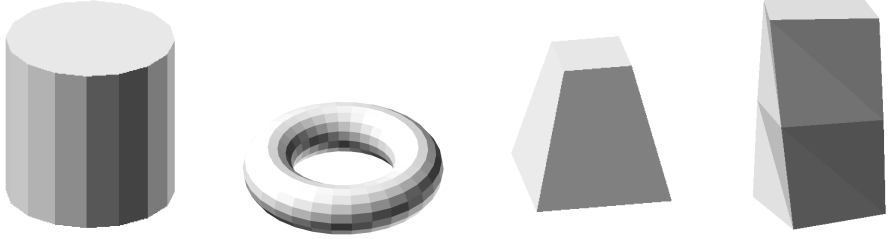
KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

# Modelování
## KMI/3DG

Mgr. Markéta Trnečková, Ph.D.

# Šablonování

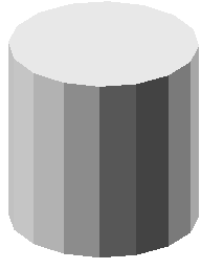
- **Sweeping** – modelovací technika
- Těleso vzniká tažením 2D objektu (**profilu**) po trojrozměrné křivce (**páteř**)
- **Typy šablonování**:
  - Translační šablonování – obrys je libovolný, páteř je úsečka → přímková plocha
  - Rotační šablonování – obrys je libovolný, trajektorie je kružnice (nebo její část) → rotace objektu kolem osy
  - Obecné šablonování – obrys i trajektorie libovolná → zobecněná válcová plocha
- Profilová křivka může v průběhu měnit tvar



# Šablonování

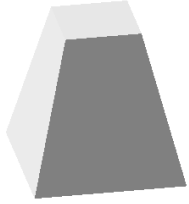
## Translační

- Přímkové plochy – v jednom směru se skládají z úseček
- Pokud profil nemění tvar – plochu získáme z rovinného objektu $P$ vytažením ve směru $\overrightarrow{t}$ o vzdálenost $d = |\overrightarrow{t}|$
- $Q(u, v) = P(u) + v \cdot \overrightarrow{t}$
- Pokud je profil definovaný křivkou (obrys), posuneme řídící body a vykreslíme



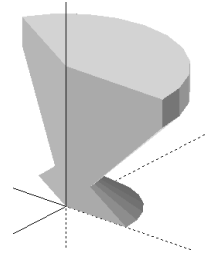
# Šablonování
## Translační

- Přímkové plochy – v jednom směru se skládají z úseček
- Pokud profil mění tvar – křivky definující obrys profilu jsou stejného stupně
- Plochu získáme propojením $P_0$ a $P_1$ přímkovou plochou
- $Q(u, v) = (1 - v) \cdot P_0(u) + v \cdot P_1(u)$



# Šablonování
## Rotační

- Rovinnou plochu (okraj zadaný křivkou) otáčíme kolem osy rotace
- Řezy jsou kružnice



**Příklad**

Jak získáme pomocí rotačního šablonování kouli?

# Modelování pomocí deformací

- Omezená nabídka primitiv, ze kterých tvoříme 3D objekty
- Modelování pouze pomocí plátování – náročné
- Dodatečné tvarování objektů – **deformace**
- Deformace (dle toho jak velkou část tělesa deformace ovlivní):
  - Lokální
  - Globální

# Modelování pomocí deformací

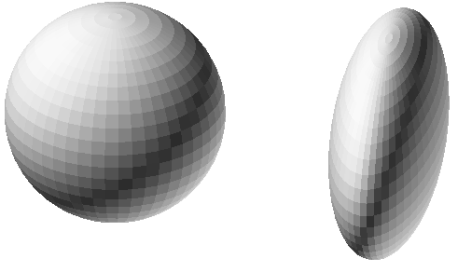
## Barrovy deformace

- Globální
- Nelineární a nelze je (lineárně) skládat s jinými transformacemi
- Nelze je aplikovat na všechny druhy modelů
- Nesmíme narušit topologii modelu
- **Nedeformované těleso**: $[x, y, z]$
- **Deformované těleso**: $[X, Y, Z]$
- **Deformace**: $X = F_x(x),\ Y = F_y(y)\ ,\ Z = F_z(z)$

# Modelování pomocí deformací

## Deformace změnou měřítka

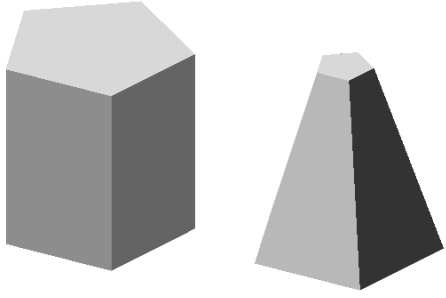
- **Deformace**: $X = s_x \cdot x$, $Y = s_y \cdot y$, $Z = s_z \cdot z$
- $s_x$, $s_y$, $s_z$ – měřítko stlačení/natažení



# Modelování pomocí deformací

## Deformace zeslabování

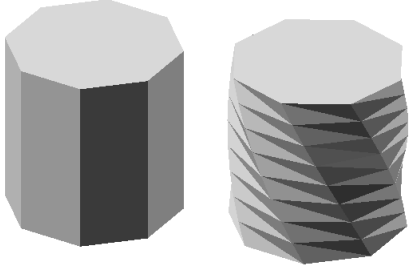
- Deformace zeslabování, zašpičatění, tapering
- Odvozena ze změny měřítka
- Zvolíme osu zeslabení a plynule měníme měřítko za zbývajících osách
- **Deformace ve směru osy z**: $X = r_x \cdot x$, $Y = r_y \cdot y$, $Z = z$
- $r_x = f_1(z)$, $r_y = f_2(z)$ – zeslabovací funkce



# Modelování pomocí deformací

## Deformace zkroucení

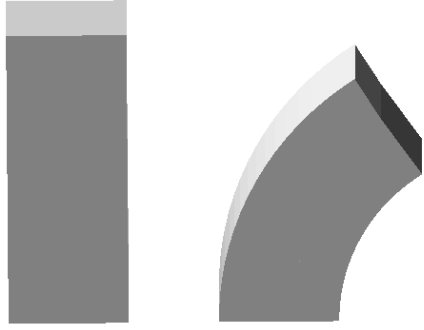
- Deformace zkroucení, twisting
- Zvolíme osu rotace
- **Deformace kolem osy z**: $X = x \cdot \cos(f(z)) - y \cdot \sin(f(z))$, $X = x \cdot \sin(f(z)) + y \cdot \cos(f(z))$, $Z = z$



# Modelování pomocí deformací

## Deformace ohýbání

- Deformace ohýbání, bending
- Složená transformace – ohýbací zóna a vnější zóna tělesa
- Ve vnější zóně – posun a natočení
- V zóně ohybu kruhová deformace

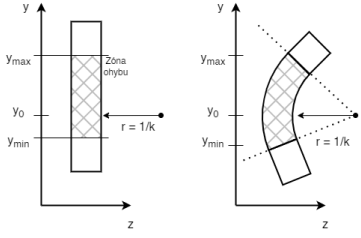


# Modelování pomocí deformací

## Deformace ohýbání

- Ohýbáme kolem osy rovnoběžné s osou $x$
- Parametry ohybu ve směru osy $y$
- Ohýbací zóna $y_{min} \leq y \leq y_{max}$
- Poloměr ohybu $1/k$
- Střed ohybu $y = y_0$
- Úhel ohybu $\theta$
- $\theta = k \cdot (y' - y_0)$

$$y' = \begin{cases} y_{min} & y < y_{min} \\ y & y_{min} \leq y \leq y_{max} \\ y_{max} & y_{max} < y \end{cases}$$



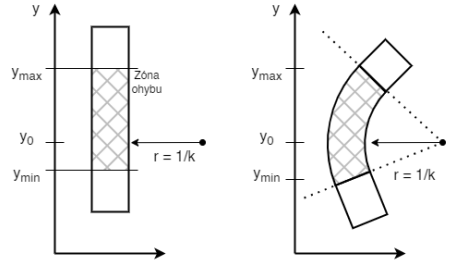
# Modelování pomocí deformací

## Deformace ohýbání

$$X = x$$

$$Y = \begin{cases} -\sin(\theta)\cdot(z-1/k)+y_0+\cos(\theta)(y-y_{min}) & y < y_{min} \\ -\sin(\theta)\cdot(z-1/k)+y_0 & y_{min} \le y \le y_{max} \\ -\sin(\theta)\cdot(z-1/k)+y_0+\cos(\theta)(y-y_{max}) & y_{max} < y \end{cases}$$

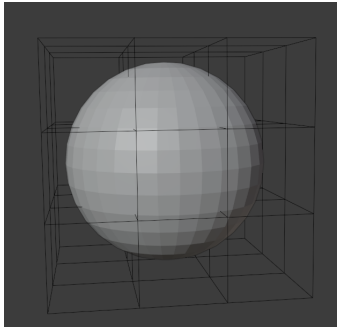
$$Z = \begin{cases} \cos(\theta)\cdot(z-1/k)+1/k+\sin(\theta)(y-y_{min}) & y < y_{min} \\ \cos(\theta)\cdot(z-1/k)+1/k & y_{min} \le y \le y_{max} \\ \cos(\theta)\cdot(z-1/k)+1/k+\sin(\theta)(y-y_{max}) & y_{max} < y \end{cases}$$

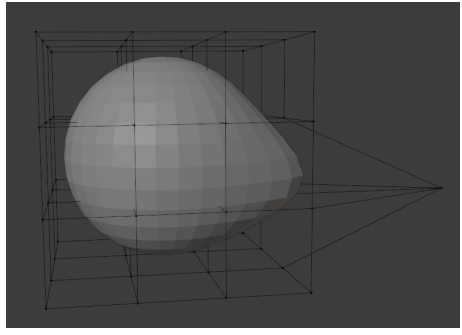


# Modelování pomocí deformací

## Volné tvarování

- Free form deformation (FFD)
- Je možné je aplikovat na různé objekty (nejen tělesa, ale i plochy), v různých reprezentacích
- Představit si to můžeme tak, že objekt vložíme do formy tvaru hranolu, na který poté aplikujeme transformace
- při deformaci hranolu, se deformuje i objekt uvnitř

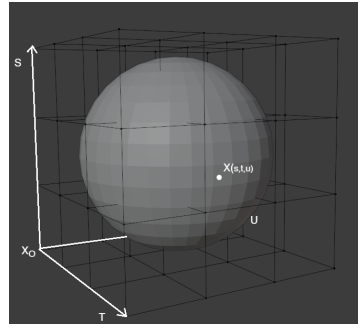




# Modelování pomocí deformací

## Volné tvarování

- Souřadný systém 0, $X$, $Y$, $Z$
- Nový souřadný systém s počátkem $X_0$ – bázi tvoří lineárně nezávislé vektory $S$, $T$, $U$ (nemusí být na sebe kolmé), vymezují rovnoběžnostěn
- Rovnoběžnostěn obsahuje objekt (nebo jeho část)
- Každý bod $X$ se světovými souřadnicemi $[x, y, z]$ má v lokálním systému souřadnice $[s, t, u]$
- $X = X_0 + s \cdot S + t \cdot T + u \cdot U$



# Modelování pomocí deformací

## Volné tvarování

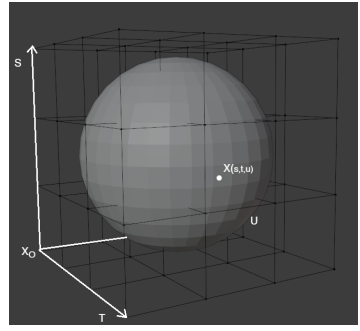
- Před aplikaci známe světové souřadnice bodu $X$ a potřebujeme určit jeho lokální souřadnice v $X_0$, $S$, $T$, $U$

$$s = \frac{T \times U \cdot (X - X_0)}{T \times U \cdot S}$$

$$t = \frac{S \times U \cdot (X - X_0)}{S \times U \cdot T}$$

$$u = \frac{S \times T \cdot (X - X_0)}{S \times T \cdot U}$$

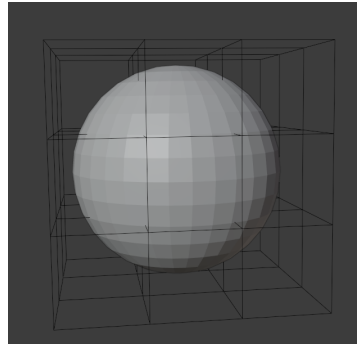
- Pro body uvnitř rovnoběžnostěnu platí $s, t, u \in \langle 0, 1 \rangle$



# Modelování pomocí deformací

## Volné tvarování

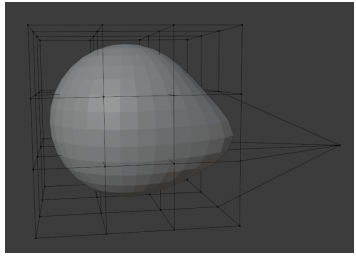
- V prostoru rovnoběžnostěnu vytvoříme pravidelnou prostorovou mřížku řídících bodů $P_{ijk}$ $(i = 0, \ldots, l,\ j = 0, \ldots, m,\ k = 0, \ldots, n)$
- Body vzniknou jako průsečíky rovin ve směrech $S$ ($l + 1$ rovin), $T$ ($m + 1$ rovin) a $U$ ($n + 1$ rovin)
- Mřížka pravidelná (roviny jsou v pravidelném rozestupu)
  $P_{ijk} = X_0 + \frac{i}{l} \cdot S + \frac{j}{m} \cdot T + \frac{k}{n} \cdot U$



# Modelování pomocí deformací

## Volné tvarování

- Rovnoběžnostěn zdeformujeme přemístěním bodů $P_{ijk}$
- Pro každý bod uvnitř rovnoběžnostěnu (lokální souřadnice jsou stále $[s, t, u]$) zjistíme jeho polohu ve světových souřadnicích
- Polohu bodu ovlivňují řídící body $P_{ijk}$ s určitou váhou
- Novou pozici můžeme spočítat například lineární interpolací pomocí nejbližších řídících bodů
- S ohledem na požadovanou spojitost/hladkost je vhodné volit jiné váhy, například Bernsteinovy polynomy



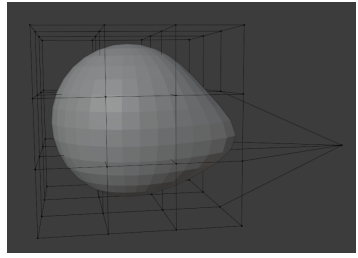
$$X_{ffd}(s,t,u) = \sum_{i=0}^{l} \binom{l}{i}(1-s)^{l-i}s^i \left[\sum_{j=0}^{m} \binom{m}{j}(1-t)^{m-j}t^j \left[\sum_{k=0}^{n} \binom{n}{k}(1-u)^{n-k}u^k \cdot P_{ijk}\right]\right]$$

# Modelování pomocí deformací

## Volné tvarování

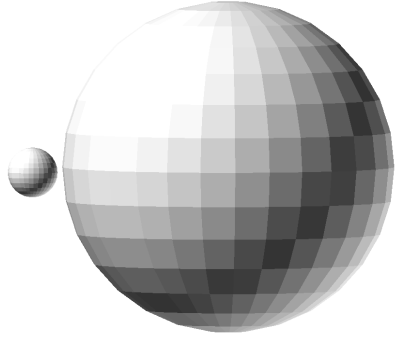
$$X_{ffd}(s,t,u) = \sum_{i=0}^{l} \binom{l}{i}(1-s)^{l-i}s^i \left[\sum_{j=0}^{m} \binom{m}{j}(1-t)^{m-j}t^j \left[\sum_{k=0}^{n} \binom{n}{k}(1-u)^{n-k}u^k \cdot P_{ijk}\right]\right]$$

- $P_{ijk}$ jsou přemístěné body
- $X_{ffd}$ je bod ve světových souřadnicích na lokálních souřadnicích $(s,t,u)$ po deformaci
- FFD zachovává vlastnosti parametrických povrchů



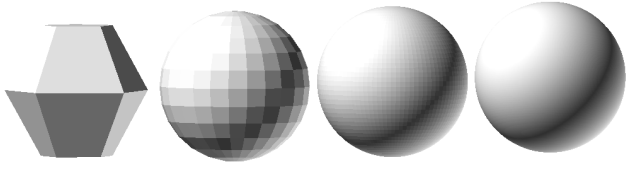
# Dělené povrchy

- Při převodu tělesa na polygonální síť nastává problém s úrovní detailů
- Např. Kouli je snadné definovat matematicky, pomocí polygonů ne
- V určitém momentu je aproximace pomocí nějakého množství trojúhelníků dostatečná
- Při změně měřítka jsou pak ale hrany opět viditelné



# Dělené povrchy

- **Polynomiální síť** = množina vrcholů určující tvar povrchu společně s informacemi o propojení mezi nimi (topologie objektu)
- **Dělení** je definováno jako rekurzivní proces, při kterém zjemňujeme polygonální síť přidáváním nových vrcholů a plošek a tím jí zahlazujeme
- Povrchy vzniklé tímto procesem nazýváme **dělené povrchy**
- Zejména se využívají v aplikacích, kde je kritická velikost dat – například posílání dat (posíláme jen hrubý obrys o vytvoření hladkého se stará příjemce), nebo v počítačových hrách, kde se potřebná úroveň detailu v průběhu mění
- Jdeme-li limitně do nekonečna, dostaneme výsledný povrch, v praxi zastavíme, když je výsledná ploška velká jeden pixel

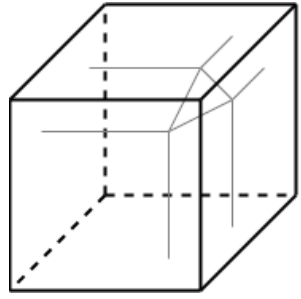


# Dělené povrchy
## Dělící schémata

- **Dělící pravidlo** = předpis, který určité skupině vrcholů přiřadí rozsáhlejší množinu vrcholů
- Zahrnuje pravidla dělení spolu s údaji, jak propojit výsledné vrcholy do nové polygonální sítě
- **Dělení povrchů** = aplikování určitého dělícího schématu na konkrétní polygonální síť
- Při srovnávání dělících schémat se vychází z vlastností, které charakterizují jejich aplikovatelnost a geometrické vlastnosti generovaných povrchů
- **Rozdělení** (podle způsobu vytváření nové sítě):
  - Dělení nad ploškami (face split)
  - Dělení nad vrcholy (vertex split)

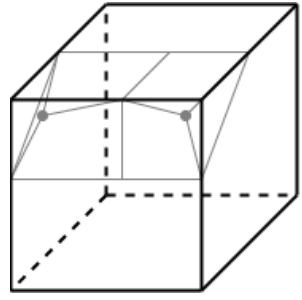
# Dělené povrchy
## Dělení nad vrcholy

- **Vertex split**
- V okolí vrcholů jsou přidány nové vrcholy v rámci každé plošky
- Jejich propojením s dalšími nově vytvořenými vrcholy, které přísluší stejné plošce, vzniknou nové plošky



# Dělené povrchy
## Dělení nad ploškami

- **Face split**
- V okolí plošek vzniknou nové vrcholy, na hranách i mimo ně
- Jejich propojením s dalšími nově vytvořenými vrcholy vzniknou nové plošky
- Aproximační i interpolační
- Při aproximaci neprochází nově vytvořená polygonální síť vrcholy původné sítě a v každém kroku se limitně přibližuje limitě povrchu
- Při interpolaci se vrcholy původní sítě stávají vrcholy nové sítě a jsou i vrcholy limitního povrchu tělesa



# Dělené povrchy

## Požadavky na dělící schéma

- Musí se dát aplikovat na model reprezentovaný polygonální sítí s libovolnou topologií, plošky mohou mít libovolný počet vrcholů
- Možnost změny měřítka – tj. možnost volby úrovně detailu (rekurzivní výpočet toto umožňuje)
- Rychlost a lokální definice – nové vrcholy jsou počítány pomocí několika jednoduchých početních operací, výpočet by neměl záviset na příliš vzdálených vrcholech
- Invariance vůči afinním transformacím

# Dělené povrchy
## Dělící schémata

- **Nad ploškami**: Loop, Catmull-Clark, Butterfly, Kobbelt
- **Nad vrcholy**: Doo-Sabin, Midedge

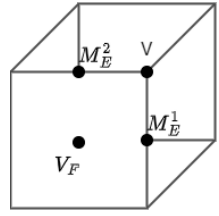
- **Libovolná topologie**: Catmull-Clark, Doo-Sabin, Midedge
- **Síť trojúhelníků**: Loop, Butterfly

- **Aproximační**: Loop, Catmull-Clark, Doo-Sabin
- **Interpolační**: Kobbelt, Butterfly

Další odvozená schémata, do kterých byly přidány další omezující vlastnosti

# Dělené povrchy
## Doo-Sabin

- Aproximační, nad vrcholy, libovolná topologie
- **Ploškový bod** $V_F$ je vypočten jako průměr všech vrcholů $V_i$ plošky (těch je $N$)
  $V_F = \frac{1}{N} \cdot \sum_{i=1}^{N} V_i$
- Pro každý vrchol $V$ vypočítáme nový vrchol $V'$ příslušný k původnímu jako průměr ploškového bodu $V_F$, původního vrcholu $V$ a středů hran $M_E^1$, $M_E^2$, které z něj vychází a jsou hranami zpracovávané plošky
  $V' = \frac{V+V_F+M_E^1+M_E^2}{4}$
- Každý nově vypočítaný vrchol přísluší k původnímu vrcholu, zpracovávané plošce a jejím dvěma hranám sdílejícím původní vrchol
- Vypočítané vrcholy pak propojujeme do výsledné sítě



# Dělené povrchy
## Doo-Sabin

### Typy plošek

- **Ploška v rámci plošky** (face-face) – Vznikne propojením nově vytvořených vrcholů pro danou plošek
- Počet vrcholů této plošky odpovídá počtu vrcholů původní plošky
- **Hranová ploška** (edge-face) – Vznikne propojením nově vytvořených vrcholů příslušných k dané hraně
- Každá tato ploška má 4 vrcholy
- **Vrcholová ploška** – Vznikne propojením nově vytvořených vrcholů příslušných k původnímu vrcholu

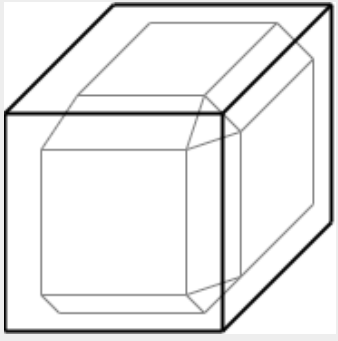
**Příklad**

Kolik vrcholů má nová (vrcholová) ploška?

# Dělené povrchy
## Doo-Sabin

### Příklad

Pojmenujte plošky, které vznikly dělením Doo-Sabin.



# Dělené povrchy
## Doo-Sabin

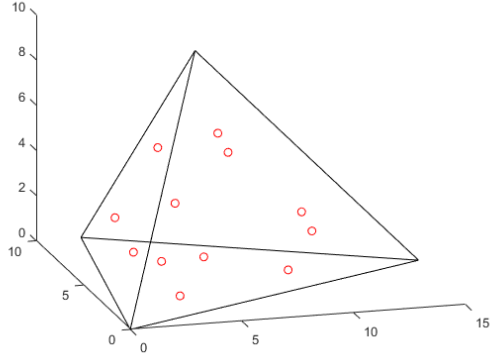
- Počet vrcholů nové vrcholové plošky je roven počtu plošek sdílejících tento vrchol
- Povrchy vzniklé dělením Doo-Sabin splňují $C^1$ spojitost
- Modely, které obsahují otevřený povrch, mohou obsahovat hraniční hrany
- Při dělení jsou nové vrcholy $V_1'$ a $V_2'$ příslušné k hraničním vrcholům $V_1$ a $V_2$ vypočteny dle vztahu
  $V_1' = \frac{1}{4}(V_1 + 3 \cdot V_2)$ a $V_1' = \frac{1}{4}(V_2 + 3 \cdot V_1)$
- Vznikne nová hranová ploška
- Počet hraničních vrcholů zůstává konstantní
- Nově vytvořené vrcholové plošky konvergují k těmto vrcholům

# Dělené povrchy

## Doo-Sabin

### Příklad

Pro čtyřstěn s vrcholy $V_1 = [0, 0, 0]$, $V_2 = [15, 5, 0]$, $V_3 = [2, 10, 0]$ a $V_4 = [5, 5, 10]$ spočítejte souřadnice nových vrcholů, které získáme metodou Doo-Sabin.



# Dělené povrchy

## Midedge

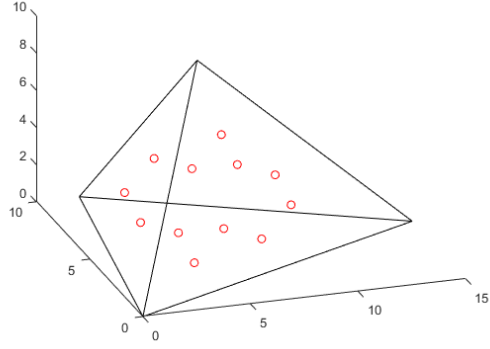
- Algoritmus podobný Doo-Sabin
- Vrcholy se počítají jednodušším způsobem a nezávisejí na všech vrcholech plošky
- Nový vrchol $V'$ vypočítáme jako průměr hran vycházejících z původního vrcholu $V' = \frac{M_E^1}{2} + \frac{M_E^2}{2}$
- Poskytuje pouze $C^1$ spojitost

# Dělené povrchy

## Midedge

**Příklad**

Pro čtyřstěn s vrcholy $V_1 = [0, 0, 0]$, $V_2 = [15, 5, 0]$, $V_3 = [2, 10, 0]$ a $V_4 = [5, 5, 10]$ spočítejte souřadnice nových vrcholů, které získáme metodou Midedge.



# Dělené povrchy
## Catmull-Clark

- Aproximační, dělení nad ploškami
- Pro každou plošku $F$ původní sítě s vrcholy $V_1, \ldots, V_N$ spočítáme **plošková bod** $V_F$ jako průměr všech vrcholů plošky (analogicky jako u Doo-Sabin)
  $V_F = \frac{1}{N} \cdot \sum_{i=1}^{N} V_i$
- Tento bod bude bodem výsledné sítě
- Pro každou hranu $E$ s koncovými body $E_1$ a $E_2$, která sdílí plošky $F_1$ a $F_2$ určíme **hranový bod** (edge point) jako průměr vrcholů hrany a ploškových bodů $V_F^1$ a $V_F^2$
  $V_E = \frac{E_1+E_2+V_F^1+V_F^2}{4}$
- Pro **vrcholový bod** $V_V$ existují různé předpisy, například spočítáme průměr všech ploškových bodů $V_F^i$ okolních plošek sdílejících původní vrchol $V$ a průměr hranových vrcholů $V_E^i$ hran z něj vycházejících
  $V_V = \frac{1}{N} \cdot \left( \frac{1}{N} \sum_{i=0}^{N} V_F^i + \frac{1}{N} \sum_{i=0}^{N} V_E^i + (N-2) \cdot V \right)$

# Dělené povrchy
## Catmull-Clark

- Výpočet je složitější než Doo-Sabin
- Nová ploška vznikne propojením ploškového bodu k oběma hranovým bodům a jejich následným propojením s novým vrcholovým bodem.

**Příklad**

Kolik vrcholů bude mít nová síť?

# Dělené povrchy
## Catmull-Clark

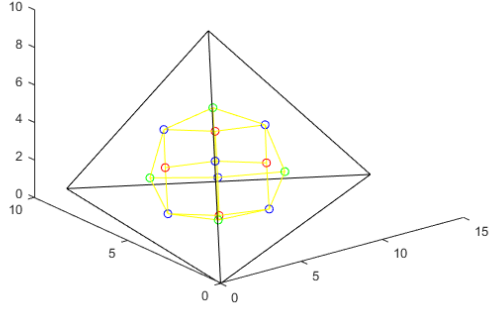
- V rámci původní plošky s vrcholy $V_1, \ldots, V_N$ se vytvoří $N$ nových plošek a každá bude mít 4 vrcholy
- Čtvercová síť se vytvoří hned v prvním kroku dělení a tato topologie pak zůstává zachována
- $C^1$ spojitý povrch (někdy i $C^2$)
- na hranicích povrchu (pokud je těleso otevřené) počítáme hranové body pouze jako průměr vrcholů hrany

# Dělené povrchy

## Catmull-Clark

### Příklad

Pro čtyřstěn s vrcholy $V_1 = [0, 0, 0]$, $V_2 = [15, 5, 0]$, $V_3 = [2, 10, 0]$ a $V_4 = [5, 5, 10]$ spočítejte souřadnice nových vrcholů, které získáme metodou Catmull-Clark.



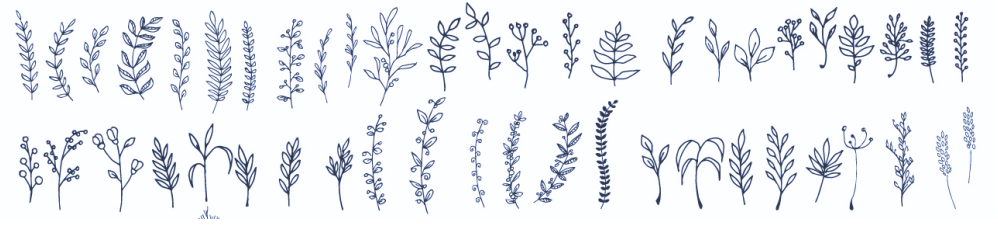
# Pokročilé modelování

- Ne všechny objekty je snadné vytvořit pomocí hraničních ploch, nebo polygonální sítí, která ho aproximuje, upravené deformacemi
- Příkladem jsou velmi členité objekty (například stromy) – jedna větev by byla tvořena mnoha ploškami
- Existuje velká řada metod, které se používají pro komplexní modelování
- **Druhy metod**
  - Procedurální modelování:
    - Vytváříme proces, který dokáže tyto modely vygenerovat
    - Například stromy, rostliny, mraky
    - Místo popisu rostliny pomocí ploch, vytvoříme metodu `růst`, která rostlinu vygeneruje
    - Vzhled rostliny je dán vstupními parametry této funkce
  - Systémy částic
    - Objekt je kolekce částic, které mají nějaké vlastnosti, které se v průběhu času mění
    - Tím se vytváří dynamický systém, který se časem vyvíjí
    - Rostliny, srst, kouř, sníh

# Pokročilé modelování

## Modelování založené na gramatice

- **Motivace – rostliny**
  - Rostliny mají složitou geometrii, která se během času ještě zesložiťuje (např. přibývají další větve)
  - Každá rostlina je jiný, i když se jedná o stejný druh. Má například různě veliké listy, jinou pozici větví a pod.
  - Tato „náhodná“ změna může být přidána do funkce `růst`, což zapřičiní, že stejnou metodou získáme více variant jedné rostliny
  - Každý rostlinný druh je dán různými vlastnostmi – možnosti toho, jak rostlina může vypadat (úhly větví, kde nové větve rostou, kde se objevují květy a pod.)



# Pokročilé modelování

## Modelování založené na gramatice

- **Pravidla gramatiky**: $A \rightarrow B$
- $A$ může být nahrazeno $B$ ($A$ se stane $B$)
- **Dělení** – dle toho jak pravidla vypadají
  - Deterministické bezkontextové
  - Deterministické kontextové
  - Nedeterministické (otevřené) systémy
  - Parametrické
- Většinou jsou tyto systémy **paralelní** – všechna pravidla se aplikují zároveň (rostlina roste ve všech větvích najednou)
- **L-systémy**: 1968 Lindenmayer vytvořil množinu gramatik, která popisuje průběh růstu vláknitých organismů (např. řasy)
- Tato myšlenka byla dále rozvíjena
- Matematický formalismus – paralelní přepisování řetězců (posloupnosti symbolů) dle pravidel
- Po několika iteracích se textový řetězec zobrazuje graficky
- Každý symbol má přiřazen geometrický význam, transformaci, generování objektu či akci

# Modelování založené na gramatice

## Deterministický bezkontextový

- Deterministický bezkontextový L-systém (dL-systém)
- $G = \langle V, P, S \rangle$
- $V$ ... konečná abeceda symbolů ($a$, $b$, ...)
- $P$ ... konečná množina pravidel $A \to B$, $A \in V$, $B \in V^*$
- $S$ ... axiom = neprázdná posloupnost symbolů $S \in V^+$
- **Derivace** – paralelní přepsání všech symbolů $X \in A$ řetězce $V^*$ na pravé strany pravidel z množiny $P$
- Každou iterací (derivací) vzniká nová generace rostliny
- Některé systémy jsou rozšířeny o **zásobník** (množina symbolů je obohacena o znaky [ a ])
- [ uložení stavu na zásobník
- ] vyzvednutí stavu ze zásobníku
- Na tomto systému byla založena Želví grafika

# Modelování založené na gramatice

## dL-systém

**Příklad**

Máme následující pravidla
$0 \to 1[0]1[0]0$,
$1 \to 11$,
$[\to [$,
$] \to ]$.
Začneme s řetězcem 0, jak budou vypadat následující dvě iterace?

# Modelování založené na gramatice

## dL-systém

**Příklad**

Máme následující pravidla
$0 \to 1[0]1[0]0$,
$1 \to 11$,
$[\to [$,
$] \to ]$.
Začneme s řetězcem 0, jak budou vypadat následující dvě iterace?

1. 0
2. 1[0]1[0]0
3. 11[1[0]1[0]0]11[1[0]1[0]0]1[0]1[0]0

# Modelování založené na gramatice

## dL-systém

- **Grafická reprezentace**
  - 0 a 1 vykreslete čáru (zadané délky)
  - [ současná pozice a úhel je zapamatován a kreslení pokračuje pod úhlem 45° (směr se může v průběhu střídat $+/-45°$
  - ] kreslení se posune zpět do poslední zaznamenané pozice a směru

# Modelování založené na gramatice

## dL-systém

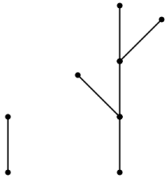
**Příklad**

Zakreslete předchozí příklad graficky.

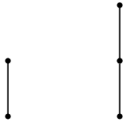
# Modelování založené na gramatice

## dL-systém

Pravidlo $0 \rightarrow 1[0]1[0]0$



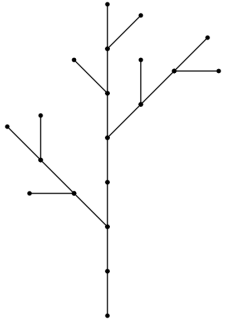
$1 \rightarrow 11$



# Modelování založené na gramatice

## dL-systém

3. krok iterace



# Modelování založené na gramatice

## dL-systém

**Příklad**

Definujte vlastní gramatiku. Podívejte se na výstupy několika kroků.

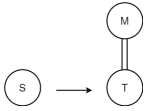
# Modelování založené na gramatice

## dL-systém

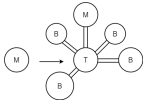
- McConnell představil vlastní gramatiku
- Pravidla nereprezentoval jako řetězce, ale pomocí 3D grafů – přímočařejší postup při vykreslování

# Modelování založené na gramatice

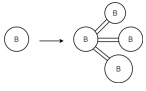
## dL-systém




Pravidlo, kde se ze semínka (seed) stane kmen (trunk) a meristém (dělivé plertivo), nebo také bod růstu




Pravidlo, kde se ze meristém stává kmenem a 4 větvemi (brunch) na něm




Pravidlo, kde se dále rozrůstají větve

# Modelování založené na gramatice

## dL-systém

**Příklad**

Jak bude vypadat výsledek po 3 iteracích?

# Modelování založené na gramatice

## Stochastické systémy

- Nedeterministické L-systémy
- V množině pravidel $P$ je možné mít více pravidel se stejnou levou stranou
- Pravidla:
  $A \rightarrow B : pp$
- $pp$ je pravděpodobnost, že symbol bude přepsán právě tímto pravidlem
- Součet pravděpodobností pro stejnou levou stranu musí být 1

# Modelování založené na gramatice

## Stochastický systém

**Příklad**

Vymyslete nějaký jednoduchý stochastický systém a simulujte několik iterací. Například pomocí hodu kostky.

# Modelování založené na gramatice

## Kontextový L-systém

- Pravidla jsou rozšířena o kontext
- Pravidla:
  $lc\langle A\rangle rc \rightarrow B$
- $lc, rc \in V^*$
- Vymyslete nějaký příklad kontextového pravidla

# Modelování založené na gramatice

## Parametrický L-systém

- Pracuje s parametrickými slovy $A()$
- V () jsou zadané parametry, může být prázdná, musí být konečná
- Díky parametrům můžeme obohatit pravidla o logické či aritmetické výrazy
- $G = \langle V, \Sigma, \omega, P \rangle$
- $V$ ... abeceda
- $\Sigma$ ... množina formálních parametrů
- $P$ ... konečná množina pravidel
- $\omega \in (V \times R^*)^+$ neprázdné parametrické slovo – axiom

# Modelování založené na gramatice

## Parametrický L-systém

- Pravidla:
  $id : lc\langle A\rangle rc : cond \rightarrow B : prob$
- $id$ . . . číslo pravidla
- $lc$, $rc$ . . . kontexty
- $cond$ . . . logický výraz
- $prob$ . . . pravděpodobnost s jakou bude pravidlo aplikováno v případě více pravidel se shodnou levou stranou

# Modelování založené na gramatice

## Parametrický L-systém

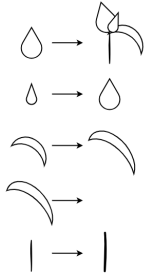
- Příklad – fazole
- Každá část uchovává informaci o svém stáří
- Pravidla mají podmínky, za kterých se aplikují a alternativní větve, co se má udělat, pokud podmínka splněná není
- Tím můžeme modelovat zpoždění vývoje některých částí
- části můžeme i odstraňovat

# Modelování založené na gramatice

## Parametrický L-systém

### Pravidla

- $A(0)$ . . . axiom
- $A(d) : d \leq 0 \rightarrow I(1)[-(10)L(1)][-(35)A(1)] + (10)/(180)A(0)$
- $A(d) : * \rightarrow A(d-1)$
- $L(d) : d == 2 \rightarrow \%$
- $L(d) : * \rightarrow L(d+1)$
- $I(d) \rightarrow I(d+1)$



**Pozn.**

$I$ . . . větev, $L$ . . . list, $A(0)$ . . . vrchol, $A(1)$ sekundární pupen
Před písmeny jsou úhly, pod kterými části přidáváme, v () je věk
$*$ značí alternativní větev, % list odstraníme (odpadne)

# Modelování založené na gramatice

## Parametrický L-systém

**Příklad**

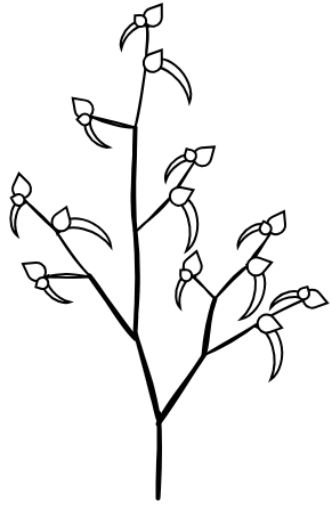
Simulujte několik prvních kroků růstu fazole.

# Modelování založené na gramatice

## Parametrický L-systém

**Příklad**

Simulujte několik prvních kroků růstu fazole.



# Modelování založené na gramatice

## Otevřené systémy

- Nedeterministické, kontextové, parametrické L-systémy
- Otevřenost = možnost interagovat s prostředím (oběma směry, směrem dovnitř například informace o překážkách, množství dopadajícího světla)
- Rozšíříme o **komunikační kanály**
  $?E(x_1, \ldots, x_n)$
- Před přepsáním pravidel se provede mezikrok, ve kterém se získávají hodnoty parametrů komunikačních modulů

# Fraktální geometrie

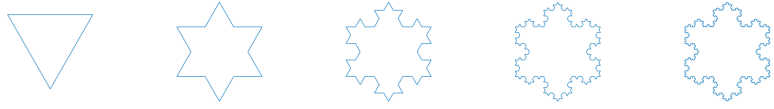
- Modelování povrchů s drobnými variacemi (detaily) je složité
- Uměle vytvořené objekty se vyznačují přesností, objekty v přírodě ne
- **Fraktální geometrie** je nástroj, který umožňuje popsat složité objekty pomocí algoritmů
- **Soběpodobnost** – invariance vůči změně měřítka
- Soběpodobnou strukturu je možné rozložit na struktury z nichž každá je zmenšenou kopií originálu
- Např. čtverec je možné složit z menších čtverců, úsečku z menších úseček
  - Přesná soběpodobnost
  - Statistická soběpodobnost
- Soběpodobnost je nutná, ale ne postačující podmínka fraktálního charakteru objektu

# Fraktální geometrie

## Přesná soběpodobnost

- $A$ je přesně soběpodobná, pokud je sjednocením konečného počtu transformovaných kopií sebe sama
  $A = \bigcup_{i=1}^{n} \varphi_i(A)$
- $\varphi_i$ jsou transformace rotace, posunutí a každá z nich je zároveň změnou měřítka s koeficientem $s_i \in \langle 0, 1 \rangle$
- Součet koeficientů musí být v intervalu $\langle 0, 1 \rangle$
- Pokud je $0 < \sum_{i=1}^{n} s_i < 1$ mluvíme o **průměrné kontraktivitě**

## Kochova vločka



# Fraktální geometrie

## Statistická soběpodobnost

- $A$ je statisticky soběpodobná, pokud je sjednocením konečného počtu zmenšených kopií sebe sama
  $A = \bigcup_{i=1}^{n} \varphi_i(A)$
- Každá kopie $\varphi_i(A)$ má stejné statistické charakteristiky jako $A$
- $\varphi_i$ jsou statisticky nerozlišitelné
- Dle toho, zda je $\varphi_i$ lineární nebo nelineární, mluvíme o lineární nebo nelineární soběpodobné množině
- Příklad: Kámen a hora (rozlišení na obrázku je složité)

# Fraktální geometrie

## Soběpodobnost

- Myšlenka soběpodobnosti – zoomováním objektu v něm najdeme stejné vzory, které se opakují s jiným rozlišením

# Fraktální geometrie

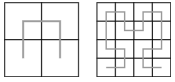
## Dimenze fraktálu

- **Topologická dimenze**:
  - bod . . . 0
  - úsečka . . . 1
  - čtverec . . . 2
  - krychle . . . 3
- Měření fraktálů je v topologické dimenzi složité
- Čím je povrch členitější, tím je měření nepřesnější
- Je-li $\epsilon$ míra, velikost objektu získáme
  $K = N \cdot \epsilon$
  Potřebujeme $N$ úseček délky $\epsilon$. $K$ je aproximovaná délka.
- Čím je $\epsilon$ menší, tím se zvětšuje hodnota $K$
- Pro $\epsilon$ blížící se 0 se $K$ blíží k $\infty$
- Aby mělo $K$ smysluplnou hodnotu, je potřeba použít upravený vztah
  $K = N \cdot \epsilon^D$
- $D$ je **fraktální dimenze**
- Existuje více definic fraktální dimenze, tato je nejpoužívanější – **soběpodobnostní FD**

# Fraktální geometrie

## Dimenze fraktálu

- Pro ne příliš členité povrchy je fraktální dimenze rovna topologické
- Pro úsečku délky $K = 1$ potřebujeme $N = 3$ úseček délky $1/3$. Abychom dostali odpovídající míru, musí být $D = 1$. Kdyby bylo $D > 1$ $K$ by byla nekonečná, pro $D < 1$ je $K$ rovno 0.
- Intuitivně křivky mají topologickou dimenzi 1
- **Hilbertova křivka** (space filling) je křivka, která má fraktální dimenzi 2



- Křivky, které se neprotínají nemohou mít dimenzi větší než 2
- Spojité křivky nemohou mít dimenzi menší než 1

# Fraktální geometrie
## Dimenze fraktálu

- U členitých objektů spíše než topologickou dimenzi měříme míru členitosti (fraktální dimenzi)
- Čím je objekt členitější, tím je fraktální dimenze větší
- Pokud má objekt topologickou dimenzi 1 a fraktální dimenzi 2, znamená to, že vyplňuje plochu.
- Vysoce členité objekty mají fraktální dimenzi větší než topologickou
- **Fraktály** – vysoce členité objekty, které splňují soběpodobnost

# Fraktální geometrie

## Fraktál

- Fraktál:
  $A = \bigcup_{i=0}^{\infty} \varphi_i(A)$
- V přírodě nejsou objekty s nekonečně detaily (nejedná se o fraktál, této definice)
- Lepší označení by bylo **fraktálům podobné struktury**
- **Přesně soběpodobné fraktály** = deterministické
- **Statisticky soběpodobné fraktály** = nedeterministické (stochastické, náhodné)
- Podle transformace dělíme na lineární a nelineární
- V počítačové grafice se setkáme nejčastěji se statistickými lineárními

# Fraktální geometrie

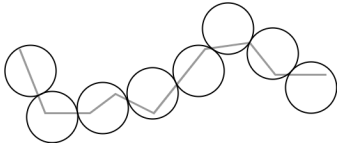
## Výpočet dimenze

- $D$ určíme z $K = N \cdot \epsilon^D$ tak, že za $K$ dosadíme 1 a $\epsilon$ budeme limitně zmenšovat ($\rightarrow 0$)
- Fraktální dimenzi $D$ deterministického fraktálu, který vznikne aplikací $n$ transformací $\varphi_i$, každá s měřítkem $s_i$ (koeficient měřítka je $s$) dle následující rovnice $\sum_{i=0}^{n} s_i^D = 1$
- Tedy $D = \frac{\log n}{\log 1/s}$
- Odhad stochastických fraktálů – různé metody založené na **metodě počítání čtverců** (box counting method)

# Fraktální geometrie

## Výpočet dimenze

- Máme křivku, jejíž dimenzi odhadujeme
- V prvním kroku ji pokryjeme $N_1$ nepřekrývajícími se kruhy stejné velikosti



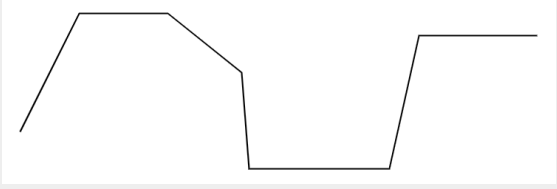
- Ve druhém kroku $N_2$ menšími kruhy stejné velikosti

- Poměr velikostí kruhů z obou kroků označíme $r$
- Odhad fraktální dimenze je
  $D \approx \frac{\log N_2/N_1}{\log 1/r}$
- Přesnější odhad získáme pomocí více měření

# Fraktální geometrie
## Výpočet dimenze

### Příklad

Určete odhad fraktální dimenze následující křivky.



Žlutá kolečka 20 mm, modrá 15 mm, světle zelená 10 mm, zelená 5 mm

# Fraktální geometrie

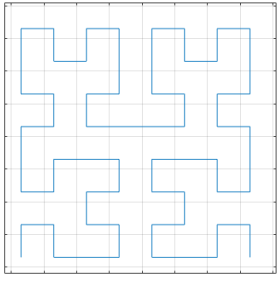
## Výpočet dimenze

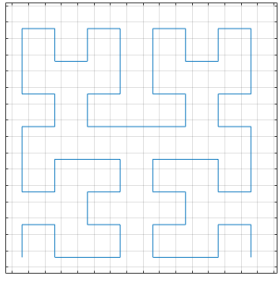
- Jednodušší způsob – pokryjeme křivku sítí čtverců $s_0 \times s_0$
- Spočítáme počet čtverců, které křivku obsahují – $n_0$
- V dalším kroku zvětšíme rozlišení sítě na $s_1 \times s_1$ a spočítáme $n_1$
- Měření můžeme několikrát opakovat
- Hodnoty naneseme na logaritmicko - logaritmickou stupnici, body proložíme přímkou (aproximujeme je)
- Sklon této přímky určuje dimenzi měřené křivky
- Při dvojnásobném zvětšení kroku se dimenze odhaduje $D = \log_2(n_1/n_0)$

# Fraktální geometrie
## Výpočet dimenze

### Příklad

Určete odhad fraktální dimenze následující křivky.





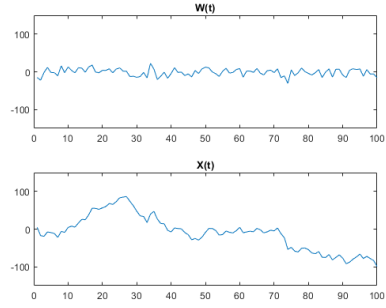
# Fraktální geometrie
## Náhodné fraktály

- Stochastické fraktály, založené na náhodných číslech
- V přírodě jsou transformace nelineární, my se spokojíme s lineárními
- **Brownův pohyb** (Brown motion, Bm) – základ náhodných fraktálů
- Brown pozoroval pohyb pylových částí ve vodě, Einstein později pohyb zdůvodnil – molekuly naráží na zrnka pylu
- Brownův pohyb byl nazván **náhodná procházka** (random walk)

# Fraktální geometrie

## Náhodné fraktály

- **1 rozměrný Brownův pohyb** (Brown motion, Bm)
- $X$ se v čase $t$ nachází v pozici $X(t)$
- $X$ se pohybuje rovnoměrnou rychlostí po ose $x$
- V diskrétních okamžicích $i = 0, 1, \ldots$ je vychýlen v kolmém směru náhodnými pulsy $W(t)$
- $W(t)$ je funkce pulzů – pulzy musí mít Gaussovské rozložení a jejich střední hodnota musí být rovna 0 (tato podmínka charakterizuje energii molekul)
- Bm má fraktální dimenzi 1.5
- Body získané průnikem Bm křivky s osou $x$ tvoří fraktální množinu s dimenzí 0.5



# Fraktální geometrie

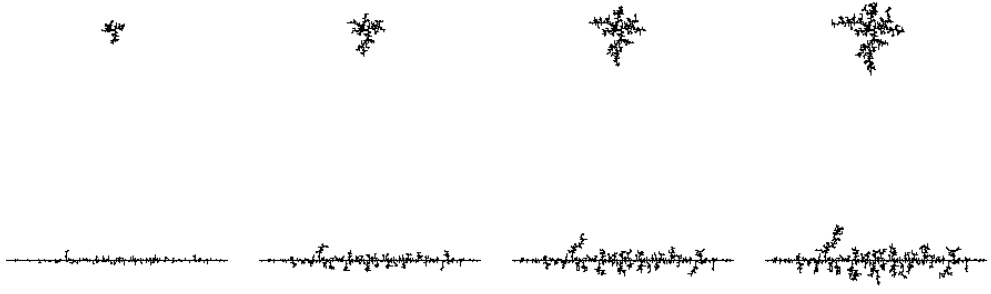
## Náhodné fraktály

- Přímou aplikací 1 rozměrného Brownova pohybu je **Difúzní omezená agregace** (Diffusion limited aggregation, DLA)
- DLA je fyzikálním jevem, při kterém vznikají dendrity, např. při elektrických výbojích
- Rostou tak obrazce na zmrzlých sklech
- Simulace růstu korálů
- **Princip**:
  - V roztoku plavou molekuly látky
  - V roztoku jsou také kondenzační jádra
  - Když se nějaká z pohybujících se molekul přiblíží ke kondenzačnímu jádru, je jím zachycena, nalepí se na něj a stane se také kondenzačním jádrem
  - Jak přibývají jádra, vzniká tvar
  - Ve 2D vzniká fraktál s dimenzí 1.7
  - V jiných variantách se molekula stane kondenzačním jádrem až po několika dotycích, nebo jen s nějakou pravděpodobností

# Fraktální geometrie

## DLA

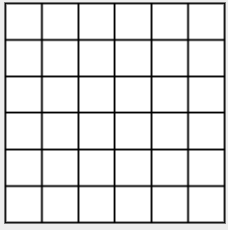
- **Algoritmus ve 2D**:
  1. Vytvoříme nulovou matici
  2. Některé hodnoty nastavíme na nenulovou hodnotu (kondenzační jádra)
  3. Postupně v cyklu přidáme na náhodné místo na okraji částici, na kterou aplikujeme Bm
  4. Pokud se při pohybu částice dotkne jádra, stane se také jádrem
  5. Pokud částice opustí oblast matice, vygeneruje se další částice
- Simulace je pomalá, algoritmus má exponenciální složitost



# Fraktální geometrie
## DLA

### Příklad

Do následující matice vložte jedno nebo více kondenzačních jader. Pomocí kostky simulujte DLA.

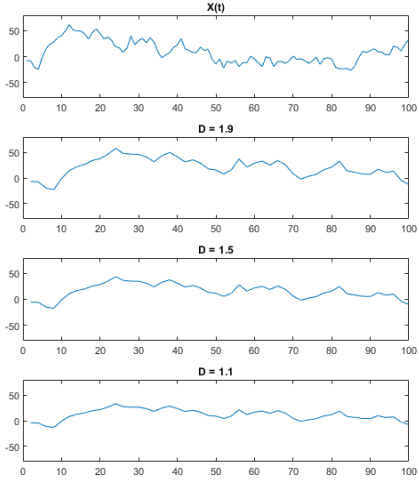


# Fraktální geometrie
## Zobecněný Bm

- Zobecnění Bm – **Zlomkový Bm** (fBm)
- Vznikají fraktály s libovolnou fraktální dimenzí
- fBm získáme z Bm změnou měřítka (s různým koeficientem v různém směru)
- Náhodné fraktály s dimenzí $D$ získáme z Bm tak, že ho vynásobíme ve směru $x$ koeficientem $r$ a ve směru $y$ koeficientem $1/r^H$
- $H$ je **Hurstův exponent**
  $D = 2 - H$ pro $0 \leq H \leq 1$
- **Postup**:
  - Vygenerujeme Bm
  - Zvolíme fraktální dimenzi $D$
  - Určíme Hurstův exponent $H$ ($H = 2 - D$)
  - Bm vynásobíme ve směru $x$ hodnotou $r$ (nejčastěji se volí 2) a ve směru $y$ koeficientem $1/r^H$

# Fraktální geometrie

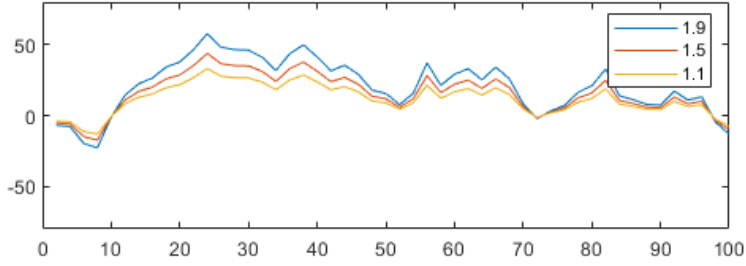
## Zobecněný Bm



# Fraktální geometrie

## Zobecněný Bm

- $H < 0.5$ – křivka je více členitá ($D > 1.5$)
- $H > 0.5$ – křivka je méně členitá ($D < 1.5$)
- $H = 0.5$ – křivka je statisticky nerozlišitelná od Bm ($D = 1.5$)



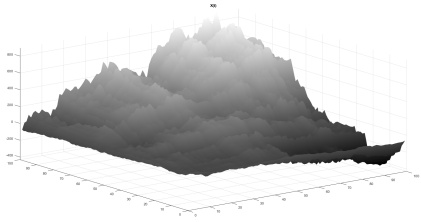
# Fraktální geometrie
## Zobecněný Bm

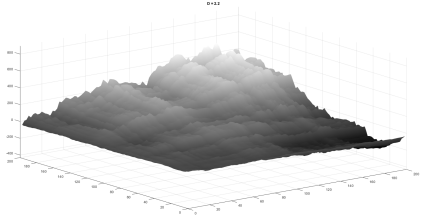
- **Rozšíření do 2D** – zlomková Brownova plocha (fBm plocha)
- Fraktální dimenze $D = 3 - H$ ($D \in \langle 2, 3 \rangle$)
- $H$ je **Hurstův exponent** $0 \leq H \leq 1$
- Aby byla zachována statistická soběpodobnost, musí mít plocha stejné statistické momenty (obecně stačí schodnost průměru a směrodatné odchylky) v každých dvou intervalech a v obou směrech

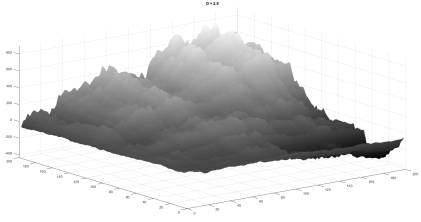
# Fraktální geometrie

## Zobecněný Bm

**Terén pomocí fBm plochy** – $H$ určuje členitost povrchu (čím je vyšší, tím je hladší)



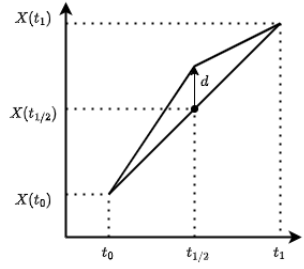




# Fraktální geometrie

## Metoda náhodného přesouvání středu

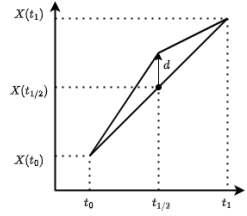
- Alternativní metoda pro modelování fBm je metoda **náhodného přesouvání středního bodu** (random midpoint displacement)
- V 1D – rekurzivně dělíme úsečku ma polovinu a přesouváme střední bod o náhodné číslo $d$
- Úsečku nahrazujeme lomenou čarou
- Tato křivka interpoluje 2 body – **fraktální interpolace**



# Fraktální geometrie

## Metoda náhodného přesouvání středu

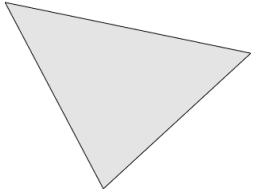
- Vstup: úsečka $[t_0, X(t_0)]$, $[t_1, X(t_1)]$ a $H$
- Výstup: fraktální křivka interpolující body $[t_0, X(t_0)]$ a $[t_1, X(t_1)]$, dimenze fraktálu je $D = 2 - H$
- Určíme střed úsečky $[t_{1/2}, X(t_{1/2})] = \left[\frac{t_0+t_1}{2}, \frac{X(t_0)+X(t_1)}{2} + d\right]$
- Stejný postup aplikujeme na nově vzniklé úsečky
- $d$ musí být v normalizovaném Gaussově rozložení s tím, že se v každé iteraci mění rozptyl v závislosti na H
- V praxi se výpočet $d$ zjednodušuje následujícím vztahem $d = H \cdot gauss(0, 1) \cdot \Delta t$
- Kritérium zastavení rekurze závisí na aplikaci, např. dle délky úsečky (úsečka je menší než uhlopříčka pixelu), na počtu úseček



# Fraktální geometrie

## Metoda náhodného přesouvání středu

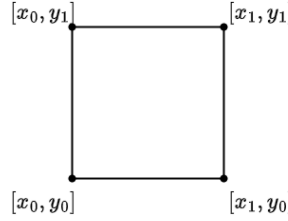
- V počítačové grafice se používá například pro generování tvaru ostrovů (kontinentů)
- Fraktální interpolace se použije na několik bodů, kterými určíme přibližný obrys



# Fraktální geometrie

## Metoda náhodného přesouvání středu

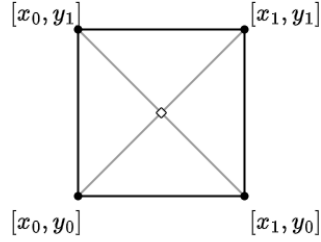
- Ve 3D aplikujeme tuto metodu buď na trojúhelníky, nebo na čtverce
- Máme 4 body:
  $[x_0, y_0, f(x_0, y_0)]$, $[x_0, y_1, f(x_0, y_1)]$,
  $[x_1, y_0, f(x_1, y_0)]$, $[x_1, y_1, f(x_1, y_1)]$
- Na obrázku je naznačen průmět do roviny $XY$



# Fraktální geometrie

## Metoda náhodného přesouvání středu

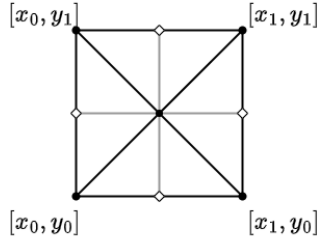
- Při rekurzivním dělení zjistíme střed čtverce $[x_{1/2}, y_{1/2}, f(x_{1/2}, y_{1/2})]$
  $x_{1/2} = \frac{x_0+x_1}{2}$, $y_{1/2} = \frac{y_0+y_1}{2}$
  $f(x_{1/2}, y_{1/2}) =$
  $\frac{f(x_0,y_0)+f(x_1,y_0)+f(x_0,y_1)+f(x_1,y_1)}{4} + d$
- Stejně jako ve 2D, $H = 3 - D$



# Fraktální geometrie

## Metoda náhodného přesouvání středu

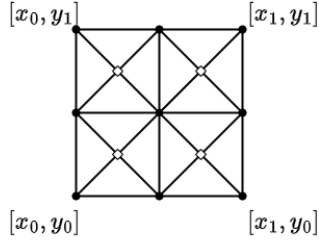
- Ve 2. kroku se vypočítají body, které leží na hranách zadaného čtverce
- Toho dosáhneme tak, že čtverec otočíme o 45ř a vypočítáme hodnoty stejným způsobem, jako v předchozím kroku
- Na okraji má každý čtverec jen 3 sousedy – hodnotu vypočítáme pouze z nich (implementačně doplníme okraje o 0 hodnoty – zero padding)
- Vzniklá chyba není patrná



# Fraktální geometrie

## Metoda náhodného přesouvání středu

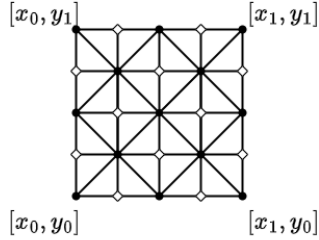
- V dalším kroku otočíme čtverec zpět a předchozí dva kroky aplikujeme na 4 nové čtverce



# Fraktální geometrie

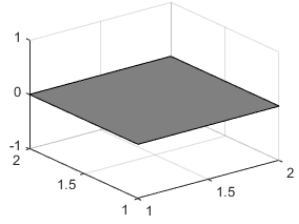
## Metoda náhodného přesouvání středu

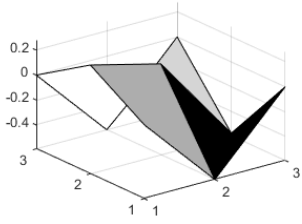
- Při otočení čtverce o 45ř vznikne diamant – **diamond square algorithm**
- Problém je opět určit, kdy zastavit rekurzi – nejčastěji v závislosti na velikosti čtverce

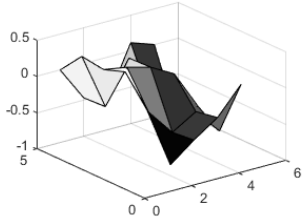


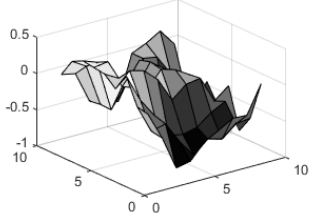
# Fraktální geometrie

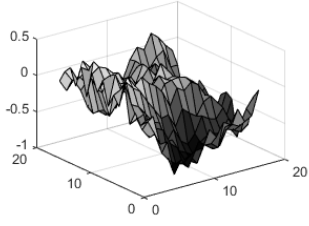
## Metoda náhodného přesouvání středu



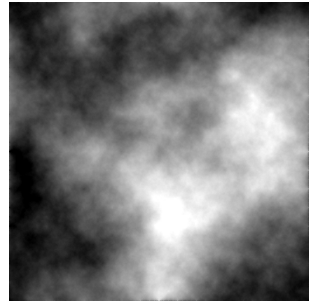



# Fraktální geometrie

## Metoda náhodného přesouvání středu

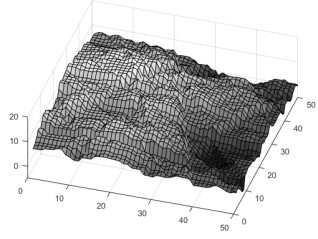
- Při vytváření 2D modelu mraku můžeme využít fraktálních charakteristik fBm plochy
- Zobrazíme-li tuto plochu v kolmém průměru tak, že výšku budeme reprezentovat jako barvu (např. odstíny šedi), získáme poměrně věrohodný obraz mraku
- 3D model – můžeme vyjít z metody náhodného přesouvání středu do vyššího rozměru
- 4. rozměr tohoto fraktálu se interpretuje jako hustota
- Výsledná struktura se obtížně vykresluje – nejčastěji se zobrazuje pomocí metody ray crossing (kde se hustota interpretuje jako útlum), o které budeme mluvit později
- O jiném generování mraků budeme mluvit později



# Fraktální geometrie

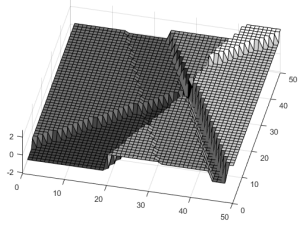
## Metoda náhodných poruch

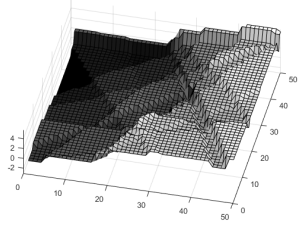
- Další z metod, jak generovat fBm je metoda **náhodných poruch** (random faults method)
- Implementačně nejjednodušší, iterativní
- Algoritmus se aplikuje na 2D matici v níž je hodnota každého bodu výškou
- Na začátku jí nastavíme na 0
- V cyklu provádíme následující kroky:
  - Rozdělíme matici na 2 části (např. protneme přímkou)
  - Body na jedné straně zvýšíme o $d$ a body na druhé o stejnou hodnotu snížíme
- Kromě protínání přímkou můžeme výběr bodů (které snižujeme/zvyšujeme) vybírat tak, že do plochy „obtiskneme“ nějaký tvar (např. různě velké kružnice)

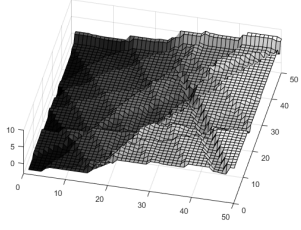


# Fraktální geometrie

## Metoda náhodných poruch



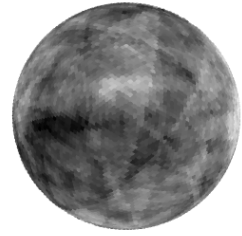



# Fraktální geometrie

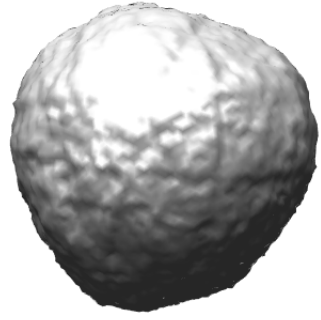
## Metoda náhodných poruch

- Metodou náhodných poruch se generují například i planety, kde se tato metoda aplikuje na kouli
- Koule = hustá síť trojúhelníků (je důležité, aby všechny měly přibližně stejnou velikost)
- Pro generování koule se používá algoritmus rekurzivního zjemňování dvou na sobě položených čtyřstěnů, jejichž vrcholy leží na povrchu koule
- Na začátku se všem vrcholům nastaví stejná barva
- Koule se iterativně rozdělí na 2 části a každé části se přiřadí změna barvy barvy, případně výšky o náhodné gaussovské číslo, které klesá s počtem iterací
- Vedeme-li řez vždy středem koule, je jedna polokoule negativem té druhé



# Fraktální geometrie
## Metoda náhodných poruch

- Takto generované planety nejsou moc reálné, velikost povrchových struktur v reálném světě je zanedbatelná s poloměrem koule
- Jak některá hodnota klesne pod určitou hodnotu, zaokrouhlíme ji a na považujeme ji za vodu
- Reálnější vzhled planet – pomocí fraktální interpolace hranic nějakého povrchového modelu



# Fraktální geometrie
## Krajiny

- Základní datová struktura – **pravidelné výškové pole** (regular height field)
- Pravidelné výškové pole = dvojrozměrná matice, prvky se interpretují jako výška terénu nad základní úrovní
- Nevýhoda – není možné popsat převisy (tedy plně 3D krajinu) – 2.5 rozměrná krajina
- Výšková pole převádíme na síť trojúhelníků
- 3D krajiny (i s převisy) se uchovávají v objemové reprezentaci (datová náročnost, složitější převod na síť polygonů, složitější vykreslování přímo)
- Formáty pro reprezentaci povrchu – DEM (Digital elevation model), rozšíření TIFF (geoTIFF)

# Fraktální geometrie
## Krajiny

- Fraktálně generované krajiny nevypadají úplně přirozeně
- Údolí a hory mívají různý tvar – fraktální krajiny při otočení vzhůru nohama vypadají stejně
- Na krajiny přirozeně působí vnější síly, které otupují ostré hrany – eroze
- Existují algoritmy, které simulují opotřebení
- Nejprve vygenerujeme hrubý tvar krajiny (např. pomocí fraktálů) a dále tyto data upravujeme **erozními algoritmy**

# Fraktální geometrie

## Krajiny a eroze

- **Termoeroze**:
  - Vliv teploty na tvar povrchu
  - Vlivem teplotních rozdílů z povrchu opadávají částečky $\rightarrow$ zahlazení hran
- **Hydroeroze**:
  - Působení vody
  - Globální – kapky deště dopadají na povrch, který je díky tomu hladší (jako u termoeroze, jen intenzivněji), materiál je přenášen z vyšších míst dolů
  - Lokální – drolení částeček tekoucí vodou
- **Větrná eroze**:
  - Nejméně používaná v počítačové grafice
  - Zejména se používá při generování pouští – uchopení částic větrem a přenesení na jiné místo

# Fraktální geometrie
## Modelování krajiny

- V prvním kroku vygenerujeme (vymodelujeme) síť řek (fBm, L-systémy)
- Poté fraktální interpolací vygenerujeme krajinu

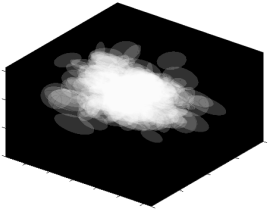
# Modelování mraků

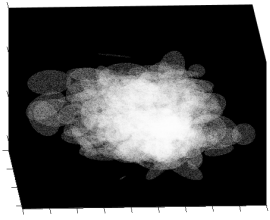
- Kromě již zmíněných možností pomocí fraktálů
  - Náhodné struktury
  - Celulární automaty

# Modelování mraků

## Náhodné struktury

- Objemová struktura, která je tvořena elipsoidy s poloprůhlednou texturou
- K elipsoidům se přidává i funkce šumu, která narušuje pravidelnost elipsoidů
- Funkce se aplikuje průhlednost se kterou se části elipsoidů zobrazují





# Modelování mraků

## Celulární automaty

- **Celulární automat** je označení pro určitý typ fyzikálního modelu reálné situace v podobě algoritmu
- Dynamický systém, diskrétní v hodnotách, prostoru i čase
- Je tvořen pravidelnou strukturou buněk v N-rozměrném prostoru
- Každá buňka nabývá jeden z $K$ možných stavů
- Hodnoty stavů buněk v dalším časovém kroku vypočítáme paralelně na základě přechodové funkce
- Argumenty této funkce jsou aktuální hodnoty stavů vyšetřované buňky a buněk v jejím okolí

# Modelování mraků

## Celulární automaty

- Prostor rozdělíme na 3D mřížku buněk
- Každá buňka nese informaci o 3 logických hodnotách – přítomnost vlhkosti (*hum*), přítomnost mraků (*cld*) a přítomnost fázového posunu (*phs*)
- V každém kroku tyto hodnoty pro každou buňku aktualizujeme na základě hodnot buňky a buněk okolních

# Modelování mraků
## Celulární automaty

**Přechodová funkce**:

- $f(i,j,k) = phs(i+2,j,k,t_i) \vee phs(i+1,j,k,t_i) \vee phs(i-1,j,k,t_i) \vee$
  $phs(i-2,j,k,t_i) \vee phs(i,j+1,k,t_i) \vee phs(i,j-1,k,t_i) \vee phs(i,j-2,k,t_i) \vee$
  $phs(i,j,k+2,t_i) \vee phs(i,j,k+1,t_i) \vee phs(i,j,k-1,t_i) \vee phs(i,j,k-2,t_i)$
- $hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \wedge \neg phs(i,j,k,t_i)$
- $cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \vee phs(i,j,k,t_i)$
- $phs(i,j,k,t_{i+1}) = \neg phs(i,j,k,t_i) \wedge hum(i,j,k,t_i) \wedge f(i,j,k)$

Hodnota $f$ vždy závisí na 2 buňkách v kladném i záporném směru os $x$ a $z$, 2 buňkách nad a 1 pod (tím se modeluje reálný svět)

# Modelování mraků

## Celulární automaty

- „Úmrtnost mraků“ se modeluje tak, že se specifikuje pravděpodobnost $p_{cld}$, v každém kroku se vygeneruje náhodné číslo pro každou buňku, pokud je toto číslo $\leq$ než $p_{cld}$ je *cld* hodnota nastavena na 0
- Pravděpodobnosti $p_{hum}$ a $p_{phs}$ ovlivní vytvoření mraku v nějaké buňce v budoucnosti – pokud je vygenerovaná náhodná hodnota $\leq p_{hum}$ nebo $p_{phs}$ vlastnost se nastaví na 1

**Příklad**

Jak se upraví přechodová funkce, aby zohlednila výše uvedená pravidla?

# Systémy částic

- **Systémy částic** (Particle systems) – se používají k modelování objektů, jejichž tvar je natolik členitý (nebo se mění takovým způsobem), že ho není možné reprezentovat povrchem
- Např. hejna ptáků, padající sníh, oheň, mlha, tráva, les a jiné
- **Reprezentace** – soubor velmi malých prvků (částic), jejichž vlastnosti se mění v čase
- V nějaké místě mohou vznikat, případně mizet, mohou mít libovolný tvar (vločka, bod, ...)
- Klíčovou roli hrají náhodná čísla
- Vlastnosti $x = s + rand() \cdot v$ ($s$ střední hodnota náhodných čísel, $v$ rozptyl, $rand()$ náhodné číslo)
- $x$ reprezentuje například barvu, dobu života a pod.

# Systémy částic

1. Na základě parametrů každé existující částice se aktualizuje její poloha a ostatní atributy
2. V celém systému se vytvoří nové částice. Ty vznikají v nějaké konkrétní oblasti, nebo mohou vznikat jako potomci jiných částic
3. Každé nové částici se přiřadí vlastnosti
4. Částice, které překročily dobu životnosti zaniknou
5. Systém se zobrazí

# Systémy částic

## Ohňostroj

### Vlastnosti

- Životnost – Každá částice má počítadlo, které se v každém kroku dekrementuje. Jakmile dojde k nule, částice přestane existovat.
- Pozice a směrový vektor – Částice si uchovává informaci o tom, kde se nachází a jakým směrem a jakou rychlostí se pohybuje. V každé iteraci se pozice změní o zadaný vektor.
- Velikost – Informace o velikosti částice. Pokud se velikost v průběhu času mění, můžeme zadat i o kolik se v každém kroku změní.
- Barva – Částici můžeme přiřadit barvu, průhlednost, případně informaci o změně v průběhu času (můžeme například snižovat jas)
- Tvar – Například koule, hvězda a jiné.

# Systémy částic
## Ohňostroj

### Chování

- Částice za sebou nechává nové malé částice, které mají krátkou životnost a buď se pohybují stejným směrem, nebo zůstávají na místě
- Částice může na konci životnosti explodovat a vytvořit kouli nových částic
- Na pohyb částice může působit gravitace (tím se bude měnit její dráha)
- Částice může blikat (v nějakých iteracích se nezobrazí)

**Příklad**

Upravte kód generování ohňostroje (popište změnu), aby měl jedno z výše uvedených chování.

# Systémy částic
## Příklad

**Příklad**

Popište chování systému částic simulující prskavku.

# Systémy částic
## Další příklady

- Bublinky v limonádě – vznikají na sklenici (mají různou velikost) a pohybují se směrem vzhůru, jak narazí na hladinu, končí jejich životnost
- Oheň – Částice vznikají v základní části ohně a pohybují se směrem nahoru. Pohyb částic je ovlivněn funkcí šumu (Flow noise) a faktory z okolí (například směr a síla větru) – ty jsou definovány jako částice, které vymezují možný tvar ohně. Čím je částice dále od zdroje, stává se průhlednější.

# Dynamické simulace

- Aplikace systému částic – dynamické simulace, které využívají fyzikálních modelů pohybu a interakcí
- Př. Koš míčů vysypaný na schodech
  - Generující objekt – koš, který udělil všem částicím (míčům) počáteční rychlost a směr
  - Každý míč má určitou pružnost a hmotnost a pohybuje se po schodech směrem dolů
  - Míče se mohou mezi sebou srážet, narážet do stěn
- Jádrem tohoto systému je na fyzice založená simulace

# Dynamické simulace

- Stav $y(t)$ v čase $t$ každé částice je určen jako uspořádaná dvojice $y(t) = [X(t), \overrightarrow{v}(t)]$
- $X(t) = [x(t), y(t), z(t)]$ ... poloha částice
- $\overrightarrow{v}(t) = (v_x(t), v_y(t), v_z(t))$ ... vektor okamžité rychlosti
- Částice má hmotnost $m$, která je v čase neměnná
- **Změna rychlosti**: $\frac{\mathrm{d}X(t)}{\mathrm{d}t} = \overrightarrow{v}(t)$
  $\frac{\mathrm{d}\overrightarrow{v}(t)}{\mathrm{d}dt} = \frac{\overrightarrow{F}(t)}{m}$
- $\overrightarrow{F}(t)$ ... vektor sil působící na částici
- Nejjednodušší způsob – aplikace **Eulerovy metody**, kde systém převedeme na
  $X(t + \Delta t) = X(t) + \Delta t \overrightarrow{v}(t)$
  $\overrightarrow{v}(t + \Delta t) = \overrightarrow{v}(t) + \Delta t \frac{\overrightarrow{F}(t)}{m}$

# Dynamické simulace

- **Postup**:
  - Vynulujeme čítač sil, který je přiřazen částici
  - K čítači přičteme všechny síly, které na částici působí
  - Nejčastější silou je zemská přitažlivost – $F_g = m \cdot g$ ($g \approx 9.82 m/s^2$), viskozita – $F_v = \overrightarrow{v}(t) \cdot c$ ($c$ konstanta udávající viskozitu prostředí), vítr a jiné.
  - Výsledná síla se použije ve vztahu $\overrightarrow{v}(t + \Delta t) = \overrightarrow{v}(t) + \Delta t \frac{\overrightarrow{F}(t)}{m}$
  - Tento výsledek se pak použije pro výpočet nové polohy částice $X(t + \Delta t) = X(t) + \Delta t \overrightarrow{v}(t)$
- Eulerova metoda je implementačně jednoduchá, ale ne vždy vhodná – u dynamických systémů může dojít k oscilaci částic v případě, kdy by měly být v klidu

# Dynamické simulace

## Řešení kolizí mezi sebou

- Detekce kolizí částic s objekty i mezi sebou
- Nejjednodušší řešení – objekty reprezentované v ploškové reprezentaci
- Každá ploška leží v rovině, která je dána bodem a normálovým vektorem $\overrightarrow{n}$
- Detekce kolize částice s rovinou je snadná – dojde ke změně znaménka skalárního součinu $\overrightarrow{n} \cdot (P - X(t))$
- Pokud platí $sgn(\overrightarrow{n} \cdot (P - X(t))) \neq sgn(\overrightarrow{n} \cdot (P - X(t + \Delta t)))$ – došlo ke kolizi s rovinou (ne však nutně s ploškou)
- Místo srážky se určí jako průsečík úsečky $X(t)$ a $X(t + \delta t)$ s rovinou $(P, \overrightarrow{n})$ – pro jednoduchost se v případě malého $\Delta t$ bere jen $X(t)$
- Po detekci kolize se určí změna směru vektoru rychlosti po odražení od roviny – rozklad vektoru rychlosti na rovnoběžnou složku a kolmou složku k normálovému vektoru roviny
- Vektor roviny orientovaný směrem k rovině, $\overrightarrow{v}'$ – vektor odrazu
- $\overrightarrow{v}' = \overrightarrow{v_t}' - \overrightarrow{v_n}'$
  $\overrightarrow{v_n}' = (\overrightarrow{n} \cdot \overrightarrow{v})\overrightarrow{n}$ (normálová složka)
  $\overrightarrow{v_t}' = \overrightarrow{v} - \overrightarrow{v_n}'$ (tečná složka)

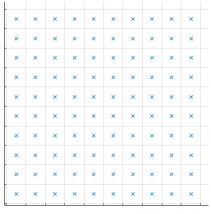
# Dynamické simulace

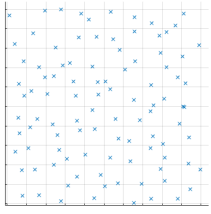
**Příklad**

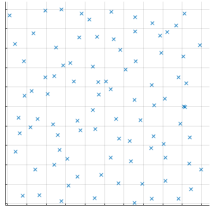
Popište systém simulující padající sníh.

# Rozmístění objektů ve scéně

- Chceme-li vytvořit scénu obsahující velké množství „stejných“ objektů (les, tráva), vyvstává problém jak ve scéně rozmístit
- Plocha, na které mají být objekty umístěny, se rozdělí na čtverce a určí se místo (střed), kde mají být objekty umístěny
- Tento bod se náhodně posune tak, aby neopustil hranici čtverce (z adaptivního vzorkování známe jako jittering)
- Poté určité množství (např. 10%) bodů odstraníme
- Na každou pozici umístíme objekt







# Rozmístění objektů ve scéně

## Rostliny

- Dříve popsaný způsob nezohledňuje přirozené shlukování rostlin do skupin
- Modely založené na principech **umělého života** (artificial life)
- Rostliny do systému umístíme náhodně, každá rostlina má své ekologické okolí (kruh získaný projekcí rostliny do země)
- Jak rostlina roste, ekologické prostředí se zvětšuje
- Pokud se dvě prostředí protnou, určí se, která rostlina je silnější, slabší se vyloučí
- Po několika iteracích je rozložení rostlin věrohodné

# Textury

- **Textura** - popis vlastnosti povrchu
- Často je efektivnější použít jednoduchou geometrii a složitou texturu, než definovat složité geometrické detaily
- Objekty, které jsou v dálce, nebo zobrazeny krátce jsou od složitě definovaných k nerozeznání
- Texturám se budeme věnovat později.