**FYSIKUPPGIFTER**

Använd Session05 kod för detta. Den har rätt dimensioner (g = 981cm/s^2). Varje pixel = 1x1 cm stor. Massan är i kilogram, tiden är i sekunder. För att få exakta värden när man simulerar rekommenderas att tidssteget är litet (dt <= 0.001)

I Session05 koden finns några nya funktionaliteter som kan användas:

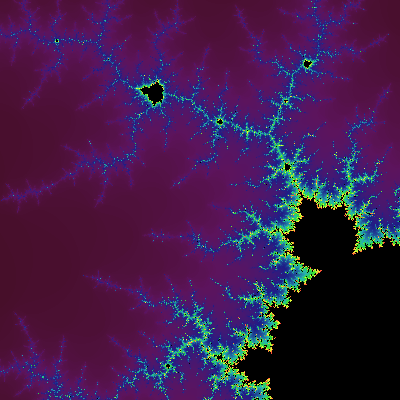
* p.drawTail = True eller False
  + Ritar upp partikelns positioner under de senaste 10 simulerade sekunderna.
* p.reverseV()
  + Byter håll på hastigheten och ändrar färg på svansen. Går endast göra 1 gång/simulering.
* Håller du nere en musknapp skrivs positionen och tidpunkten ut för den närmaste svans-delen.

1. En boll med radie 10 cm befinner sig 3.5 m över marken och släpps från vila.
   1. Vid vilken tidpunkt slår den i marken?
   2. Vilken hastighet har den precis innan den når marken?
2. En boll befinner sig på marken när den helt plötsligt får samma hastighet som bollen i uppgift 1 hade när den träffade marken, dock in motsatt riktning.
   1. Hur högt når bollen?
   2. Vilka likheter finns det med bollen i uppgift 1?
3. En boll studsar runt i en låda.
   1. Vad händer om den byter riktning efter 5 s?
   2. Var kommer bollen vara efter 5 s till?
   3. Vad händer om bollen förlorar energi i varje studs?
   4. Vad händer om bollen känner luftmotstånd?
4. Sätt upp en riktig modell av bollen ni får tillgång till.
   1. Lägg in: Vikt, Storlek, Luftmotstånd, Energi den förlorar när den studsar.
      1. Studera gärna ekvationen här:

<http://www.softschools.com/formulas/physics/air_resistance_formula/85/>  
och ta reda på vad ni måste mäta för att simulera bollen mer exakt.

* 1. Simulera bollen och jämför med verkligheten.

**PROGRAMMERINGSUPPGIFTER**

**Mandelbrotfraktal**

Mandelbrotsetet är alla komplexa tal som följer följande begränsning:

divergerar inte när ,

är det komplexa talet den börjar på och

I programmet kommer vi använda y-axeln som den komplexa axeln. För varje komplext tal (x,y) i bilden ska vi kolla om ekvationen ovan divergerar. Ekvationen divergerar om > 2.

Låt *n* gå upp till *255*. Om ekvationen divergerar, använd *n* för att bestämma färgen på pixeln *(x,y)* i fönstret.

Pseudokod:

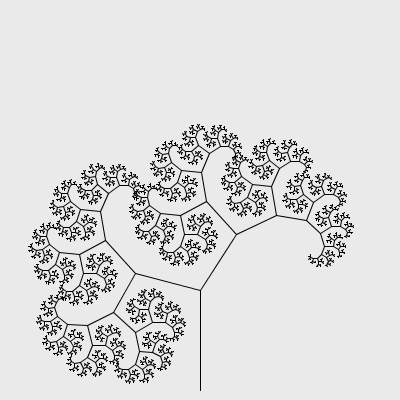
**for each pixel:**  
 find the graph\_values from the window\_values  
 cRe = map(x)  
 cIm = map(y)  
 zRe = 0  
 zIm = 0  
 n = 0

**while n < 255:**  
 new\_zRe = cRe + zRe\*zRe – zIm\*zIm  
 new\_zIm = cIm + 2\*zIm\*zRe  
 zRe = new\_zRe  
 zIm = new\_zIm  
 **if** dist(zRe, zIm, 0, 0) > 2:  
 break  
 n = n+1

set the color of the pixel at [x, y], this should depend on n

Använd map() för att skala om pixel-positionen till ett värde mellan (-1, 1).  
x = map(pixel\_X, 0, width, -1, 1)  
y = map(pixel\_Y, 0, height, -1, 1)

**L-system-fraktal**

L-System är ett sätt att beskriva träd-liknande system som följer speciella regler. Läs gärna mer här: <https://en.wikipedia.org/wiki/L-system>  
  
I exemplet till höger används en av de simplaste reglerna för att skapa komplexa mönster.

För att skapa dessa mönster används en ide som kallas *Rekursion*, vilket betyder att något definieras av termer av sig själv. I detta fall har vi en gren som definieras som:

**Gren**En linje som delas upp i två grenar.

Denna rekursion pågår i oändligheten eftersom grenen innehåller två grenar, som innehåller två grenar var, som innehåller två grenar var ...

För att undvika att vi hamnar i oändliga loopar när vi programmerar, behöver vi ett sätt att stoppa rekursionen. Detta görs bäst om vi stannar rekursionen om linjen är kortare än en pixel lång.

Pseudokod:

**def branch(x, y, L, a):  
 if L < 1:**  
 return  
 **else:**  
 x2 = x + L\*cos(a)  
 y2 = y + L\*sin(a)  
 line(x, y, x2, y2)  
  
 branch(x2, y2, L/1.5, a + 5.0)  
 branch(x2, y2, L/1.5, a – 5.0)

För att ändra hur den ser ut, ändra ±5 värden i sista raderna. De behöver inte vara samma. Experimentera med olika. Längden på linjen behöver inte heller ändras likadant. Vad händer om man lägger in slumpmässiga värden?

(OBS, programmet kan krascha om linjen inte blir kort fort nog. Processing kan endast hantera ett visst antal rekursioner.)