1, z_0-z_1)$$

$$p_0 + p_1 = (x_0+x_1, y_0+y_1, z_0+z_1)$$

Ha $p_0 - p_1 = (0, 0, 0)$, akkor a két pont egyenlő, $x_0 = x_1$, $y_0 = y_1$, $z_0 = z_1$ és $p_0 = p_1$.

Látható, hogy a koordináta tengelyen elhelyezkedő pontok koordinátái speciálisak.
Ha P(x, y, z) az x tengely helyezkedik el, akkor $y=-x/2, z=-x/2$ a hányadosokat a

kerekítés szabályai szerint képezzük, tehát 5 tizedtől felfelé kerekítünk. Az y tengelyen elhelyezkedő pontra igaz $x=-y/2, z=-y/2$, és a z tengelyen elhelyezkedő pontra pedig az $x=-z/2, y=-z/2$ összefüggés igaz.

Egy P(x, y, z) rácspont abszolút vektorán, vagy abszolút pontján, a P koordinátainak abszolút értékeiből elő álló vektort értjük, P_{abs} = (|x|, |y|, |z|).
Az abszolút vektor, vagy pont általában nem pontja a rácsnak, mivel nem tesz eleget a koordináta összeg megszorításnak.

Szomszédosság

Két rácspontra azt mondjuk, hogy közvetlen szomszédok, ha a két rácspont különbségének abszolút vektora egyenlő az u, v, vagy w vektorral.

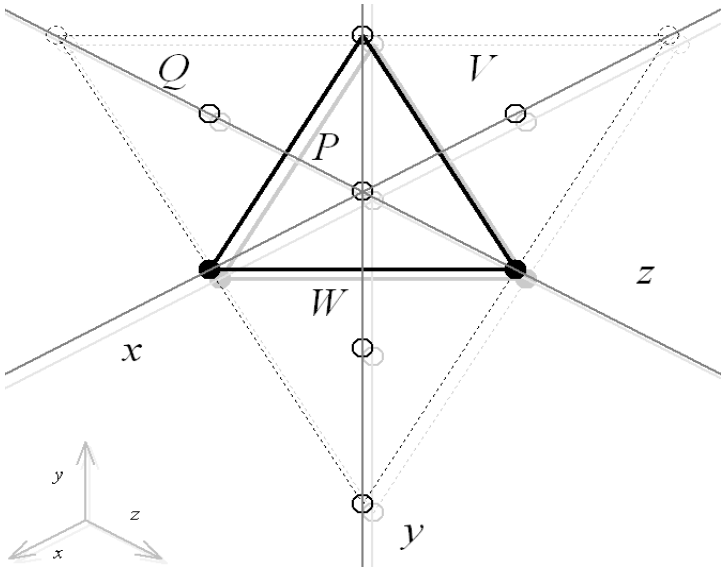
Ezeket a pontok nevezzük közvetlen, vagy egy-szomszédos pontoknak, ezek csak egy koordinátában térnek el.

Ha a két rácspont különbségének abszolút vektora egyenlő az u+v, vagy u+w, vagy az v+w vektorral, akkor a pontok két-szomszédos pontok.
Ha a két rácspont különbségének abszolút vektora egyenlő az u+v+w vektorral, akkor a két rácspontok három-szomszédos pontok.

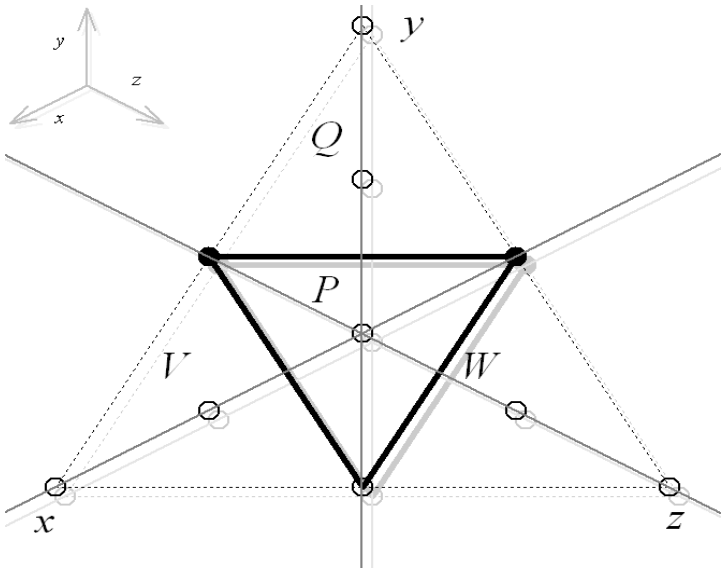
Egy páros pont minden egyszomszédos szomszédja páratlan pont, minden két-szomszédos szomszédja szintén páros, és minden háromszomszédos szomszédja páratlan.
Egy páratlan pont minden egyszomszédos szomszédja páros pont, minden két-szomszédos szomszédja szintén páratlan, és minden háromszomszédos szomszédja páros.

Tehát ha veszek a rácson egy egy-szomszédos pont sorozatot, akkor igaz lesz az, hogy a sorozat egymás után következő elemei váltakozva párosak és páratlanok, minden páros elem mindkét szomszédja páratlan, és minden páratlan elem mindkét szomszédja páros.

Páros és páratlan pontok közötti összefüggések



Ha megvizsgáljuk a páros és páratlan pontok egy-szomszédos szomszédait, akkor két nagyon fontos összefüggést figyelhetünk meg. Az ábrán látható P páros pont és az egy-szomszédos szomszédai, a P(p_x, p_y, p_z) koordinátáiból felírhatóak a Q,V,W pontok koordinátái: Q(p_x, py, p_z+1), V(p_x+1, p_y, p_z), W(p_x, p_y+1, p_z)Mint látható ha egy páros pontról valamely egy-szomszédos szomszédára megyünk, akkor valamely tengely mentén haladtunk pontosan egy egységet pozitív irányban. Hasonló összefüggést figyelhetünk meg a páratlan pontokra is:



Az ábrán látható P pártalan pont és Q,V,W páratlan egy-szomszédos szomszédai, itt is felírhatjuk a P(p_x, p_y, p_z) koordinátáiból a szomszédai koordinátái:Q(p_x, p_y-1, p_z), V(p_x-1, p_y, p_z), W(p_x, p_y, p_z-1)Látható, hogy ha egy páratlan pont valamely egy-szomszédos

szomszédára lépünk akkor valamely tengely mentén haladtunk pontosan egy egységet negatív irányban.

Távolság fogalom

Ha veszünk két pontot, amelyek egy-szomszédosak, akkor ezen két pont távolsága függ az elhelyezkedésüktől, függ attól, hogy melyik tengely mentén szomszédosak, ezért a háromszögrácson nincs egyértelmű távolság fogalom, szemben a pixelráccsal.
Egy P(x, y, z) origótól vett távolságán értsük a hozzá tartozó abszolút vektor koordinátáinak összegét.

Euklideszi távolság fogalom

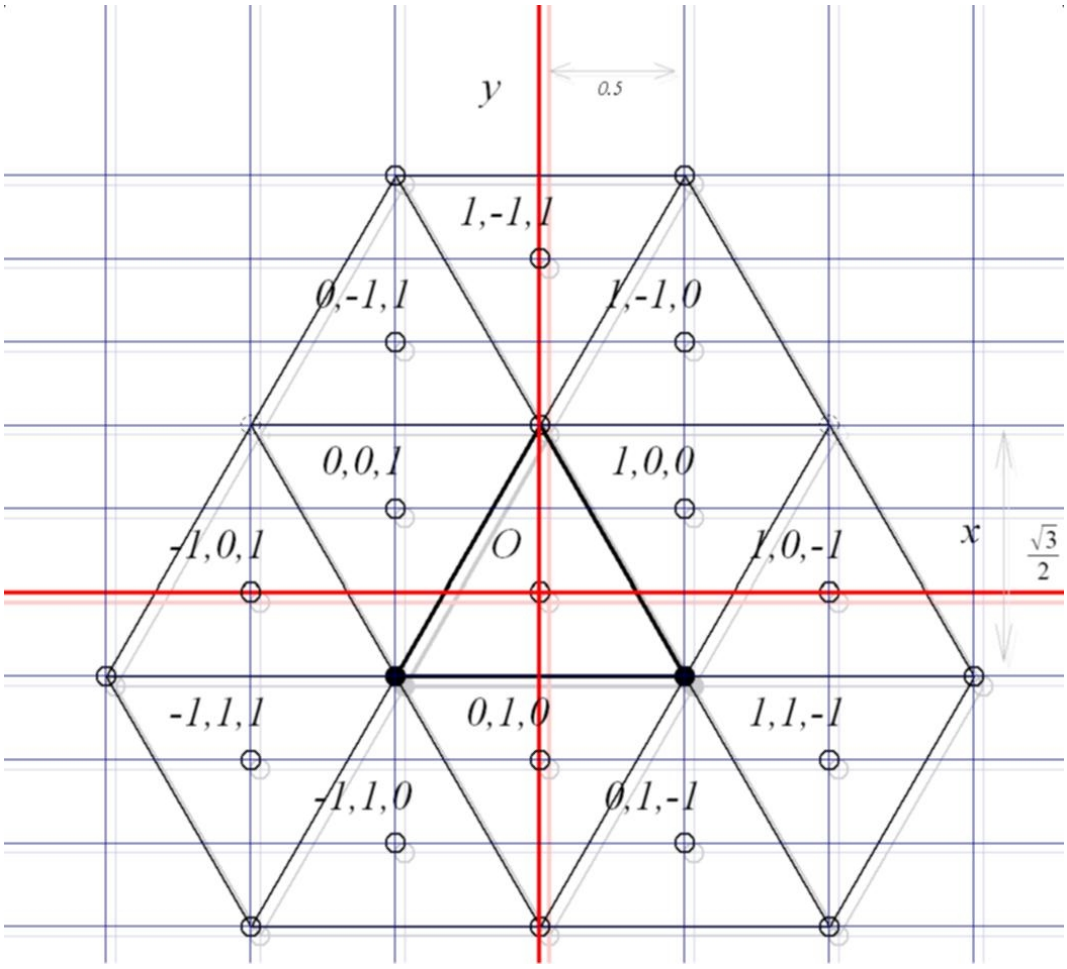
Értelmezhetünk két rácspont között Euklideszi távolságot is, ehhez határozzuk meg egy adott pont Euklideszi távolságát az origótól.
Mivel a rács szabályos egységnyi oldalhosszú háromszögekből épül fel, és a rács pontjai a háromszögek súlypontjainak feletetjük meg, felírhatjuk a következőket:

- A (0,0,0) háromszög súlypontja legyen az origó, legyenek a koordinátái O(0,0).
- Látható, hogy ezen pont összes egy-szomszédos páratlan szomszédja ugyanolyan távolságra található, illetve minden egy-szomszédos páratlan szomszédtól ugyanolyan távol van az origó.

Látni fogjuk, hogy a távolságot egy egy-szomszédos pontsorozattal fogjuk definiálni, amely az origóból indul és a keresett távolsághoz rendelt pontban végződik, továbbá a távolság független lesz a pontsorozatban szereplő elemektől.

Végül definiálunk egy egyszerű képletet a távolság gyors megadására, illetve a két pont közötti távolság gyors kiszámolására.

Descartes koordináta tengely elhelyezése a háromszögrácson



Illesszük rá a háromszögrács pontjait egy Descartes koordináta rendszerre, úgy hogy a két rendszer origója egy pontba essen.

Az ábra az origó és 1-3 szomszédos szomszédait ábrázolja, az ábra alapján felírhatjuk a következő táblázatokat:

A szomszédos pontok távolságai

Az egy-szomszédos szomszédok:

Szomszédos pont	**0,0,1**	**1,0,0**	**0,1,0**
Távolság az origótól	$(-0.5, \frac{\sqrt{3}}{6})$	$(0.5, \frac{\sqrt{3}}{6})$	$(0, -\frac{\sqrt{3}}{6})$

A két-szomszédos pontok

Szomszédos pont	**0,-1,1**	**1,-1,0**	**-1,0,1**	**1,0,-1**	**-1,1,0**	**0,1,-1**
Távolság az origótól	$(-0.5, \frac{\sqrt{3}}{2})$	$(0.5, \frac{\sqrt{3}}{2})$	**(-1, 0)**	**(1, 0)**	$(-0.5, -\frac{\sqrt{3}}{2})$	$(0.5, -\frac{\sqrt{3}}{2})$

A három-szomszédos pontok távolsága:

Szomszédos pont	**1,-1,1**	**-1,1,1**	**1,1,-1**
Távolság az origótól	$(0, 2\frac{\sqrt{3}}{3})$	$(-1, \frac{\sqrt{3}}{3})$	$(1, -\frac{\sqrt{3}}{3})$

Látható, hogy elég ismerni az egy-szomszédos pontok távolságát, hiszen ezek segítségével felírhatjuk a többi pont távolságát.
Például, írjuk fel az (1,1,-1) pont távolságát kicsit másképpen, vegyünk fel egy egy-szomszédos pont sorozatot, amely az origóból indul és a (1,1,-1) pontban ér véget:

(0,0,0), (0,1,0), (0,1,-1),(1,1,-1)

Haladjunk végig a pontokon és írjuk fel a koordináták távolságait:

Pontok	**A pontok közötti távolság**
(0,0,0) -> (0,1,0)	$(0, -\frac{\sqrt{3}}{3})$
(0,1,0) -> (0,1,-1)	$(0.5, \frac{\sqrt{3}}{6})$
(0,1,-1) -> (1,1,-1)	$(0.5, -\frac{\sqrt{3}}{6})$
összesen	$(1, -\frac{\sqrt{3}}{3})$

A végeredménynek ugyanezt kapjuk ha a (0,0,0),(1,0,0),(1,0,-1),(1,1,-1) pontsorozatot vesszük alapul a számoláshoz, bizonyítható, hogy az eredményt nem befolyásolja a választott egy-szomszédos pontsorozat, mivel ha bármelyik tengely mentén távolodnánk a vég ponttól, lesz a sorozatnak olyan részsorozata amely ugyanannyival közelebb visz a végponthoz, mint amennyire eltávolodtunk, és ezek a tagok az összegzésnél kiegyenlítik egymást, tehát az összeg szempontjából nem játszanak szerepet.
Vegyük észre, hogy elég az egyszomszédos pontok távolságát ismernünk a távolság meghatározáshoz.

Tengely menti távolságok

A páros pontok táblázatát már ismerjük, írjuk fel most a tengelyek mentén megtehető lépésekhez párosítva az értékeket, itt minidig pozitív irányban léphetünk 1 egységet:

Tengely	**+x**	**+y**	**+z**
Távolság a ponttól	$(0.5, \frac{\sqrt{3}}{6})$	$(0, -\frac{\sqrt{3}}{3})$	$(-0.5, \frac{\sqrt{3}}{6})$

Írjuk fel a páratlan pontokra is:

Tengely	**-x**	**-y**	**-z**
Távolság a ponttól	$(-0.5, -\frac{\sqrt{3}}{6})$	$(0, \frac{\sqrt{3}}{3})$	$(0.5, -\frac{\sqrt{3}}{6})$

Mint látható a két táblázat elemei csak előjelekben különböznek egymástól, ezért gyakorlatilag elég csak az egyiket ismerni.
Az első esetben még használjuk mindkét táblázatot a számoláshoz.

Két táblázatos módszer

Legyen adott P(p_x, p_y, p_z) pont, ennek a pontnak akarjuk meghatározni az Euklideszi távolságát az origótól.
Ahhoz, hogy az origóból eljussunk a P pontba minimum p_x egységet kell mennünk az x tengelyen p_y egységet az y tengelyen és p_z egységet a z tengelyen.
Ha valamelyik koordináta értéke pozitív, akkor vegyük alapul az első táblázatot, mivel annak segítségével tudunk pozitív irányba haladni, ha negatív, akkor a második táblázatot használjuk, az egységet felszorozva az egységnyi távolsággal megkapjuk a pont távolságát az adott tengely mentén, ezt így folytatva mind három koordinátára megkapjuk a tengelyeken mért távolságokat, ezeket összegezve pedig a végleges távolságot.

Egyszerűsített módszer

Ezen az algoritmuson egyszerűsíthetünk is, mivel elég egy táblázat is, illetve a fenti algoritmust alapul véve megadhatunk egy zárt képletet:

$$P_{táv}(x, y) = P_x * (0.5, \frac{\sqrt{3}}{6}) + P_y * (0, -\frac{\sqrt{3}}{3}) + P_z * (-0.5, \frac{\sqrt{3}}{6})$$

Az így kapott $P_{táv}$ pontra már alkalmazhatjuk a pixelrácsnál használt $|P_{táv}| = \sqrt{x^2 + y^2}$ képletet, amely megadja a P pont origótól vett Euklideszi távolságát.

Két tetszőleges rácspont egymáshoz viszonyított távolsága

Megadhatjuk két tetszőleges Q, V pontok egymáshoz viszonyított távolságát is, képezzük a két pont különbségét, a következő módon:

$$P = V - Q$$

Az eredményül kapott P pontra pedig alkalmazzuk a fenti eljárást, így gyakorlatilag a V pont Q-val közelebb toltuk az origóhoz, és kiszámítottuk az eredmény origótól mért távolságát.

Szimmetria

A háromszögrács szimmetrikus, origó körüli $n*(\pi/3)$, $n \in \mathbf{Z}$ forgatással a rácspontok egymásba transzformálhatóak.

Egy $P_0(x, y, z)$ pont $\pi/3$ szögű órajárásával megegyező irányú forgatásán a következő összefüggést értjük: $P_0' = (y, z, x)$.

A rendszernek három szimmetria tengelye van, ezek rendre a rendszer alaptengelyei, tehát x, y, z.

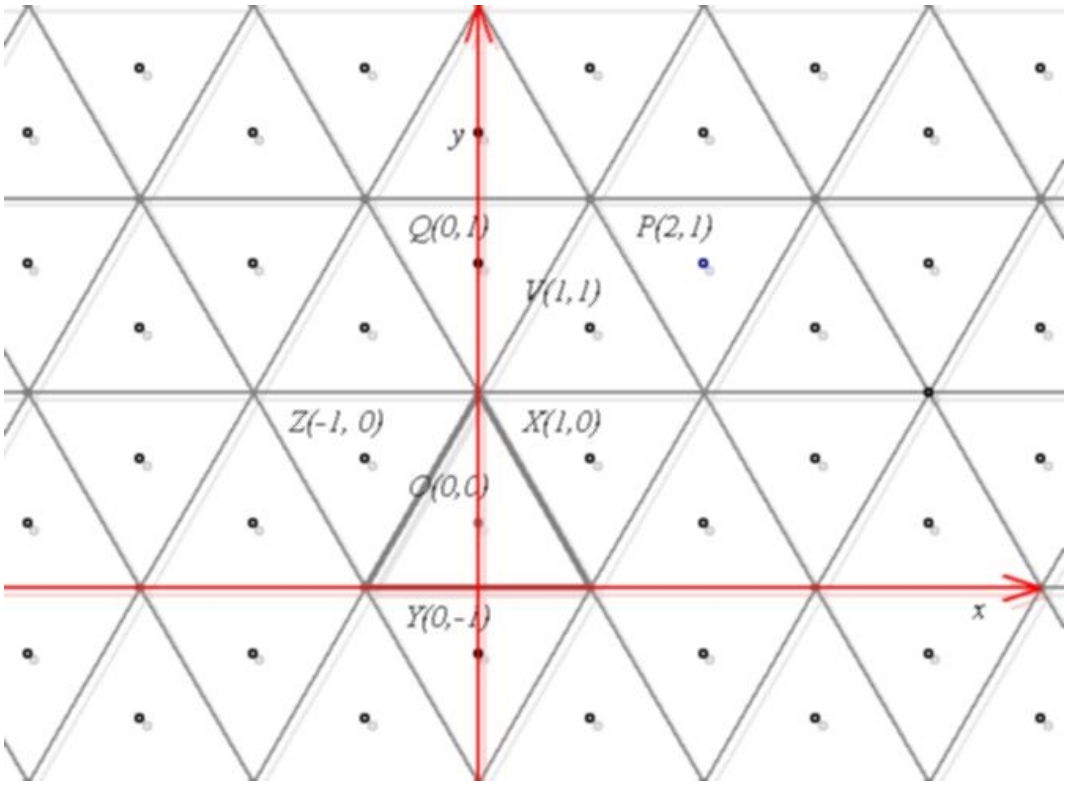
A rács önmagába transzformálható $6n*(\pi/3)$, $n \in \mathbf{Z}$ forgatással bármelyik irányban.

Ha minden egy-szomszédos rácspontot összekötünk, akkor egy szabályos hatszög rácsot fogunk eredményként kapni, ennek egyenes következménye, hogy az ilyen felépítésű háromszögrácsot származtathatnánk egy szabályos egységnyi oldalhosszú szabályos hatszögrácsból is ahol, ha a szomszédos hatszögek oldalfelezőinek metszés pontjait összekötjük, akkor megkapnánk az egységnyi oldalhosszú szabályos háromszögekből álló rácsot.

A háromszögrács kétkoordinátás reprezentációja

Minden P(x, y, z) $x, y, z \in \mathbb{Z}$ rács ponthoz megadhatunk egy vele ekvivalens Q(i, j) $i, j \in \mathbb{Z}$ pontot, a Q pont a P pont „kétdimenziós” reprezentációja, bármely P(x, y, z) ponthoz megadható egy egyértelmű vele ekvivalens Q(i, j) pont, és minden Q(i, j) ponthoz megadható egy egyértelmű vele ekvivalens P(x, y, z).

Származtatása



Az ábrán látható módon származtatjuk a rácspontok kétdimenziós reprezentációját.
A képzés hasonló a pixelrácsnál alkalmazott képzéshez, csak itt a pixel rács pontjainak egy torzított képével dolgozunk, mivel feltettük, hogy csak egész koordinátákra használjuk az eljárást, ezért például a Q és V pontoknak megegyezik az y koordinátája.

A konverziós művelet

Legyen a művelet jele: ∂ , és definiáljuk a mindkét alkalmazást külön-külön.
Definiáljuk először a $\partial(P(x,y,z))=Q(i,j)$ műveletet:

$$i = y$$

$$j = x - z$$

A $\partial(Q(i,j))=P(x,y,z)$ művelet már kicsit konyultabb:
Ha az i, j érték paritása megegyezik, akkor:

$$y = j$$

$$z = (-j - i) / 2$$

$$x = -z - j$$

Ha az i, j érték paritása különböző, akkor:

$$y = j$$

$$z = (1 - j - i) / 2$$

$$x = 1 - z – j$$

Műveletek a kétkoordinátás reprezentációban, ekvivalencia a háromkoordinátás reprezentációval

Legyenek p_0 ponthoz tartozó reprezentáció $q_0(x_0, y_0)$ és p_1 ponthoz tartozó reprezentáció pedig $q_1(x_1, y_1)$. Értelmezzük az összeadás és a kivonás műveleteket ezeken a pontokon:

$q_0 – q_1 = (x_0-x_1, y_0-y_1)$, ez egyenértékű a $p_0 – p_1$ művelettel.

$q_0 + q_1 = (x_0+x_1, y_0+y_1)$, ez egyenértékű a $p_0 + p_1$ művelettel

Ha $q_0 – q_1 = (0, 0)$, akkor a két vektor egyenlő, $x_0 = x_1$, $y_0 = y_1$ és $q_0 = q_1$, ekkor igaz a következő is: $p_0 = p_1$. Tehát,

$$\partial(q_0 - q_1) = p_0 - p_1$$

$$\partial(q_0 + q_1) = p_0 + p_1$$

$$q_0 - q_1 = \partial(p_0 - p_1)$$

$$q_0 + q_1 = \partial(p_0 + p_1)$$

Egyenes szakasz rajzolása 2 pontra

Pixelrácson alkalmazott eljárás

Legyenek adottak P0(x0, y0), P1(x1, y1), pontok keressük az ezekre illeszkedő egyenest. Az egyenest a rácson balról jobbra fogjuk megrajzolni, tehát igaz az, hogy a P0 pont van közelebb az origóhoz, ha ez nem igaz, akkor cseréljük fel a 2 pontot.

Induljunk ki az egyenes egyenletéből, ami a következő: $y = mx + B$.
Az m az egyenes meredekségét jelöli, erre 2 esettől eltekintve mindig igaz, a következő:
$0 < m < 1.$
Az m = 0 esetben egy x tengellyel párhuzamos egyenest kell elő állítani, az m = 1 esetben pedig egy y tengellyel párhuzamost.

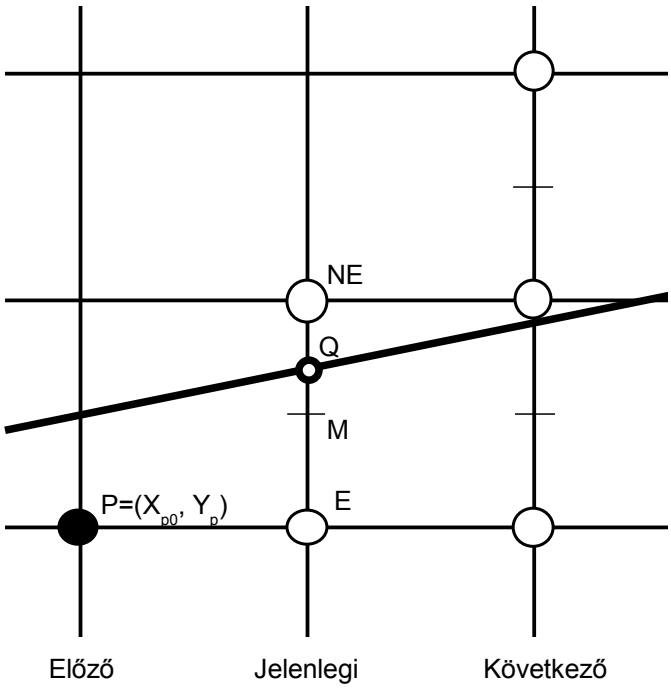
Az egyenletet kicsit másképpen megfogalmazva kapjuk az alábbi egyenlet az egyenesre, bármilyen alakban felírt egyenes egyenlet hozható ilyen alakra, ax + by + c = 0, ahol a és b egyszerre nem lehet nulla, mivel az már nem egyenes.

A triviális esetekben az első rácspont legyen a P0, a következőt mindig az előzőből kapjuk, az m=0 esetben az előző pont x koordinátáját növeljük 1-el, az m = 1 esetben az y-t, amíg el nem érjük a P1 pontot.

A nem triviális esetekben, 0 < m < 1, a következő eljárást fogjuk alkalmazni.

Legyen az utolsónak megjelenített pont a P(xp, yp) pont, ekkor a következő lépésben 2 rácspont közül tudunk választani, E, NE, a 2 pont közül azt fogjuk kiválasztani, amelyik közelebb van az elméleti szakaszfelező ponthoz.

Jelölje az E és NE pontokat összekötő szakaszok felező pontját M, és Q a kereset egyenes és a 2 pont közötti szakasz metszés pontját Q.



Az elv röviden: azt a képpontot válasszuk *NE* és *E* közül, amelyik közelebb van a *Q* metszésponthoz, azaz, hogy *Q* az *M* felezéspont melyik oldalán van, az döntsön a választásban.

Analitikusan a következőképpen számolhatjuk az egyenes pontjait:

legyen $F(x,y) = ax+by+c = 0$

$dx = x1-x0$

$dy = y1-y0$

$F(x,y) = dy \cdot x - dx \cdot y + B \cdot dx = 0$

$$F(x,y)\begin{cases} >0, ha\,(x,y)\,az\,egyenes\,alatt\,van, \\ =0, ha\,(x,y)\,az\,egyenesen\,van, \\ <0, ha\,(x,y)\,az\,egyenes\ fölött\,van. \end{cases}$$

Definiáljuk a választási kritériumot a következőképpen:

$$d = F(M) = F(x_p + 1, y_p + \tfrac{1}{2}) \begin{cases} > 0, ha\ M\ alul, NE, \\ = 0, ha\ M\ rajta, NE\ vagy\ E \\ < 0, ha\ M\ fölül, E \end{cases}$$

A következő pontnál egyszerűsíthetünk a választásnál:

Ha az előző lépésben az *E*-t választottuk, akkor legyen

$$\Delta_E = d_{új} - d_{régi} = F(x_p+2, y_p+1/2) - F(x_p+1, y_p+1/2) = dy,$$

ha *NE*-t választottuk, akkor legyen

$$\Delta_{NE} = F(x_p+2, y_p+3/2) - F(x_p+1, y_p+1/2) = dy - dx.$$

Az eljárást kezdjük a következő értékekkel:

$$d_{start} = F(x_0+1, y_0+1/2) = F(x_0, y_0) + dy - dx/2 = dy = dx/2$$

Számolhatunk végig egész aritmetikával is, ha az F(x,y) függvényt a következőképpen definiáljuk:

$$F(x,y) = 2 \cdot (dy \cdot x - dx \cdot y + B \cdot dx) = 0$$

A következő lépésben már a most kiválasztott pontból kiszámolt Δ értékekkel számolhatunk tovább.

Az eljárás a megvalósítható tisztán egész aritmetikai műveleteket használva.

Egy-szomszédos pontokból álló vonal rajzolása

A fent definiált algoritmus csak a koordináta tengelyekkel párhuzamos egyenesekhez állít elő egyszomszédos pontsorozatot. Az algoritmust kiegészítve könnyen készíthetünk olyan rajzoló algoritmust, mely minden esetben egyszomszédos vonalat ad eredményül.

Vegyük újra a fenti ábrán lévő esetet, itt az algoritmusnak megfelelően az NE pontot kell kiválasztanunk következő pontnak, de mivel a P és az NE pont nem egyszomszédos, így még egy pontot be kell venni az egyenes pontjai közé. Jelen esetben az E pontot kell felvennünk az egyenes pontjai közé.

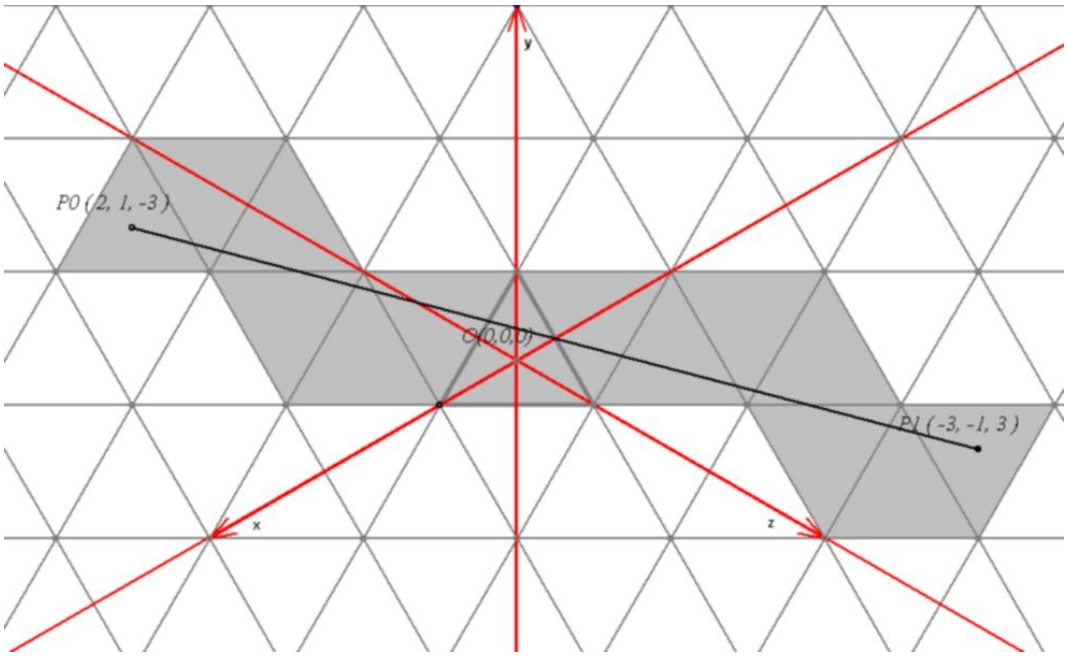
Általános esetben ha a következő pont nem egyszomszédos pontja a jelenlegi pontnak, akkor mindig 2 pont közül kell kiválasztani azt a pontot amelyiket be kell venni a pontok közzé. Ez a két pont mindig az aktuális és a kiválasztott következő pont által meghatározott négyzet másik két csúcsa, a fenti példában az E és a p pont felett elhelyezkedő pont.

A döntés alapja legyen az egyenes rajzolás irányától függően válasszuk az egyenes 'alatt' elhelyezkedő pontot, jelen esetben az E-t. Viszont ha például a generálás iránya fordított lenne akkor a p feletti pontot kell választanunk.

A döntést definiálhatjuk egy másik módszerrel is, ennek a módszernek a lényege, ha adott lépésben nem egyszomszédos pontot választottunk, akkor az egyenes az M pont fölött halad el, ilyenkor az egyenes pontjai közzé az M pont alatt lévő pontot is hozzá kell vennünk.

Háromszögrácson értelmezett algoritmus

A háromszögrácson értelmezett vonal, tulajdonképpen egy egy-szomszédos pontsorozat, amelyben szereplő pontok befoglaló háromszögeit metszi az ideális egyenes. A pontsorozatban szereplő pontokat egy inkrementális eljárással fogjuk meghatározni, amely kihasználja a rács speciális páros és páratlan pontokra megfogalmazott állításait.



Legyenek adottak $P_0(x_0, y_0, z_0)$, $P_1(x_1, y_1, z_1)$, pontok keressük az ezekre illeszkedő egyenest.

Első lépésben határozzuk meg egy növekmény vektort, amivel az egyenest fogjuk pontról pontra közelíteni, ehhez először határozzuk meg a két pont különbség vektorát, nevezzük ezt a vektort az egyenes egyszerű meredekségének, vagy meredekségnek:

$$v = P_1 - P_0$$

Ez a vektor gyakorlatilag megmutatja, hogy melyik tengely mentén, hányat kell lépnünk, ahhoz, hogy a P_0 -ből eljussunk a P_1 -be.

A növekmény vektor a v vektorból fogjuk kiszámolni, oly módon hogy a v vektor minden koordináta értéket elosztunk a v vektor hosszával, tehát:

$$v_n = (\frac{v_x}{|v|}, \frac{v_y}{|v|}, \frac{v_z}{|v|})$$

Ezt a vektort nevezzük az egyenes normált meredekségének, vagy egyszerűen normának.

A tovább lépéshez észre kell vennünk egy nagyon fontos megfigyelést, amely az összes vonalat alkotó egy-szomszédos pont sorozatra igaz a rácson:

Vegyük az ábrán szereplő $\overline{P_0P_1}$ vonalat(szakaszt), képezzük a v vektorát:

$$v=(-5,-2,6)$$

Mint látható a legnagyobb abszolút értékű elem a z koordinátán lévő érték, ezek után ha az ábrán szereplő szürkével színezett háromszögeket nézzük a z tengely mentén, észre vehető, hogy minden egyes z érték-re 2db háromszög jut. Szemléletesen, ha képzeletben ráteszünk egy vonalzót a z tengelyre merőlegesen amely a átmegy a P_0 ponton, akkor a vonalzó 2 olyan háromszöget is érinteni fog amely a szakasz része, ha a vonalzót egy egységgel eltoljuk a z tengely mentén a P_1 irányába, akkor a következő z pontban is ugyanezt fogjuk tapasztalni, és így tovább egészen amíg el nem érjük P_1 -et.

Ezt a megfigyelést megfogalmazhatjuk egy állításként is:

Az egyenes meredekség vektorában szereplő legnagyobb abszolút értékű koordinátához tartozó koordináta tengely mentén legalább annyit kell haladnunk, mit a másik két tengely mentén összesen, csak az ellenkező irányban.

Mit is jelent pontosan ez az ellenkező irány ?

A fenti példában szereplő $v=(-5,-2,6)$, esetén látszik, hogy a z tengely mentén kell haladunk pozitív irányban a legtöbbet, ha a P_0 pontból indulva akarunk végig lépkedni az egyesen akkor, minden lépésnél két eset lehetséges vagy lépünk egyet z irányban, vagy lépünk egyet x, vagy y irányban. Mivel pozitív irányba csak páratlan háromszögről tudunk lépni tehát ha z irányba megyünk, akkor egy páros háromszögre érkezünk, innen viszont csak páratlanra tudunk lépni, tehát valamely tengely mentén léphetünk egyet vissza negatív irányban.

Az algoritmus következő lépéseként határozzuk meg azt a tengelyt melynek mentén a legtöbbet kell lépnünk a végpont felé, ezt meghatározhatjuk a v és a v_n vektorból is, jelölje ezen tengelyt t , szükségünk van továbbá az irányra is, ha a kiválasztott koordináta értéke kisebb nullánál, akkor legyen páros, ellenkező eseteben páratlan, jelölje ezt a választást s .

Az s fogja meghatározni, hogy egy adott pontból milyen irányba is fogunk tovább menni, ha az aktuális pont paritás megegyezik az s -el akkor t mentén fogunk lépni, ha nem akkor a maradék 2 tengely valamelyike mentén.

A mindenkori aktuális pontot jelölje P_a .

Szükség lesz 2db számlálóra:

c_t - hányszor próbáltunk t tengely mentén lépni.

c_{nt} - hányszor próbáltunk nem a t tengely mentén lépni.

Az algoritmust egy egyszerű pszeudokód fogja prezentálni.

$P_a = P_0 \wedge c_t = c_{nt} = 0$

while($P_a \neq P_1$)

begin

if(P_a paritása = s)

begin

$c_t = c_t + 1$

P_u = P_a

i = t tengelynek megfelelő koordináta index

$P_{ui} = c_t * v_{ni}$, az i koordináta legyen egyenlő a norma i koordinátájának és az aktuális lépés szám szorzatának kerekített értékével.

If(P_u pontja a rácsnak és P_a és P_u egyszomszédosak)

P_a = P_u

end if

else

begin

$c_{nt} = c_{nt} + 1$

P_{u1} = P_{u2} = P_a

i = t tengelynek nem megfelelő egyik koordináta index

j = t tengelynek nem megfelelő egyik koordináta index és $j \neq i$

$P_{ui} = c_{nt} * v_{ni}$

$P_{uj} = c_{nt} * v_{nj}$

if(P_{u1} és P_{u2} is pontja a rácsnak és P_{u1} és P_{u2} is egyszomszédos P_a -val)

begin

$r_{u1} = |c_{nt} * v_{ni} - P_{u1}|$, $c_{nt} * v_{ni}$ szorzat törtrészének abszolút értéke

$r_{u2} = |c_{nt} * v_{nj} - P_{u2}|$, $c_{nt} * v_{nj}$ szorzat törtrészének abszolút értéke

if($r_{u1} = 0.0$)

P_a = P_{u1}

else

if($r_{u2} = 0.0$) P_a = P_{u2}

else

if($r_{u1} < r_{u2}$) P_a = P_{u1}

else

if($r_{u1} > r_{u2}$) P_a = P_{u2}

else

P_a = P_{u1} , válaszuk ki valamelyik pontot tetszőlegesen, de minden ilyen helyzetben konzekvensen ugyanaz az indexű pontot, legyen ez most az 1-es indexű.

end if

else

begin

if(P_{u1} pontja a rácsnak és P_a és P_{u1} egyszomszédosak)

P_a = P_{u1}

else

if(p P_{u2} pontja a rácsnak és P_a és P_{u2} egyszomszédosak)

P_a = P_{u2}

end else

end else

end while.

Magyarázatra szorul a következő pont kiválasztása abban az esetben amikor nem a t mentén léptünk és mindkét (P_{u1} és P_{u2}) keletkezet új pont pontja a rácsnak és mind kettő egyszomszédos P_a -val.

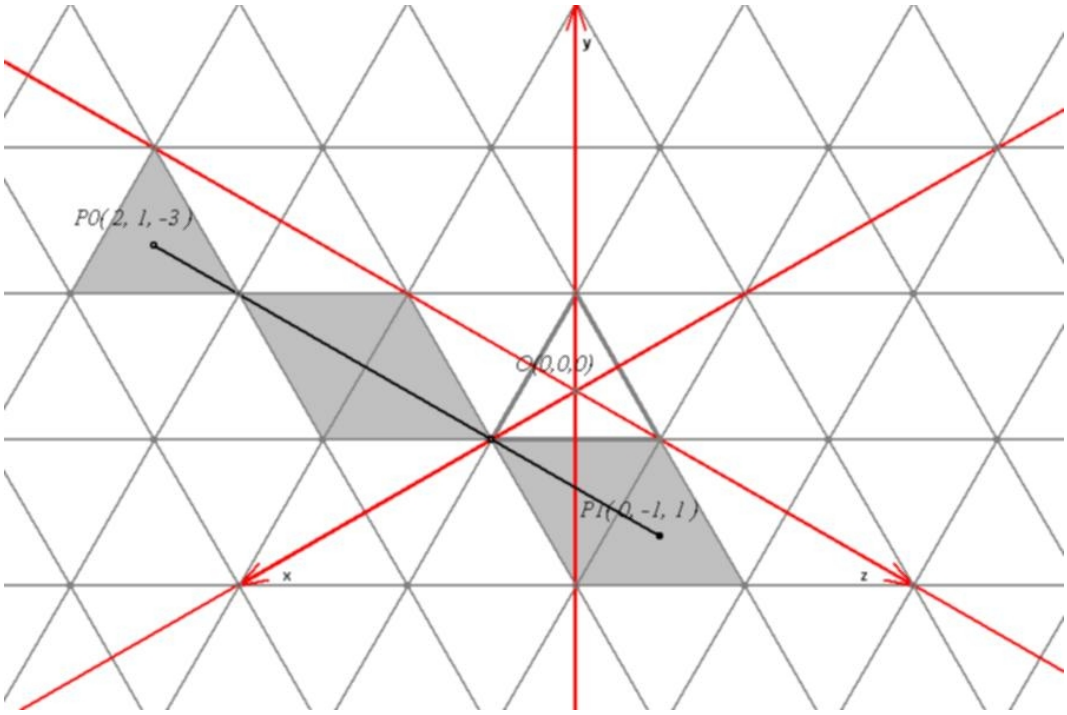
Mindkét pontra meghatározzuk a pont kiszámításakor használt koordináta kerekítéssel elhagyott törtrészének az origótól vett távolságát. A két pont között az alapján fogunk dönteni, hogy melyik is esik közelebb a pontot tartalmazó háromszög súlypontjához.

Ha valamelyiknek a maradéka megegyezik 0-val akkor egybe esik a súlyponttal.

Mivel mindkét pont csak egyetlen koordinátában tér el a saját háromszögének súlypontjától ezért elég ha csak ezen koordináták távolságait hasonlítjuk össze.

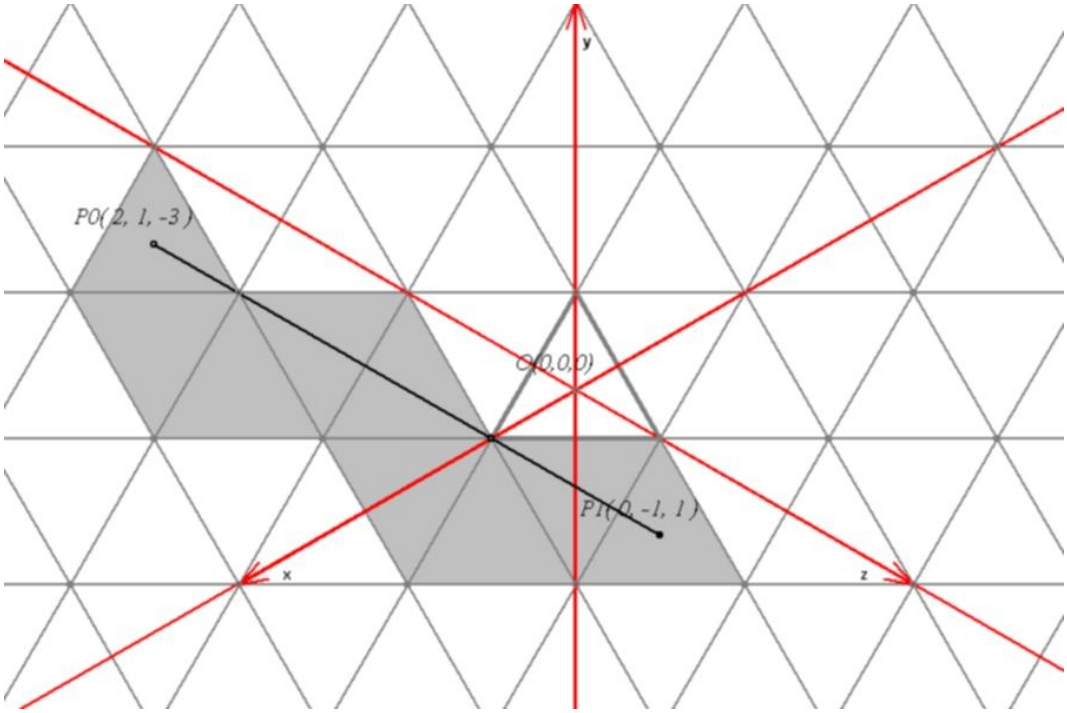
A két távolság közül a kisebbik az van közelebb a saját háromszögének súlypontjához, tehát ezt fogjuk választani.

Ha mindkettő azonos távolságra van akkor elméletben bármelyiket választhatjuk. Ilyen eset akkor fordulhat elő, ha az egyenes párhuzamos valamelyik tengellyel. Ezekben az esetekben az ideális vonal nem egyszomszédos pontsorozatot metsz ki a rácsból.



Mint látható ebben az eseteben nem egy olyan pontsorozaton megy át az ideális egyenes amely váltakozva tartalmaz egyszomszédos és háromszomszédos pontokat.

Ha nem engedjük az ilyen pontsorozatok vonalat alkossanak a rácson akkor fent említett bizonyos döntésnél válaszuk mindig valamelyik pontot konzekvensen. Az eredmény:



Az ideális vonal által kijelölt pontsorozat erősen eltorzult és nem kelti tengellyel párhuzamos egyenes látszatát.

Ha megengedjük az ilyen speciális vonalakat is akkor módosítanunk kell a vonal alap definícióján, és ezeket a speciális pontsorozatokat is hozzá venni azon pontsorozatok halmazához amelyek vonalat alkotnak a háromszögrácson.

A fent megadott algoritmus kisebb módosítással képes ezen vonalak kezelésére is.

Az ilyen speciális helyzetű vonalak felismerhetőek, az ilyen vonalak meredeksége speciális alakú:

$v_x+v_y+v_z\in[0,1]\wedge v_y=v_z$ - x tengellyel párhuzamos egyenes

$v_x+v_y+v_z\in[0,1]\wedge v_x=v_z$ - y tengellyel párhuzamos egyenes

$v_x+v_y+v_z\in[0,1]\wedge v_x=v_y$ - z tengellyel párhuzamos egyenes

A rajzoló algoritmus elején ellenőrizhetjük ezen feltételek, ha valamelyik teljesül, akkor egy speciális eljárással rajzoljuk ki a vonalat, ha nem akkor maradunk a hagyományos

eljárásnál.

A v és a v_n vektorok kiszámítása ugyanaz, illetve a hagyományos algoritmus megkezdése előtt már rendelkezésre állnak. Jelölje t azon tengelyt melyre teljesült, a párhuzamossági kritérium, jelölje s az irányt ugyanúgy mint eddig.

Belátható, hogy a legnagyobb koordináta mindig az lesz amelyik tengellyel párhuzamos az egyenesünk, mivel ha a másik két koordináta egyenlő, akkor a harmadiknak legalább akkora az abszolút értéke mint az első kettő összegének abszolút értéke.

$P_a = P_0$, $Q = (0,0,0)$, $V = (0,0,0)$

if($v_s \leq 0$)

$Q_s = 1$, $V_{i \neq s} = -1 \wedge V_{j \neq i,s} = -1$

else

$Q_s = -1$, $V_{i \neq s} = 1 \wedge V_{j \neq i,s} = 1$

while($P_a \neq P_1$)

begin

if(P_a paritása = s) P_a = $P_a + Q$

else P_a = $P_a + Q + V$

end while

Kör rajzolása

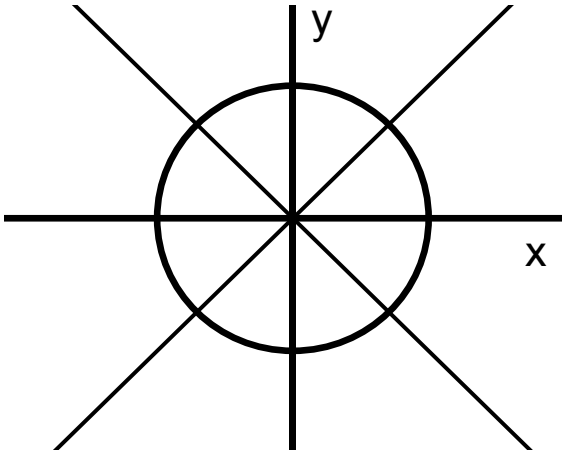
Az r sugarú kör egyenlete $x^2+y^2=r^2$. Az egyenletet átrendezve a következő összefüggést kapjuk: $F(x,y)=x^2+y^2-r^2=0$. A függvénybe behelyettesítve a sík pontjait, eldönthetjük, hogy az egyes pontokról, hogy belső, külső pont, vagy éppen pontja a körvonalnak. Ha az F(x,y) < 0, akkor belső pont, ha >0, akkor külső pont, ha pedig =0, akkor éppen pontja a körvonalnak.

Midpoint algoritmus

A körvonal maghatározásához a pixelrácson, egy a vonalrajzoláshoz nagyon hasonló módszer fogunk használni, mely a kör adott pontjában felvett érintő segítségével fogja meghatározni a körvonal pontjait. Minden eseteben origó középpontú kört rajzolunk, melynek pontjait a kirajzolás során el toljuk a rajzolandó kör középpontjának koordinátáival.

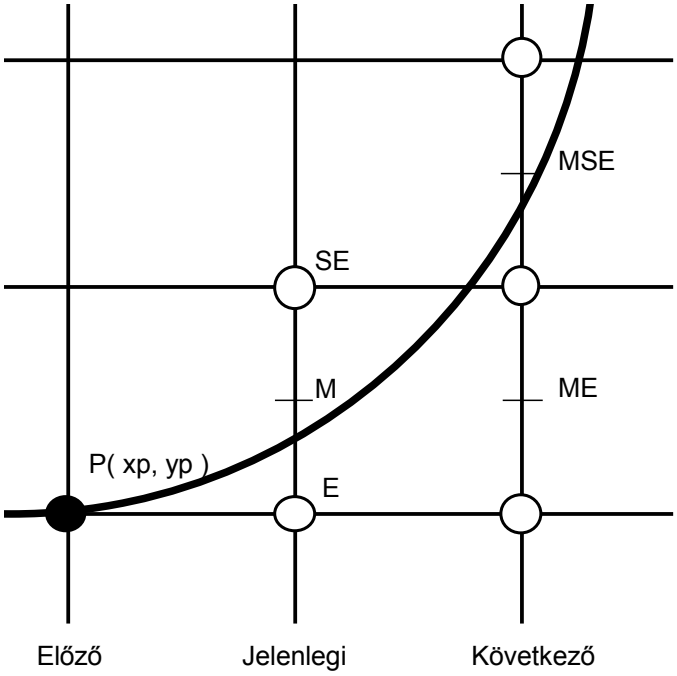
Nyolcas szimmetria

Használjuk ki a kör szimmetriáját, ahelyett, hogy az egész kört megrajzolnánk elég csak egy kör cikket kiszámolni, a többi pontot ezekből tükrözések segítségével előállíthatjuk. A triviális tengelyes tükrözések segítségével elég csak egy negyed kört megrajzolni. Azonban ez még nem a legoptimálisabb közelítés, ugyanis felvehetjük a Descartes koordináta tengelyek szögfelező egyeneseit, amelyek szintén szimmetria tengelyek, így nyolcadokra osztva a síkot. Ezen felosztás mellet pedig már elegendő csak egy nyolcad kört megrajzolni.



Az algoritmus rövid leírása

Legyen az aktuális kivilágított pixel a P, az ideális körvonalhoz legközelebb eső pont. Az eljárás ugyanaz mint a vonal rajzolás eseten, a következő lépésben két pixel (E,SE) közül kell kiválasztani az ideális körívhez közelebb esőt.



A kiszámolt körív minden pontjában a körérintőjének meredeksége -1 és 0 között van. Ebből adódóan a következő kirajzolandó pont az (xp+1, yp), vagy az (xp+1, yp+1) lehet. Jelöljük az E és SE pontok által meghatározott szakasz felező pontját M-el. Ha körvonal M alatt halad el, akkor E-t választjuk, ha felette, akkor SE-t.

Jelöljük d-vel az F(M) értékét:

$$d = d_{old} = F(M) = F(x_p + 1, y_p + \frac{1}{2}) = (x_p + 1)^2 + (y_p + \frac{1}{2})^2 - r^2$$

Ha d<0, akkor az E-t fogjuk választani, és ekkor d új értéke a következő lesz:

$$d_{new} = F(ME) = F(x_p + 2, y_p \frac{1}{2}) = (x_p + 2)^2 + (y_p + \frac{1}{2})^2 - r^2 = d_{old} + (2x_p + 3)$$

$$d_E = d_{new} - d_{old} = 2x_p + 3$$

Ha d>= 0 akkor az SE-t választjuk, ekkor a d értékek a következőféleképpen alakulnak:

$$d_{new} = F(MSE) = F(x_p + 2, y_p + \frac{3}{2}) = (x_p + 2)^2 + (y_p + \frac{3}{2})^2 - r^2 = d_{old} + (2(x_p + y_p) + 5)$$

$$d_{SE} = 2(x_p + y_p) + 5$$

Az egyenes rajzolással szemben a itt a differenciák értéke az xp és yp lineáris függvénye, ezért ezeket mindig az adott pontban megfelelő értékekkel újra kell számolni.

Az algoritmust a (0, -r) pontból indítva a kezdő differencia értéke:

$$d_0 = F(1, r + \frac{1}{2}) = \frac{3}{4} - r$$

Az eljárást addig ismételjük az eredményül kapott pontokra amíg a teljes nyolcad kör pontjait meg nem kaptuk. Utána az eredményül kapott pontok tükrözésével a megrajzolható a teljes kör. A nyolcad kört az x = -y pontban érjük el.

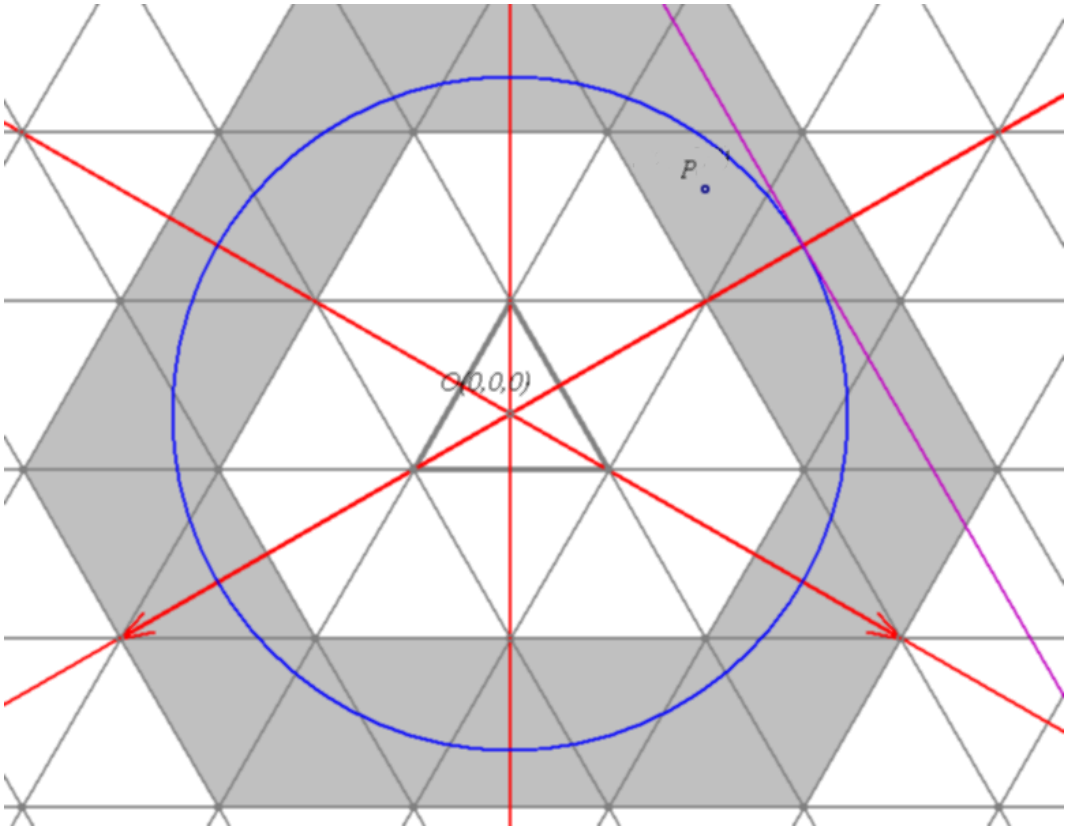
Egy-szomszédos körvonal rajzolása

A vonal rajzoló algoritmushoz hasonlóan definiálhatunk olyan körvonal rajzoló algoritmust, amely egyszomszédos pontsorozatot állít elő.

A választható pontok száma itt is minden esetben kettő. A kér pont közül az egyik mindig belső pontja a körnek.

A fenti algoritmust kiegészítjük egy plusz lépésél, abban az esetben, ha az aktuális pont és a kiválasztott pont nem egyszomszédos. Ilyen esetben a két lehetséges pont közül rajzoljuk azt a pontot is, amely belső pontja a körnek.

Kör rajzolása a háromszögrácson



Az ábrán egy r=2 sugarú kör látható. Mint látható a kis sugarú körök ezen a rács típuson fokozatosan hatszöggé torzulnak.

Az alkalmazott eljárás alap ötlete ugyanaz lesz mint a pixelrácsos párjánál, vagyis vegyük az adott pontban a kör érintőjét, amely segítségével kijelöljük a következő olyan rácspontot amely pontja a körívnek. Minden eseteben origó középpontú kört rajzolunk, melynek pontjait a kirajzolás során el toljuk a rajzolandó kör középpontjának koordinátáival.

Tizenkettes szimmetria elve

Mint a pixelrács esetében itt sem kell a teljes kört megrajzolni, alap esetben elég egy hatod kört rajzolni. Ha viszont felvesszük a tengelyek szögfelező egyeneseit, akkor már látszik, hogy valójában elég egy tizenketted kört megrajzolni, és ezt a kör ív darabot tükrözni.

Körvonal definíciója

Legyen a körvonalunk egy olyan egyszomszédos pontsorozat, melynek minden pontja

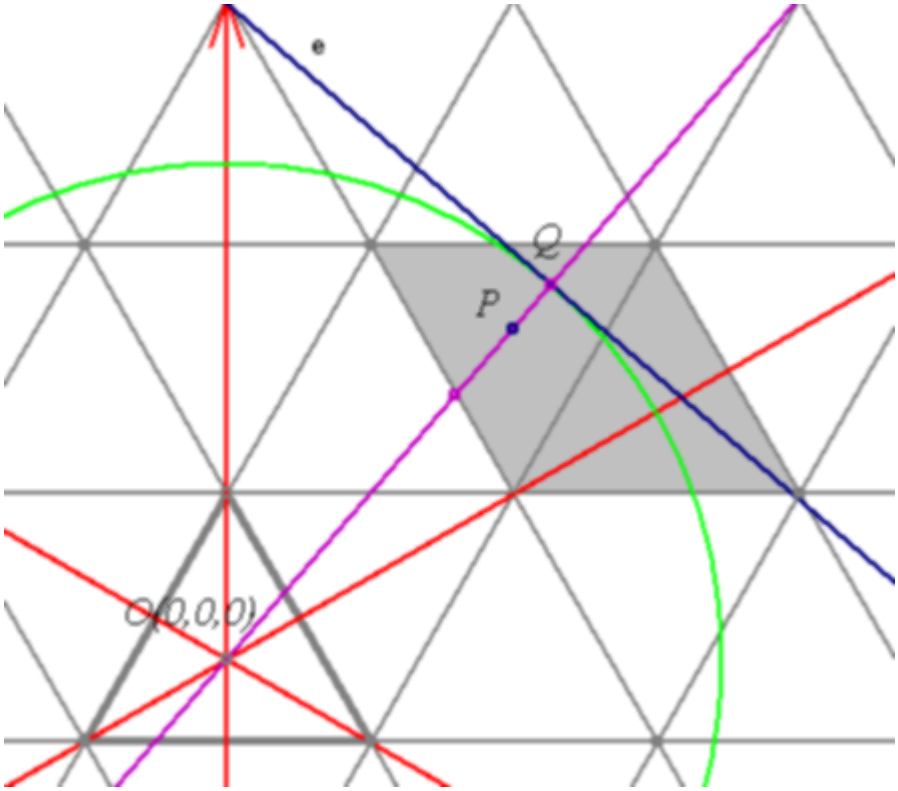
azonos távolságra van egy adott O pontól, és a sorozat első és utolsó tagja is egyszomszédos a rácson.

Ebből a definícióból adódik, egy fontos észrevétel: Tekintsük a feljebb lévő ábrán a P pontot, ennek a P pontnak 3 darab egyszomszédos pontja van. Minden rács pontra amely része egy körvonalnak igaz, hogy a 3 egyszomszédos pontja közül az egyik biztosan nem lehet rajta ugyanazon a köríven. A P pont esetében ez a (1, -1, 0) pont. Tehát valójában itt is 2 darab pont közül kell majd választani minden lépésben.

A módszer rövid leírása

A mindenkori aktuális pontot jelölje: P_a A kör rajzolását valamelyik tengely menti pontból fogjuk indítani, használjuk mondjuk az x tengely. Tehát az első pontunk a (r, -r/2, -r/2) pont lesz. A következő pont kiszámolásához ,meg kell határozni a kör érintő egyenesét ebben a pontban, az egyenes iránya legyen M(x,y,z). A következő pontot az egyenes rajzoló rutinnal fogjuk megkerestetni még pedig, úgy, hogy a most kiszámolt meredekség egyenesnek meghatározzuk a következő pontját a P_a-ból kiindulva. Gyakorlatilag minden pontban meghatározzuk a kör érintőjét és ezen érintő mentén lépünk egyet.

Érintő meghatározása az aktuális pontban



Tegyük fel, hogy az aktuális pont a P. Első lépésben felvesszük azt ez egyenest mely áthalad a körközép pontján és a P is, keressük ennek a segédvonalnak és a körívnek a metszés pontját, amelyet jelöljünk Q-val. A keresett e érintő a segédvonalra merőleges egyenes amely áthalad a Q ponton.

Az OP egyenesnek ismerjük az irányát ez N = P-O-val, mivel origó középpontú kört rajzolunk ezért az O mindig 0, így valójában az N = P-vel. Innen az M-et egy 90° forgatás után kapjuk. Majd m-ből meghatározhatjuk az irány normáját.

$$v_n=(\frac{M_x}{|M|},\frac{M_y}{|M|},\frac{M_z}{|M|})$$

A következő pont meghatározása

A következő pont kiszámolásához meg kell határoznunk az egyenes rajzolásához szükséges adatokat, el kell végezni a vonal rajzolásnál leírt lépéseket, majd ezután az aktuális pontból v_n irányvektor mentén egy pontot kirajzolni.

Ismételjük az előző lépéseket addig amíg el nem érjük az (r/2, -r/2, 0) pontot. Ezen a ponton előállt a tizenketted körívünk, amelyet tükrözve megkapjuk a teljes kört

A teljes algoritmus

Az eljárás egy origó központú r sugarú kört rajzol a háromszögrácson.

$P_a = (r, -\frac{r}{2}, -\frac{r}{2})$

while($P_a \neq (r, -\frac{r}{2}, 0)$)

begin

$M = P_a$ 90 ° *fokkal elforgatva*

s, t értékek maghatározása M-ből.

$v_n = (\frac{M_x}{|M|}, \frac{M_y}{|M|}, \frac{M_z}{|M|})$

if(P_a paritása = s)

begin

P_u = P_a

i = t tengelynek megfelelő koordináta index

$P_{ui} = c_t * v_{ni}$, az i koordináta legyen egyenlő a norma i koordinátájának és az aktuális lépés szám szorzatának kerekített értékével.

If(P_u pontja a rácsnak és P_a és P_u egyszomszédosak)

P_a = P_u

end if

else

begin

P_{u1} = P_{u2} = P_a

i = t tengelynek nem megfelelő egyik koordináta index

j = t tengelynek nem megfelelő egyik koordináta index és $j \neq i$

$P_{ui} = P_{ui} + v_{ni}$

$P_{uj} = P_{uj} + v_{nj}$

if(P_{u1} és P_{u2} is pontja a rácsnak és P_{u1} és P_{u2} is egyszomszédos P_a -val)

begin

$r_{u1} = |v_{ni}|$

$r_{u2} = |v_{nj}|$

if($r_{u1} = 0.0$)

P_a = P_{u1}

else

if($r_{u2}=0.0$) P_a = P_{u2}

else

if($r_{u1}<r_{u2}$) P_a = P_{u1}

else

if($r_{u1}>r_{u2}$) P_a = P_{u2}

else

P_a = P_{u1} , válaszuk ki valamelyik pontot tetszőlegesen, de minden ilyen helyzetben konzekvensen ugyanaz az indexű pontot, legyen ez most az 1-es indexű.

end if

else

begin

if(P_{u1} pontja a rácsnak és P_a és P_{u1} egyszomszédosak)

P_a = P_{u1}

else

if(p P_{u2} pontja a rácsnak és P_a és P_{u2} egyszomszédosak)

P_a = P_{u2}

end else

end else

end while.

Megjegyzések

A bemutatott algoritmus egyenlőre még elméleti szinten van, az implementálása ezen dolgozat írása alatt még folyamatban van.

Mint látható az algoritmus implementálásához szükséges egy nem triviális forgatási művelet az érintő meghatározásához. Főleg ezen művelet kidolgozása miatt nem teljes még az implementáció.

Poligonok kitöltése

A pixelrácson értelmezett bármelyik kitöltési algoritmus alkalmazható kisebb módosításokkal háromszögrácson is.

Az összehasonlíthatóság mindkét rácson értelmezett alakzatoknál használjuk az Euklideszi távolság fogalmat.

Ha adott egy egység négyzetnyi terület mindkét rácson, akkor a következőket mondhatjuk:

- A háromszögrácson az egységnégyzet alatti terülten több lesz a rácspont mint a pixelrácson vet egységnégyzet alatt.
- Emiatt az egység négyzet kitöltése több időt vesz igénybe a háromszögrácson.

Mint látható, ugyanakkora terület kitöltése mindig több erőforrást fog igényelni a háromszögrácson, ezért ebben az összehasonlításban a háromszöges algoritmusok mindig alul maradnak, ám ha hasonló képpont számú alakzatokat veszünk alapul, akkor belátható, hogy az eltérés minimális a két fajta rácson értelmezett módszerek között.

Flood-fill algoritmus

A legegyszerűbb kitöltési algoritmus, a flood-fill, mindkét rácsra ugyanolyan módon alkalmazható.

Az algoritmus bemenő paramétere a rács egy P pontja, illetve egy C szín, valamilyen reprezentációban, ez lesz a kitöltő szín. Az algoritmus működése nagyon egyszerű, ha az aktuálisan vizsgált pont színe megegyezik a P pont színével akkor azt átállítjuk C-re és az aktuális pont összes szomszédját megvizsgáljuk.

```
flood_fill( Pa, Cp, C )
begin
  if( Pa színe == Cp  )
  begin
    Pa színe = C
    flood_fill( Pa bal oldali szomszédja, Cp, C );
    ...
    flood_fill( Pa bal-felső szomszédja, Cp, C );
  end if;
end;
```

Mint látható ez egy tradicionálisan rekurzív algoritmus, ezért az implementáció során ügyelni kell a memória problémákra.

A flood-fill algoritmust nagyon egyszerűen átültethetjük háromszögrácsra is.

Az első lehetőség, hogy használjuk a háromszögrács két koordinátás reprezentációját, így az eredeti algoritmus szomszédos pont képzésén szem kell változtatni.

A másik lehetőség, hogy kihasználjuk a háromszögrács sajátosságait, a szomszédos pont képzésénél egyszerre képezzük az egy-, két- és háromszomszédos pontokat.

A pixelrácson egy pontnak egyszerre 8db szomszédját tudjuk képezni, a háromszögrácson ezzel szemben egyetlen pontnak egy lépésben 12db szomszédját tudjuk előállítani. Így a háromszögrácson alkalmazott algoritmusnak egy lépésben nagyobb a tárigénye. Viszont ha a kitöltendő terület azonos rácspontot tartalmaz, akkor a háromszögrácson értelmezett algoritmus kevesebb lépésben vizsgálja meg az összes pontot, ha nem a két koordinátás reprezentációt használjuk.

Scan-line fill (terület kitöltés)

Az algoritmus részletes leírása megtalálható az [SZK01] jegyzetben, itt a teljes ismertetés nem cél csak a két rács közötti eltérésekből adódó eltérések ismertetése.

Az eredeti algoritmus vízszintes pásztákat használ, a háromszögrácson is használhatjuk ezt a fajta pásztát, azonban a rács specialitása miatt nem mindegy, hogy melyik reprezentációban állítjuk elő a pásztákat. A két koordinátás reprezentációban csak x koordinátában térnek el a pászta végpontjai, viszont a 3 koordinátás reprezentációban csak az y egyezik meg. Tehát célszerű a két koordinátás reprezentáció használata.

A pászta és él metszéspont meghatározáshoz a háromszögrácson a pontok Euklideszi távolság fogalmát kell használni, ami plusz számításokat igényel minden egyes metszés pont kiszámításakor. Ezen a ponton veszítünk az algoritmus végrehajtási idejéből, viszont a több rácspontot kell kitöltenünk, így a az azonos területű kitöltésnél a több számlásért

cserében több pontot is töltünk fel. Az azonos rácspontszámú kitöltésnél az eredeti pixelrácsos algoritmus a hatékonyabb.

A módszer optimalizált változata is átültethető háromszögrácsra is, itt is elég a polygon minimális és maximális y koordinátái között metszés pontokat keresni. Az inkrementális elv is alkalmazható, de itt is a rácspontokat mint Euklideszi pontokat kell kezelnünk, tehát itt is konverzióra van szükség, amely plusz időt igényel.

Tehát a terület kitöltő algoritmus is alkalmazható háromszögrácsra is, de a módszer nem olyan hatékony mint a pixel rácson, viszont a háromszögrácson ugyanakkora Euklideszi terület alatt 1,5x annyi rácspont található, így végül is a módszerek több rácsponton dolgoznak mint a pixelrácson.

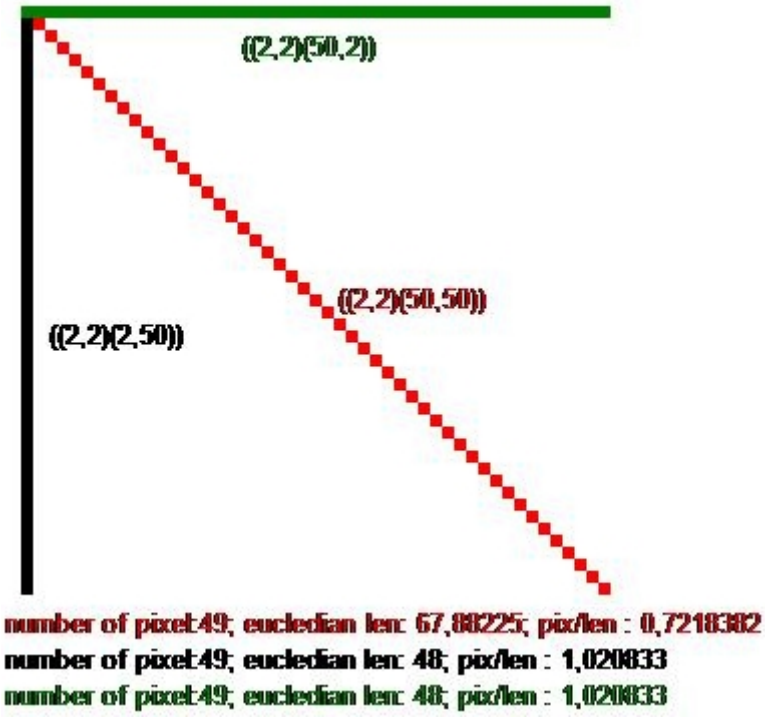
Telítettség vizsgálatok

Vizsgáljuk meg, hogy az egyes szakaszok kirajzolásához mennyi rácspontot kell felhasználni, annak függvényében, hogy mekkora a szakasz Euklideszi hossza.

Legyen adott egy P_0P_1 szakasz, legyen *n* a szakasz kirajzolásához szükséges rácspontok száma, és legyen *l* a szakasz Euklideszi hossza. Nevezzük a szakasz telítettségének az

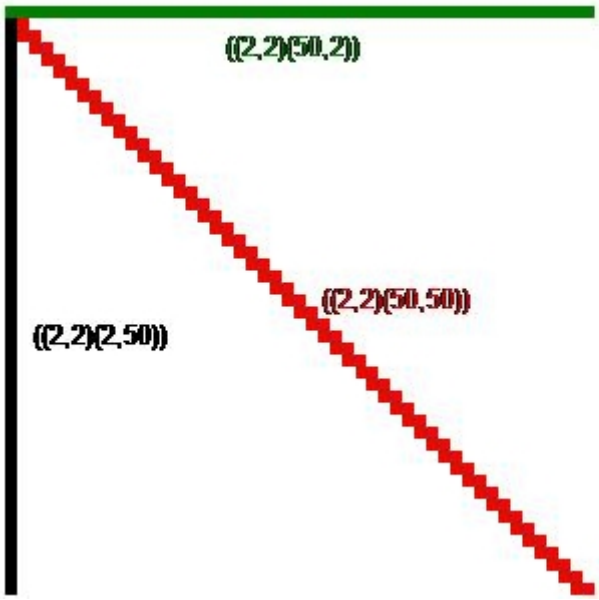
$\frac{n}{l}$ hányadost.

Vegyük először a pixelrácsot vegyük fel egy vízszintes, egy függőleges és végül egy 45° egyenest. Számoljuk ki az egyes vonalak telítettségét.



Az ábrán láthatóak a vizsgálat végeredményei. Megfigyelhető, hogy a tengellyel párhuzamos vonalak telítettsége közel 1, míg a 45° vonalé ettől messze elmarad. Mint látható ezen a rács típuson fenn állhat az úgynevezett Jordan paradoxon, azaz fel tudunk venni 2 olyan egyenest amely geometriailag metszik egymást, még sincs közös pontjuk a rácson.

Az eredmények jelen állapotukban nem összevethetőek a háromszögrácson kapott eredményekkel, mivel a háromszögrácson a vonalakat egyszomszédos pontsorozatnak definiáltuk, ezért a pixelrács tesztjéhez használjuk az egyszomszédos vonalat rajzoló algoritmust.

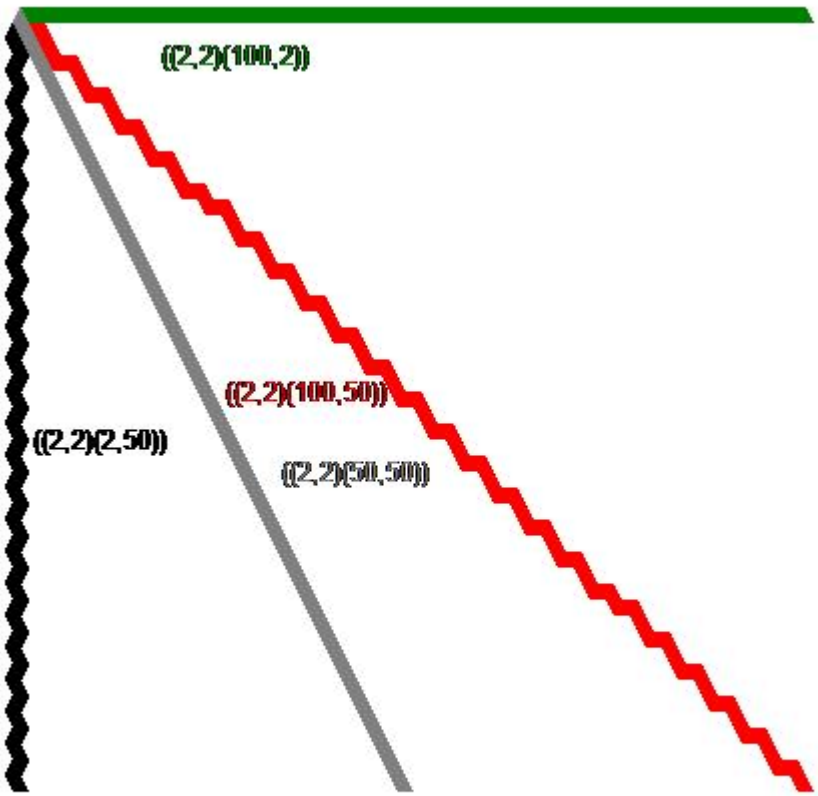


A vízszintes és a függőleges vonalak adatai nem változtak, viszont a 45° vonal adatai a következőképen alakulnak:

$$pixelek\ száma = 97$$
$$eklideszi\ távolság = 67{,}88225$$
$$\frac{pixelek\ száma}{távolság} = \frac{97}{67{,}88225} = 1{,}428944$$

Mint várható volt a 45° vonal telítettsége a duplájára nőtt. A telítettségi összehasonlítást a két rácstípus között ezen adatok alapján fogjuk elvégezni.

Most nézzük meg ugyanezen elhelyezkedésű egyenesek adatait a háromszögrácson. Ebben a vizsgálatban csak az ideális egyenes mentén elhelyezkedő egyszomszédos pontsorozatot tekintjük egyenesnek.



Mint az ábrán látható ezen rács merőben eltérő adatokat mutat. Páronként hasonlítsuk össze az egyes típusú vonalak eredményeit a pixelrácsos vonalak eredményeivel. A vízszintes és függőleges vonalak esetén a telítettség duplája a pixelrácsos megfelelőjénél. A 45° vonal esetén az eltérés az előző adat közel másfélszerese.

A háromszögrácson nem állhat elő az az esete, hogy két szakasz geometriailag metszi egymást, de a rácson nincs közös pontjuk.

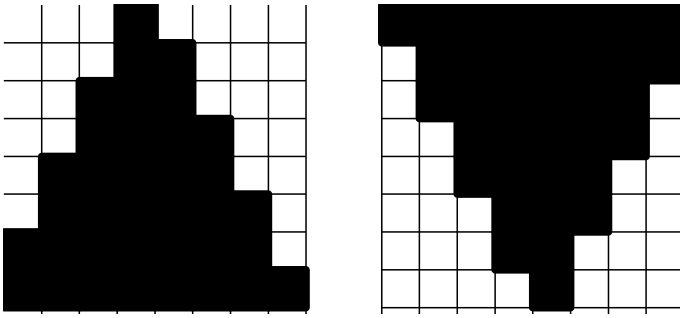
A vizsgálatok alapján állíthatjuk, hogy a háromszögrácson vett szakaszok telítettebbek mint a pixelrácson. A nagyon fontos észrevétel a legkisebb és legnagyobb telítettségek aránya. A pixelrácsnál ez $q \approx \frac{\sqrt{2}}{1{,}020833} \approx 1{,}399$, míg a háromszögrács esetében $q \approx \frac{2.333457}{2{,}020408} \approx 1{,}154$.

Képek tárolása

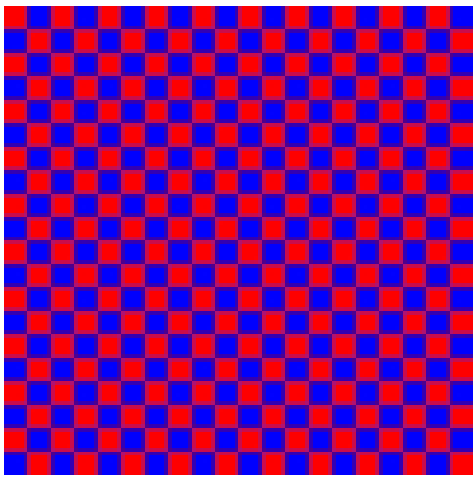
A pixelrácson és a háromszögrácson is tárolhatunk képeket. Az elv mindkét rácstípusnál megegyezik, az eredeti 2 dimenziós kép x és y koordinátáihoz rendeljük hozzá a rács egy pontját, a pontot tartalmazó geometriai formát (egység négyzet, vagy egység háromszög) színezzük olyan színűre mint az eredeti x, y képpont színe.

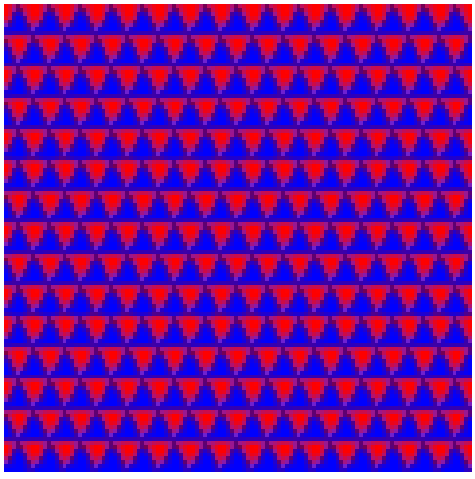
Mivel jelen pillanatban nem áll rendelkezésre olyan eszköz mely háromszögrácsot használna megjelenítésre, ezért a pontosabb összehasonlítás véget emulálni fogjuk mindkét rács típust.

A pixelrácsot 6x6-os négyzetekből aló halóval fogjuk modellezni, így egy eredeti pixelnek 36db pixel fog megfelelni a szimulált rácson. A háromszögrácsot a következő két alakzattal fogjuk modellezni:



A páros és páratlan háromszögek is 36 pixelből állnak, így a két rács egyes pontjainak 'pixelezettsége' megegyezik. Nézzük meg, hogy az egyes szimulált rácsok milyen mintát követve veszik fel a mintát az eredeti képből.





A háromszögrácson való tároláshoz a rács speciális két koordinátás rendszerét fogjuk használni, így mindkét rácson két tengely mentén kell össze rendelnünk a rács pontokat az eredeti képpontokkal.

Induljunk egy nagyon híres veszteség mentes képből, ezen a képen bemutatjuk a különböző tárolási technikákat.

Mivel az alapkép a két rácson tárolva 6x nagyításban fog megjelenni, ezért a további képeket ezen dolgozat csak kicsinyítve mutatja be, a teljes méretű képek megtekinthetők a dolgozat példa képei között.

Tárolás pixelrácson

A szimulált pixelrácson a kép minden egyes p(x, y) pontját a rács r(u,v) pontjának feletetjük meg a következő módon : x = u és y = u

Mint látható a kép megtartja az arányait, minden szín információ ami az eredeti képen megvolt megtalálható az eredmény képen is.

Tárolás háromszögrácson

A háromszögrácson is használjuk a fenti p(x,y) r(u,v) megfeleltetést, akkor eredmény képünk tartalmaz minden szín információt, amit az eredeti is tartalmazott, viszont mint

Látható a kép az arányait nem őrzi meg.

A torzulás a rács specialitásával magyarázható, mivel a háromszög rácson a speciális kétkoordinátás ábrázolásnál a vízszintes tengely mentén sűrűbben helyezkednek el a rácspontok, pontosan 2x sűrűbben.

Ritka tárolás

Előállíthatunk méretarány helyes képet is ha, az eredeti képpontokat a háromszögrács csak páros, vagy csak páratlan pontjainak feletetjük meg.

Az alábbi két képen látható a két eredmény kép a baloldalin a páratlan a jobboldalin a páros háromszögek felhasználásával készült kép látható.

Mint látható a méretarány helyes, mindkét változat tartalmazza az eredeti kép összes szín

információját. Nevezzük ezt a technikát ritka tárolásnak. Ha meg akarjuk különböztetni a ritka tárolás két fajtáját, akkor a nevezhetjük ezeket a felhasznált háromszögek paritása után páros vagy páratlan ritka tárolásnak.

A ritka tárolásnál maradnak olyan rácspontok amiket nem használtunk fel, ezek a rács pontok az eredeti kép szín információján felüli pontok, ezeket használhatjuk plusz információk tárolására, vagy egyszerűen a szomszédos rácspontokból kiszámolható szín

információkat tárolhatunk ezekben a pontokban. Az fenti képen ezeknek a pontoknak a színét a szomszédos pontok színeinek átlagával lettek feltöltve.

A ritka tárolásnak nagyobb a rácspont igénye mint a hasonló méretű pixelrácsos tárolásnak, viszont cserében olyan képet kapunk amelynek méretarány megegyezik az eredeti kép méretarányával, és az eredeti szín információin felül viszont plusz információ tárolására is van lehetőség.

Körvonal vizsgálatok

A képtárolás fejezeteben előállított képeket fogjuk tovább vizsgálni. Egy körvonal kereső algoritmus[EDGE01] segítségével különböző finomsági szinteken körvonalakat fogunk keresni az egyes képeken.

A vizsgálathoz a háromszögrács ritka tárolással előállított képét fogjuk használni, szemben a pixelrács 6x6-os pixelekből előállított képével.

Hasonlóan a képtárolás fejezethez, a hely hiány miatt itt sem a teljes méretű képek szerepelnek, az eredeti képek megtekinthetők a dolgozathoz mellékelt példaképek között.

Pixelrács	**Háromszögrács**	**Finomság**
		1.5

A legdurvább finomsággal készült képeken, már látszik némi eltérés, főleg a vonalak alakja szembetűnő, a pixelrácson szabályosnak tűnő négyzetek jelentek meg, míg a háromszögrácson készült képen szabálytalanabb formák jelentkeznek. Megfigyelhető, hogy a főbb körvonalak mindkét ábrán megvannak, de mintha a háromszögrácson 'sűrűbb' lenne az információ. Szemmel láthatóan több részlet marad meg a háromszögrácsos képből.

A következő próbához növelni fogjuk a finomságok 2.5-re. A várakozások szerint ezen a finomsági szinten már tisztulnia kell a képnek, és a gyengébb körvonalaknak el kell tűnnie a képről.

Pixelrács	Háromszögrács	Finomság
		2.5

Az eredmény szembetűnő, a pixelrácson mintha több részlet maradt volna, viszont a főbb vonalak sokkal gyengébbek, vékonyabbak, mint a háromszögrácsos kép esetében. Ezt a különbséget a kicsinyítés még jobban előtérbe hozta. A képen baloldalon látható oszlop párhuzamos vonala már eléggé kezd felbomlani a pixelrácsos képen, míg a másik képen még egyértelműen felismerhető. Az arc és a kalapon lévő tárgyak körvonalai is jobban kivehetőek a jobb oldali képen.

Végső próbaként emeljük a finomságot a duplájára, és nézzük, hogy mit kapunk.

Pixelrács	Háromszögrács	Finomság
		5.0

Az eredmények ilyen finomság mellett már majdhogynem megegyeznek, a főbb vonalak között elenyészőnek mondható az eltérés, viszont a kicsinyített képek közül inkább a

pixelrácsos kép a felismerhetőbb.

Az összehasonlításban használt háromszögrácsos képen elméletben több információ van, mint a pixelrácsos képen, mivel a ritka tárolással tárolt képet használtuk, ahol a páratlan háromszögeket interpolációval előállítottuk. Az összehasonlítás viszont így is megállja a helyét, mivel mindkét kép az adott rács legjobb tárolási képességét reprezentálja, és mindkét kép ugyanabból a forrás képből lett előállítva, így ezek összehasonlíthatóak.

Az összehasonlítás jelenlegi formáját, helyettesíthetnénk valamilyen matematika formulával, de ezen vizsgálatnak, csak annyi volt a célja, hogy 'ránézésre' bizonyítsuk, hogy a háromszögrácson tárolt kép van olyan jó minőségű, mint a pixelrácsos párja.

Összegzésül elmondható, hogy a háromszögrács ezen a téren is megállja helyét a pixelráccsal szemben.

Összegzés

Mint láttuk a háromszögrács, minden olyan alkalmazásban, ahol most a pixelrács az egyeduralkodó, megállja a helyét, és adott esetben jobb tulajdonságokkal is rendelkezik.

Az összehasonlítás mégsem teljes, és nem is lehet teljes, amíg nincs olyan megjelenítő eszköz, amely háromszögrácsot használ. Csak akkor lehet a két rácsot ténylegesen összehasonlítani, ha mindkettőt a saját rács típusához kapcsolódó megjelenítő eszközök segítségével vizsgáljuk. Illetve a az egyes megjelenítők végtermékét hasonlítjuk össze, ez lenne az igazi összehasonlítás.

Vizsgálhatnák még az egyes algoritmusok számítás és tárigényét, illetve optimalitását, ezen vizsgálatok meghaladják a dolgozat kereteit. Ahhoz, hogy lássuk ezen rács típus alkalmazásnak fő előnyeit, nincs szükség ezekre a számításokra. Az egyik ilyen nagy előny a telítettség, melynek adatai adatok magukért beszélnek. Egy másik izgalmas alkalmazási terület lehet a ritka tárolás, ahol az üres rácspontok segítségével, lehetőség van sokkal jobb minőségű képek előállítására, mint hasonló alkalmazásban a pixelrácson.

A dolgozatban vázolt algoritmusok szinte mindegyike tovább optimalizálható, például az első nagy lépés lenne csak egész aritmetikát használó algoritmusok alkalmazása, illetve a háromszögrács specialitásának jobb kihasználásai.

Mindezek alapján nem is olyan merész a következő kijelentés:

A háromszögrács lehet az alapja a jövő grafikus szabványainak és hardver eszközeinek, mivel nagyon sok előnyös tulajdonságokkal rendelkezik, amelyek nem jellemzőek a pixelrácsra.

Az alkalmazási lehetőségeknek semmi sem szabhat határt.

Irodalomjegyzék

SZK01: Szirmay-Kalos László, Számítógépes Grafika, ,,45
EDGE01: , Techniques in Computational Vision: Advanced Edge DetectionTechniques:The Canny and the Shen-Castan Methods , ,Advanced Edge DetectionTechniques:The Canny and the Shen-Castan Methods,
NAGY: Benedek Nagy, Transformations of the triangular grid, 2005GRAFGEO,ThirdHungarian Conference on Computer Graphics and Geometry, Budapest, Hungary,155-162
GRAF01: Kovács Emőd, Komputer grafika jegyzet, ,,
GRAF02: I. Her, Geometric transformations on the hexagonal grid, 1995,IEEE Transaction on Image Processing,1213-1221
NAGY2: Nagy, B., A symmetric coordinate frame for hexagonal networks, 2004, Proceedings ofthe conference Theoretical Computer Science - Information Society, Ljubljana,Slovenia,,193-196
NAGY3: Nagy, B, Digital geometry of various grids based on neighbourhood structures, ,KEPAF 2007, 6th Conference of Hungarian Association for Image Processing andPattern Recognition, Debrecen, Hungary,46-53
GRAF03: HER, I, A symmetric coordinate frame on the hexagonal grid for computergraphics and vision, 1993,ASME Journal of Mechanical Design 115,447-449