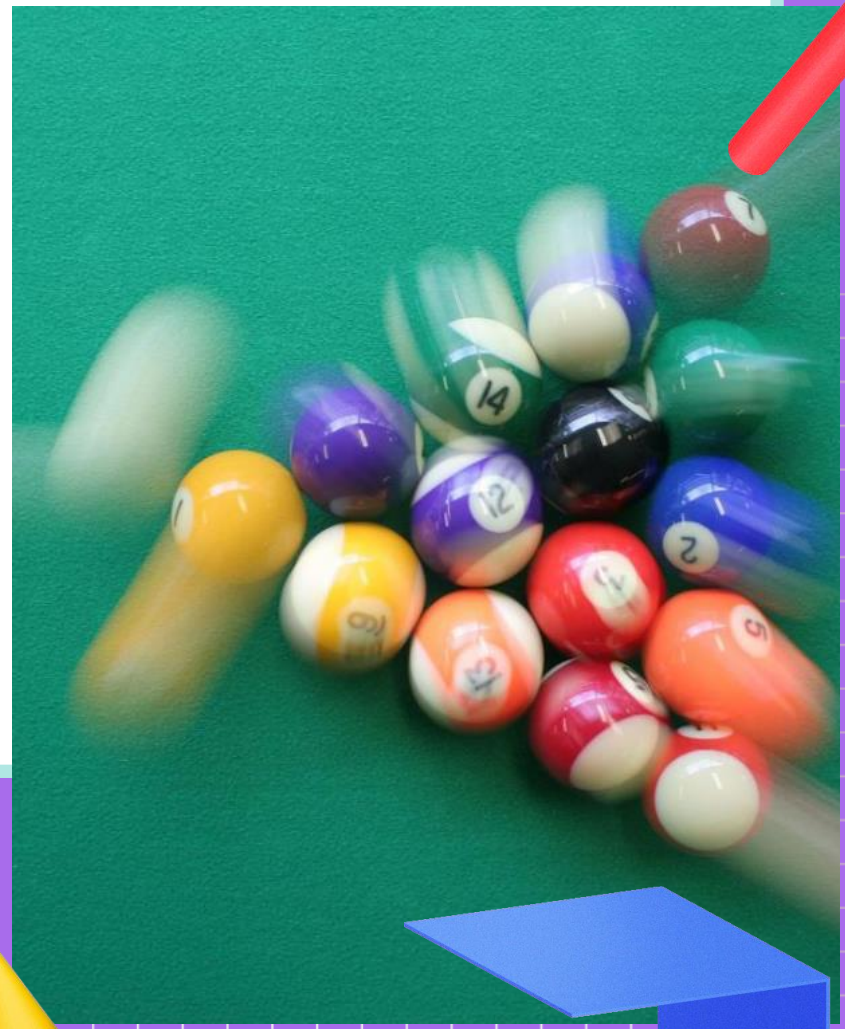
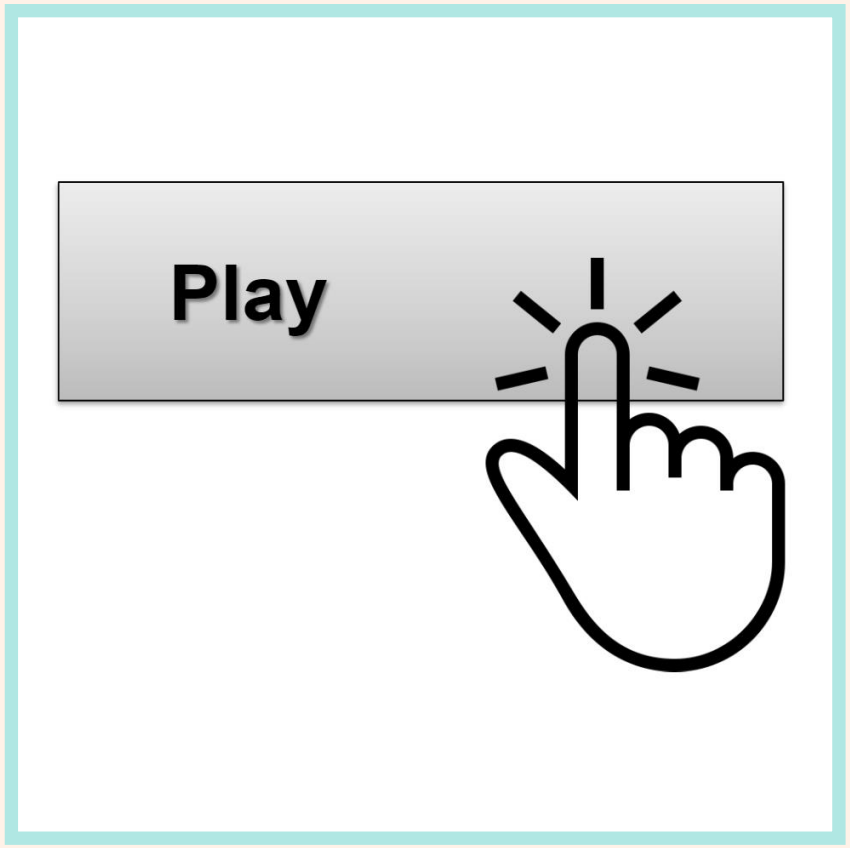
Programarea aplicațiilor de simulare

Detectarea coliziunilor



Detectarea coliziunilor

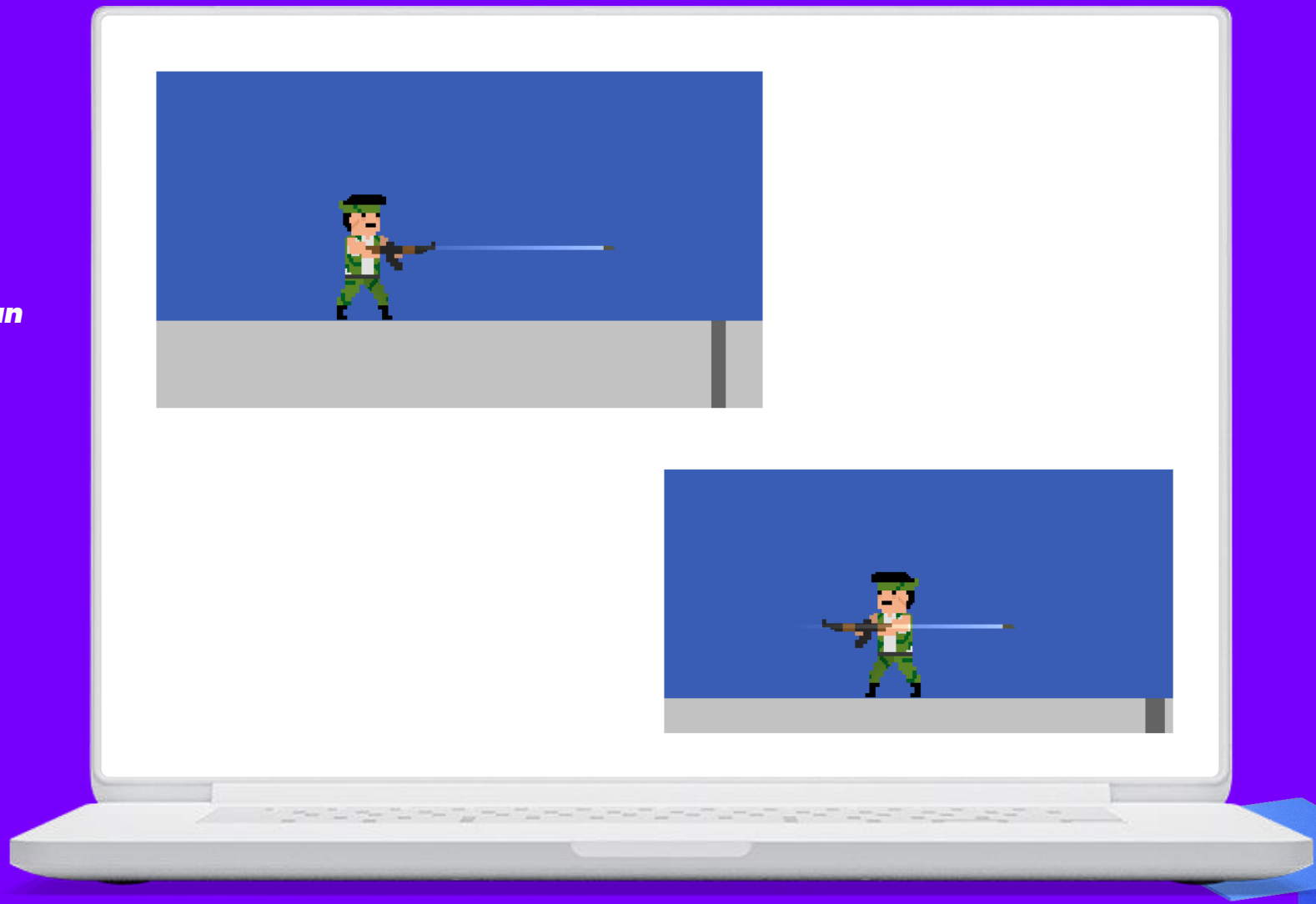
Motivație: Selectarea obiectelor



- *Punctele din interiorul unei regiuni?*

Motivație: Traiectoriile gloanțelor

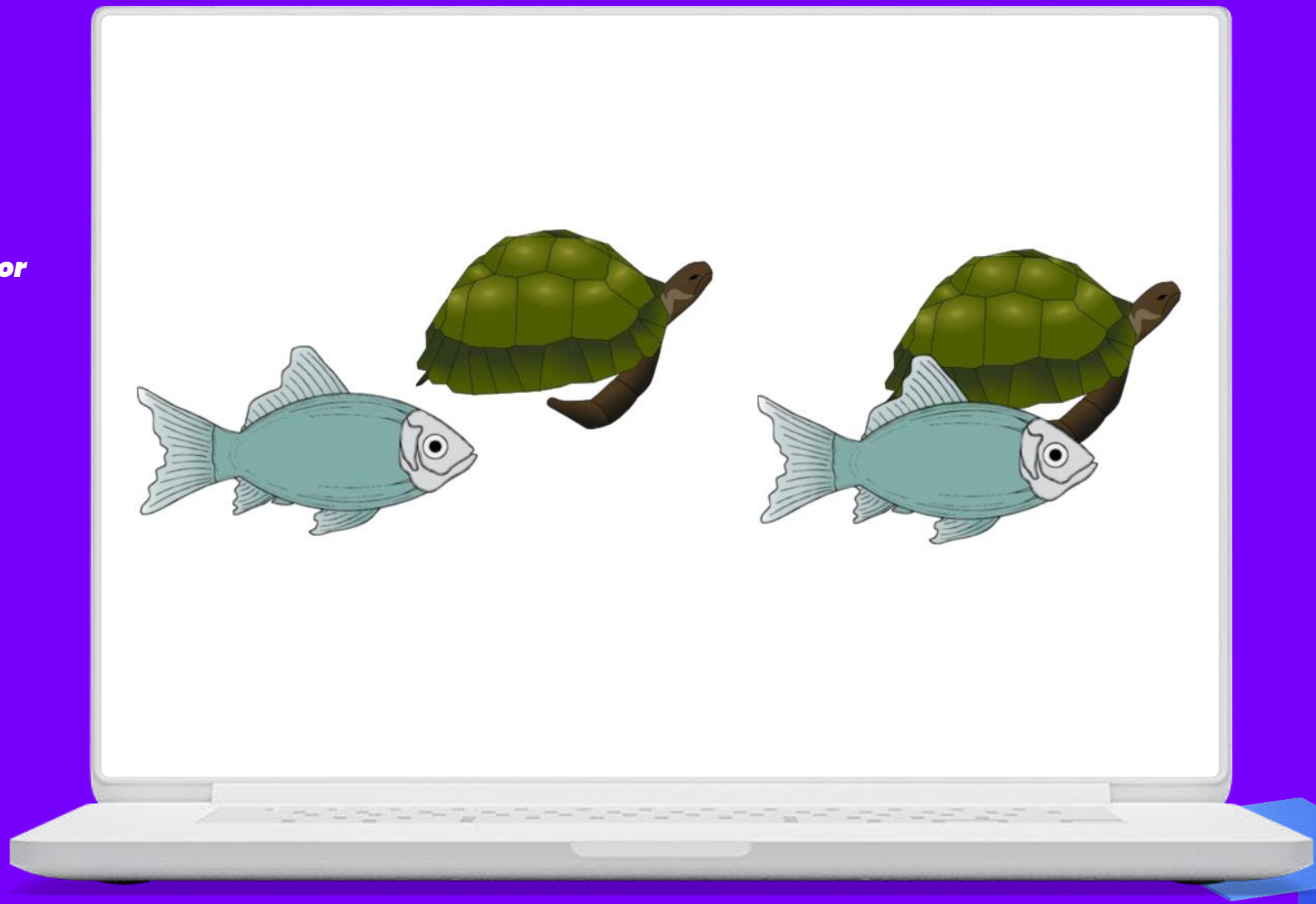
- ***Intersecția dintre un punct și un obiect sau intersecția dintre o linie și un obiect?***



https://forum.unity.com/threads/2d-platformer-shooting.365971/

Motivație: Coliziuni

- ***Prevenirea penetrării obiectelor***
- ***Cum?***



Configurații posibile?

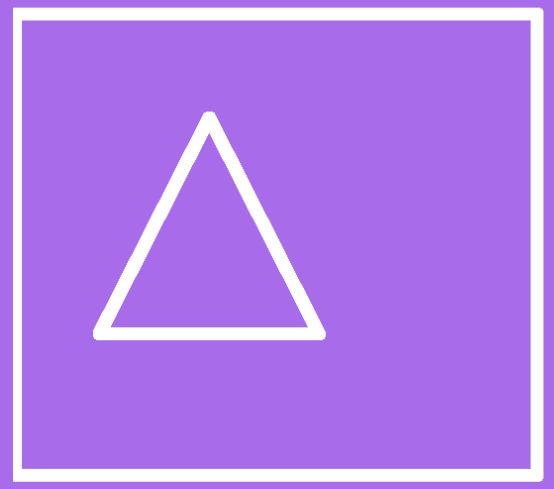
Pentru a detecta coliziunea dintre două poligoane este suficient să verificăm dacă muchiile acestora se intersectează

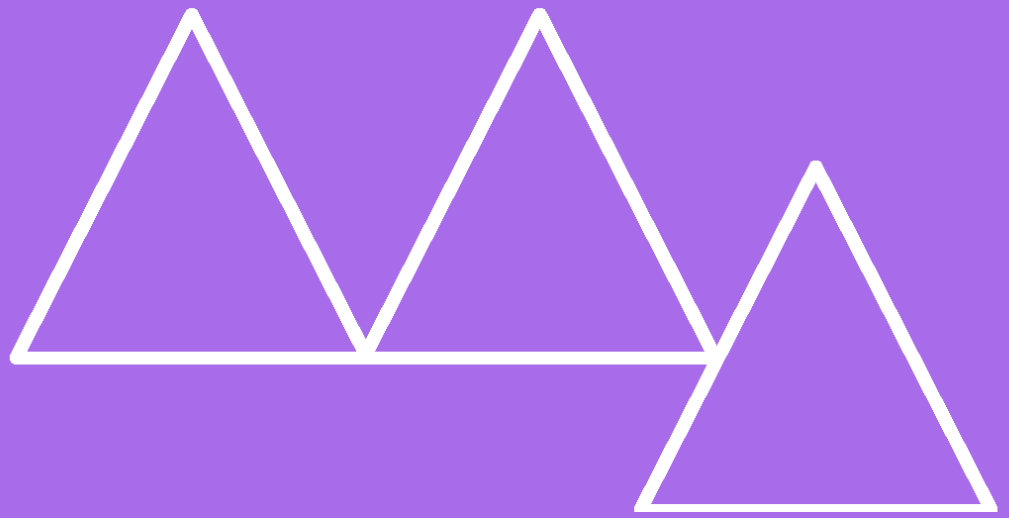
A. Adevărat

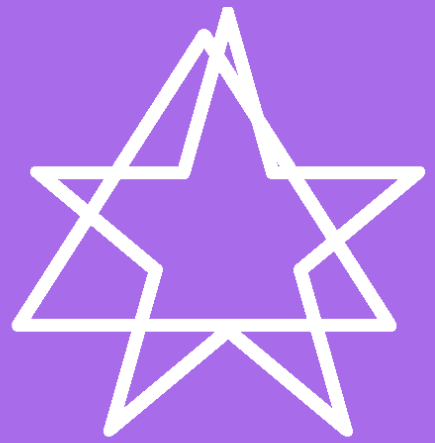
B. Fals

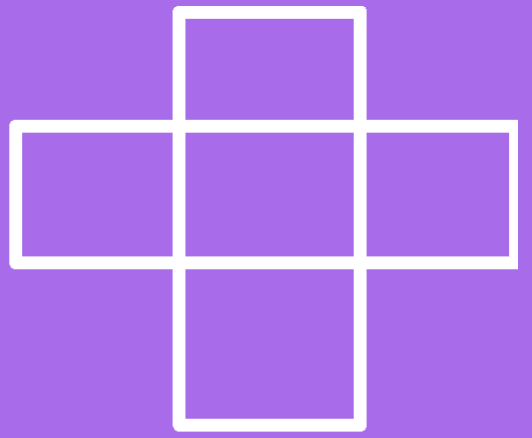
Configurații posibile?

- ***Intersecția dintre segmente***
  - ***Punct PE segment***
- ***Poligon în interiorul altui poligon***









Detectarea dacă un poligon se află în interiorul altuia.

- ***Cum?***
- ***Pentru fiecare punct verificăm dacă se află în interiorul celuilalt?***
- ***Cum?***
  - ***Poligon convex***
    - ***Testul de orientare returnează același rezultat pentru toate muchiile***
  - ***Poligon concav***
    - ***Subdivizare = triangulare***
    - ***Cum?***
    - ***Tragem raze (avem grijă la colțuri și cazuri speciale)***

Funcții explicite

- ***Reprezentări matematice***

- **Funcții explicite**
- **Funcții parametrice**
- **Funcții implicite**

Funcții explicite

- ***Reprezentări matematice***
- $y = f(x)$
- **Ex.** $y = ax + b$
- **O singură valoare y pentru fiecare x**
- **Folositor pentru?**
  - **Teren**

Funcții parametrice

- **2D: x și y sunt funcții care acceptă un parametru t**
- **3D: x, y și z sunt funcții care acceptă un parametru t**

$$C(t) = \begin{pmatrix} p_x \\ p_y \end{pmatrix} t + \begin{pmatrix} q_x \\ q_y \end{pmatrix} (1 - t)$$

Segment de dreaptă

$$C(t) = \begin{pmatrix} cos(t) \\ sin(t) \end{pmatrix}$$

Arc de cerc

- **Parametrul t este cuprins într-un interval $t_1 < t < t_2$**

Funcții implicite

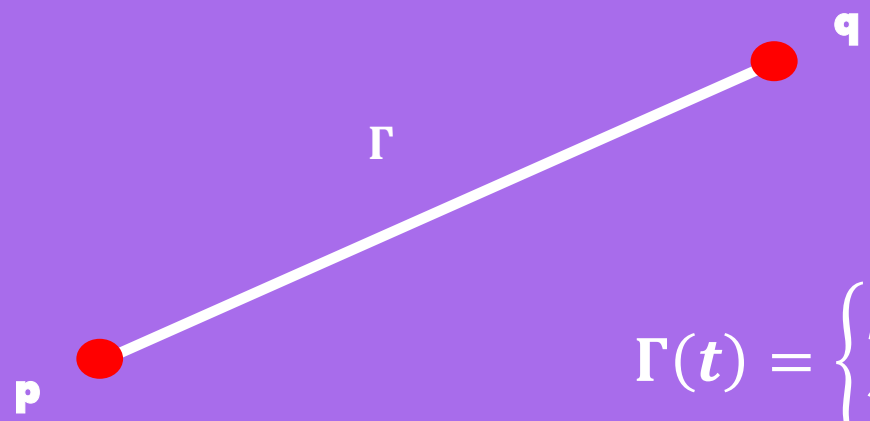
- **Curbă (2D) sau Suprafață (3D) definită de soluțiile unei ecuații.**
- Ex:

$$S(x, y): x^2 + y^2 - 1 = 0$$

$$S(x, y, z): x^2 + y^2 + z^2 - 1 = 0$$

Drepte și segmente

Segmentul Γ de la $p = (x_o, y_0)$ către $q = (x_1, y_1)$



$$\Gamma(t) = \begin{cases} x_1(t) = x_0 + (x_1 - x_0)t \\ y_1(t) = y_0 + (y_1 - y_0)t \end{cases}$$

Găsirea liniei care trece prin punctele $p = (x_o, y_0)$ și $q = (x_1, y_1)$

- **Funcție parametrică:** $\Gamma(t), t \in (-\infty, \infty)$
-
- **Funcție implicită:** $Ax + By + C = 0$
  - ***Sistem de 2 ecuații cu 2 necunoscute*** ***($A^2 + B^2 = 1$)***

Drepte implicite

Explicit: $y = mx + b$

$y = mx + b \Rightarrow$

$\Rightarrow \mathrm{y} = \frac{\Delta \mathrm{y}}{\Delta x} x + b \Rightarrow$

$\Rightarrow \Delta \mathrm{x}\, \mathrm{y} = \frac{\Delta \mathrm{x} \Delta \mathrm{y}}{\Delta x} x + \Delta \mathrm{x} b \Rightarrow$

$\Rightarrow \Delta \mathrm{x}\, \mathrm{y} = \Delta \mathrm{y}\, \mathrm{x} + \Delta \mathrm{x} \mathrm{b} \Rightarrow$

$\Rightarrow 0 = \Delta \mathrm{y}\, \mathrm{x} - \Delta \mathrm{x}\, \mathrm{y} + \Delta \mathrm{x} \mathrm{b} \Rightarrow$

$\Rightarrow A = \Delta \mathrm{y}, \mathrm{B} = -\Delta \mathrm{x}, \mathrm{C} = \Delta \mathrm{x} \mathrm{b}$

Implicit: $Ax + By + C = 0$

Exemplu

$y = \frac{-1}{3} x + 0$

$\Delta \mathrm{x} = -3, \Delta \mathrm{y} = 1, \mathrm{A} = 1, \mathrm{B} = 3, \mathrm{C} = 0 \Leftrightarrow$

$\Leftrightarrow 1\mathrm{x} + 3\mathrm{y} = 0$

Linii implicite – stânga sau dreapta?

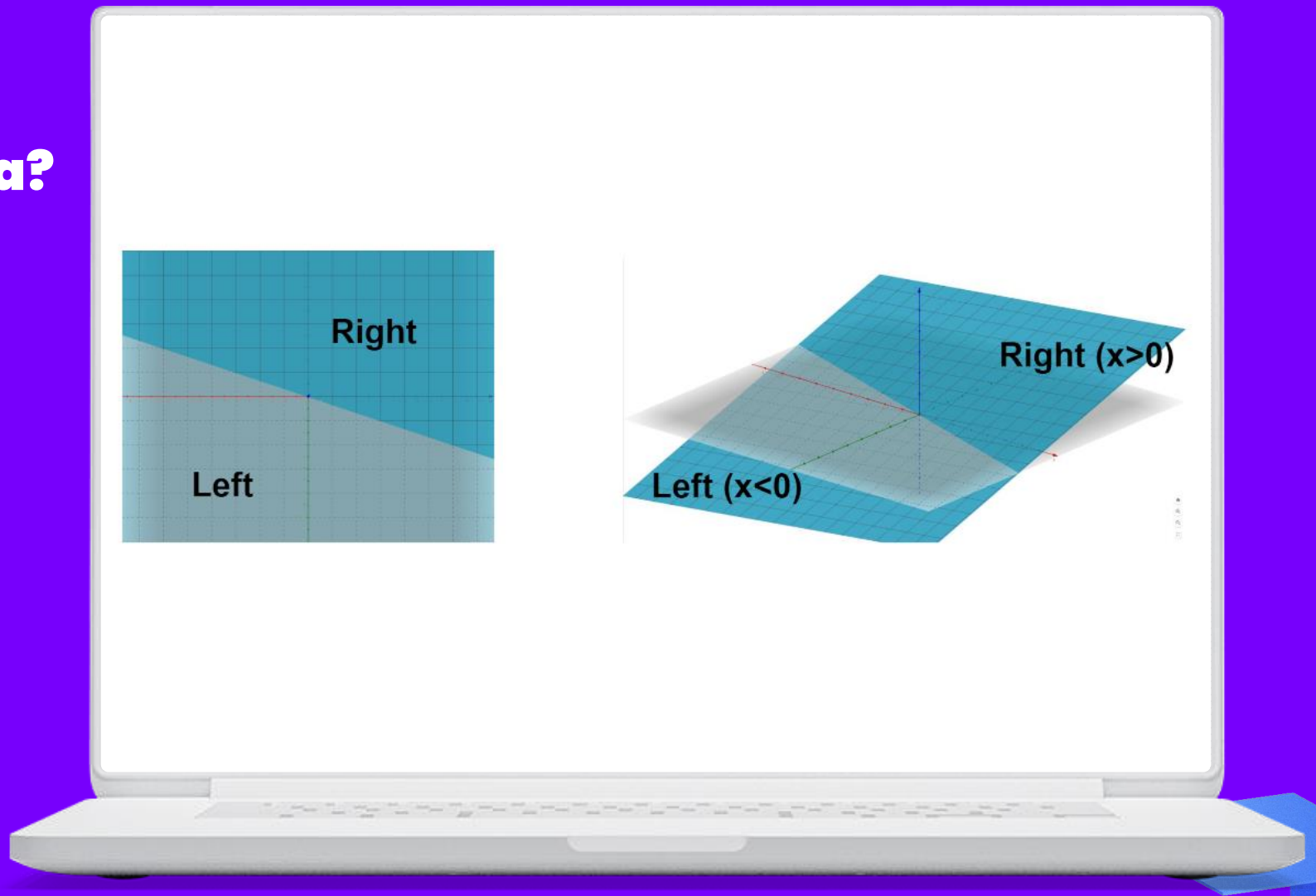
Linie implicită în 2D

$$0.1x + 0.3y = 0$$

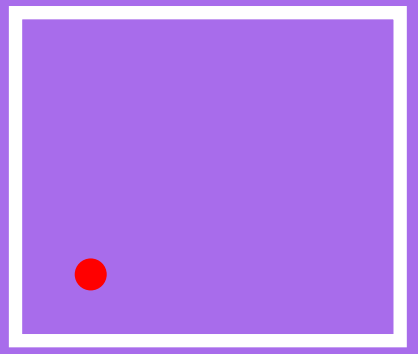
↓↑

Plan explicit în 3D

$$f(x, y) = 0.1x + 0.3y$$



Punct vs dreaptă (rază)

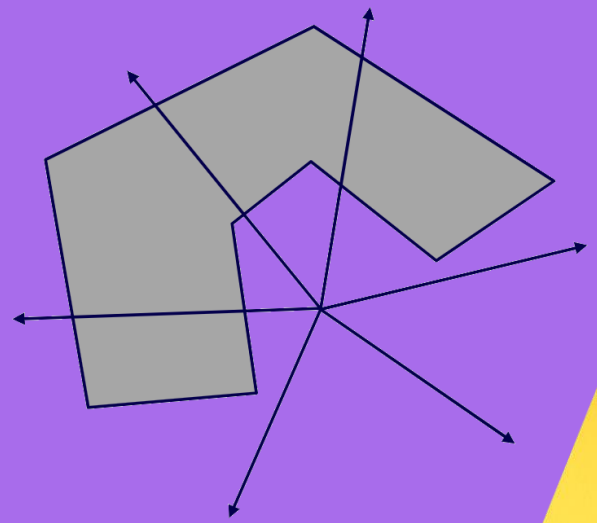


- **Punctul** $p = (p_x, p_y)$
- **Se folosește ecuația implicită**
  - ***Implicit*** $Ax + By + C = 0$
  - ***Se rezolvă un sistem de 2 ecuații cu 2 necunoscute (*** $A^2 + B^2 = 1$***)***
  - ***Pe dreaptă:*** $Ap_x + Bp_y + C = 0$
  - ***Se aplică testul de orientare față de toate muchiile poligonului***
    - ***Dacă semnul este același, atunci punctul se află în interior***
    - ***Ex: Pentru TOATE muchiile avem*** $Ap_x + Bp_y + C < 0$

Recap:
Detectarea dacă un poligon se află în interiorul altuia.

Pentru fiecare punct verificăm dacă se află în interiorul celuilalt

- ***Cum?***
  - ***Poligon convex***
    - ***Testul de orientare returnează același rezultat pentru toate muchiile***
  - ***Poligon concav***
    - ***Subdivizare = triangulare***
    - ***Cum?***
    - ***Tragem raze (avem grijă la colțuri și cazuri speciale)***
    - ***Alte moduri?***



Studiu individual:
Numărul de înfășurare

Punct în poligon?

- **Dacă numărul de înfășurare este diferit de 0**

- **Cum se calculează?**
- **http://geomalgorithms.com/a03-_inclusion.html**

Numărul de înfășurare

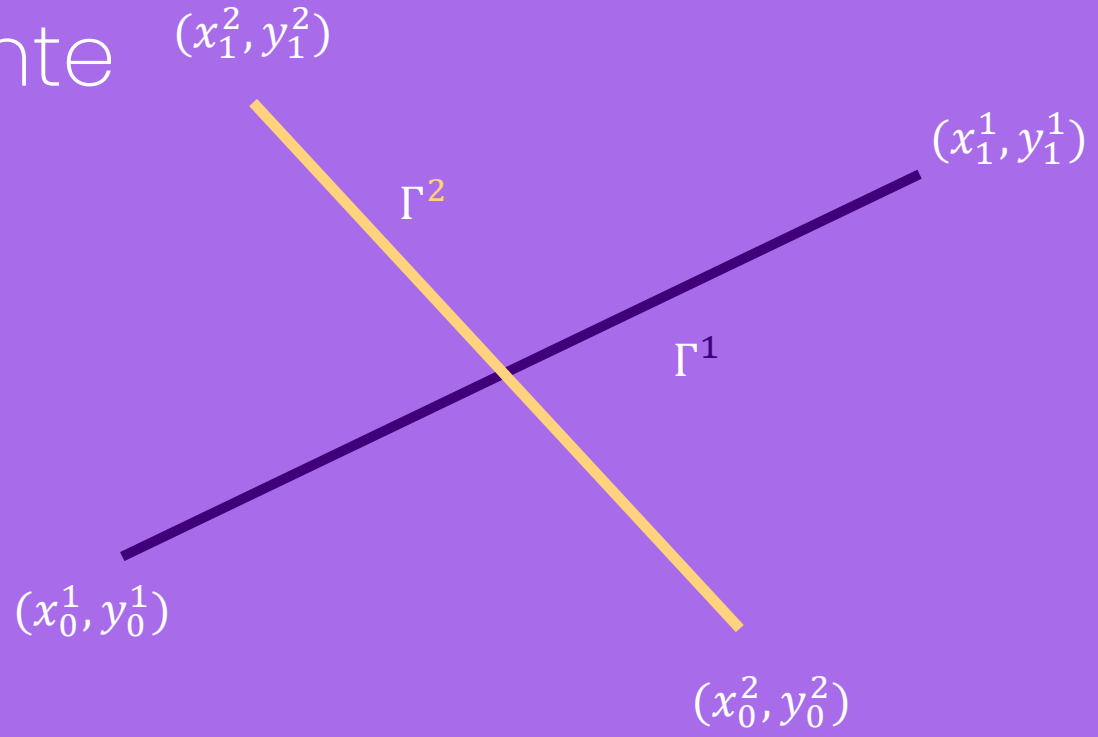
- **De câte ori curba traversează în sens trigonometric în jurul punctului**

- **Negativ în sensul acelor de ceasornic**

Intersecția între două segmente

$$\Gamma^1(t) = \begin{cases} x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ y^1(t) = y_0^1 + (y_1^1 - y_0^1)t \end{cases} \quad t \in [0,1]$$

$$\Gamma^2(r) = \begin{cases} x^2(t) = x_0^2 + (x_1^2 - x_0^2)r \\ y^2(t) = y_0^2 + (y_1^2 - y_0^2)r \end{cases} \quad r \in [0,1]$$



Intersecția: valorile x & y astfel încât acestea sunt egale în ambele reprezentări - Se obțin 2 ecuații cu 2 necunoscute (r,t)

$$x^1(t) = x_0^1 + (x_1^1 - x_0^1)t = x^2(t) = x_0^2 + (x_1^2 - x_0^2)r$$
$$y^1(t) = y_0^1 + (y_1^1 - y_0^1)t = y^2(t) = y_0^2 + (y_1^2 - y_0^2)r$$

Întrebare: ce inseamnă dacă soluțiile ecuației sunt r,t < 0 sau r,t > 1?

Întrebare: **Ce inseamnă dacă soluțiile ecuației sunt r,t < 0 sau r,t > 1?**

A. Segmentele încă se intersectează

B. Segmentele nu se intersectează

C. Se află într-o stare cuantică în care ne este imposibil să decidem dacă se intersectează sau nu.

Eficiență

- **Implementarea naivă**
  - **Verifică coliziunile între TOATE obiectele mișcătoare la un anumit pas**
  - **Foarte costisitoare**
- **Cum îmbunătățim?**

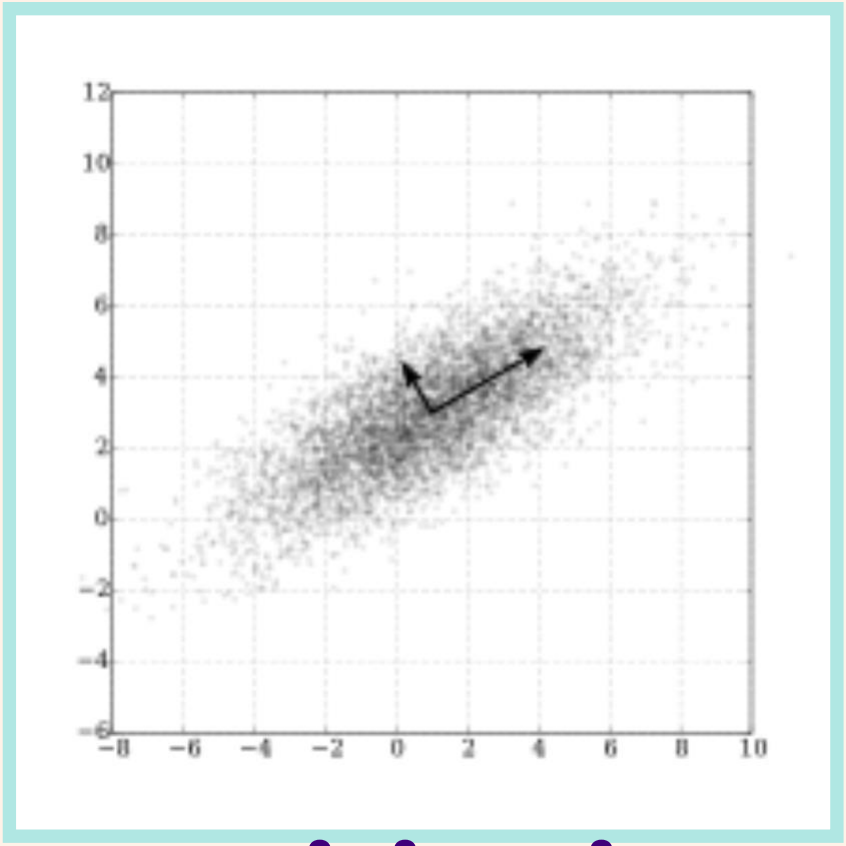
Eficiență

- **Implementarea naivă**
  - **Verifică coliziunile între TOATE obiectele mișcătoare la un anumit pas**
  - **Foarte costisitoare**
- **Îmbunătățiri**
  - **Bounding Volumes**
  - **Ierarhii**

Bounding Volumes

- **Axis aligned bounding box (AABB)**
  - **Computațiile sunt triviale**
  - **Eficiente**
  - **Posibil să fie prea mari...**
- **Tight bounding box**
  - **Mai greu de calculat: Principal Component Analysis (PCA)**
  - **Evaluare mai lentă**
  - **Compacte**

Principal Component Analysis(PCA)



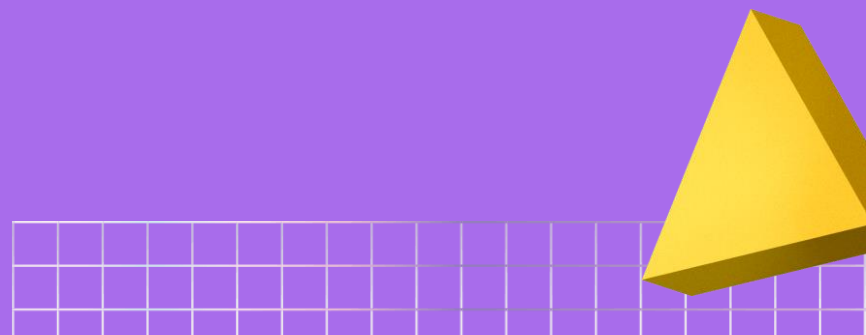
Wikipedia

Se găsesc direcțiile de varianță maximă

$$W_{(1)} = \underset{||w|| = 1}{\arg max} \left\{ \sum_i \left(x_{(i)} \cdot w\right)^2 \right\}$$

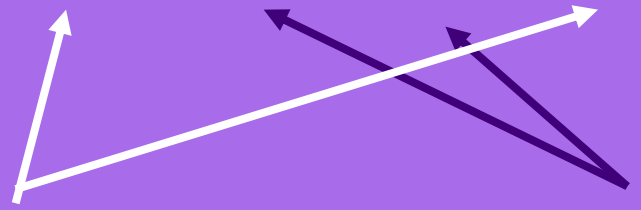
Bounding Volumes

- **Bounding circle**
  - **Mai multe metode de a calcula eficient**
- **Acoperirea convexă**
  - **Vezi curs Algoritmi Avansați**



Bounding Volumes Intersection

- **Axis aligned bounding box (AABB)**
  - **A.LO <= B.HI && A.HI >= B.LO (și pentru X și pentru Y)**



- **Limita inferioară**
- **Limita superioară**

- **Cercuri**
  - **||A.C – B.C|| < A.R + B.R**

Centrele **Razele**

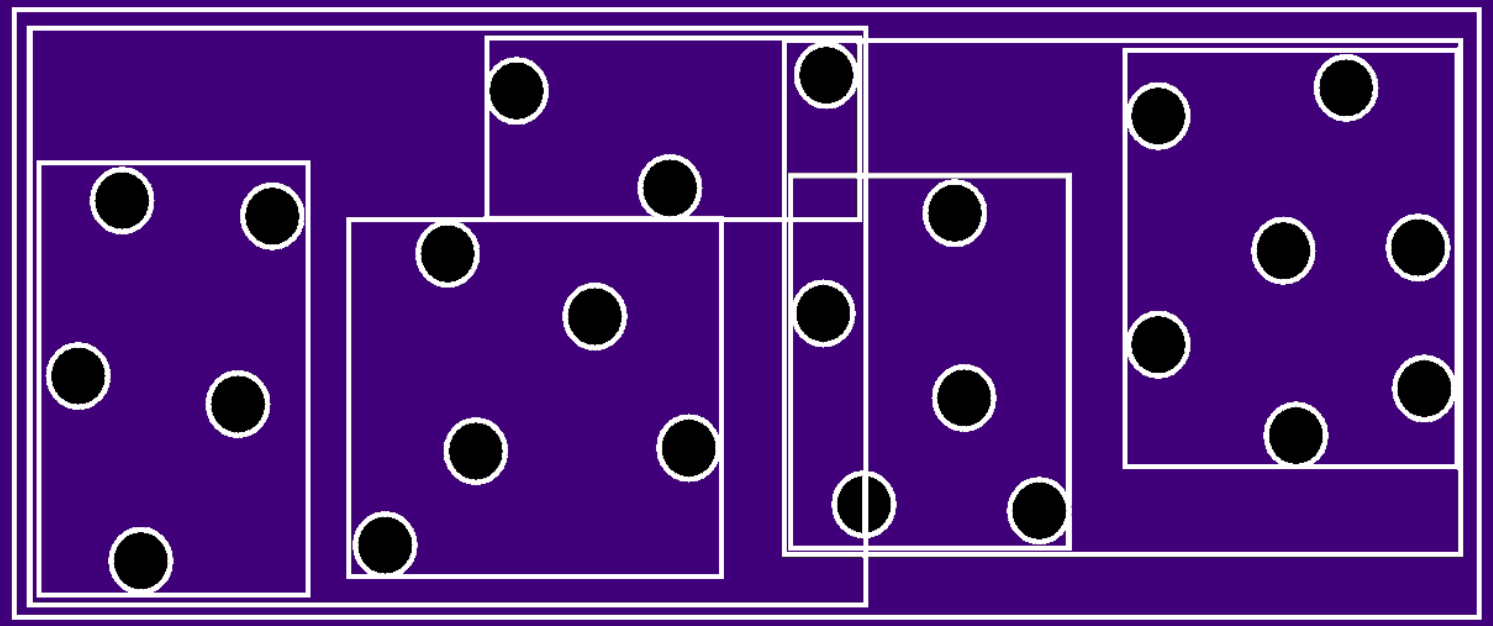
Obiecte mișcătoare

- **Baleiere – testează intersecțiile pe segmentul dintre poziția inițială și cea finală**
  - **Se rezolvă glitch-urile prin care obiectele trec prin alte obiecte**
  - **Cum facem asta eficient?**

- **Boxes?**
- **Spheres?**

Hierarchical Bounding Volumes

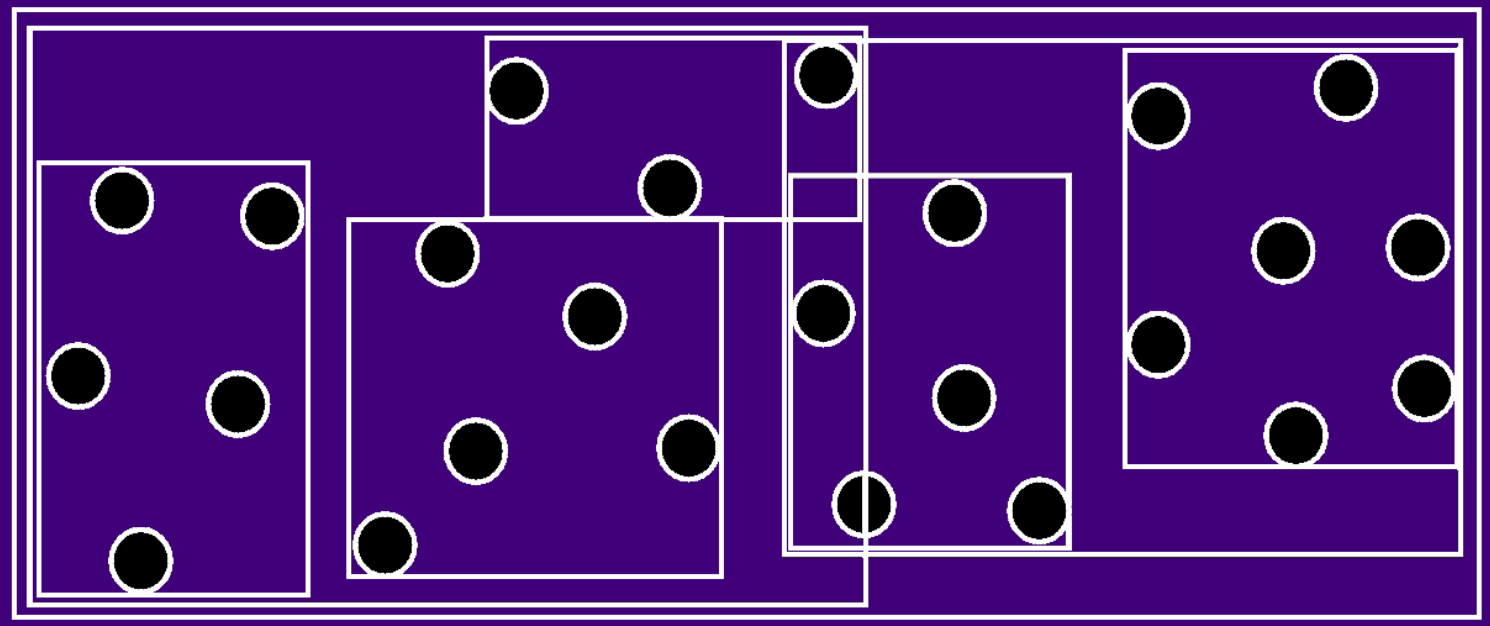
- **Se folosesc bounding volumes ierarhice pentru a grupa obiectele.**



- **Cum grupăm box-urile?**
  - **Cele mai apropiate**
  - **Cele mai compacte (cum?)**

Hierarchical Bounding Volumes

- **Se folosesc bounding volumes ierarhice pentru a grupa obiectele.**



- **Provocare: obiecte mișcătoare...**
  - **Ierarhia trebuie actualizată eficient**

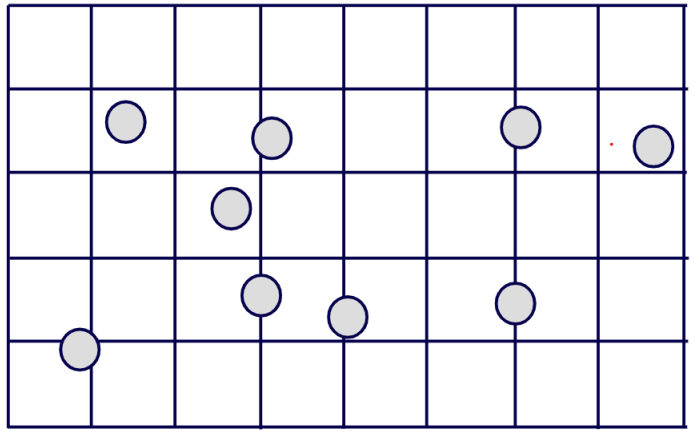
Subdivizare spațială STRUCTURI DE DATE

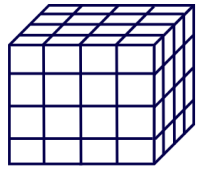
- **Subdivizează spațiul (bounding box-ul “lumii”)**
- **Ierarhice**
  - **Subdivizează fiecare sub-spațiu (sau orice sub-spațiu nevid)**
- **Multe metode**
  - **Grid, Octree, k-D tree, (BSP tree)**

Grid normal

Subdivizează spațiul sub forma unui grid dreptunghiular:

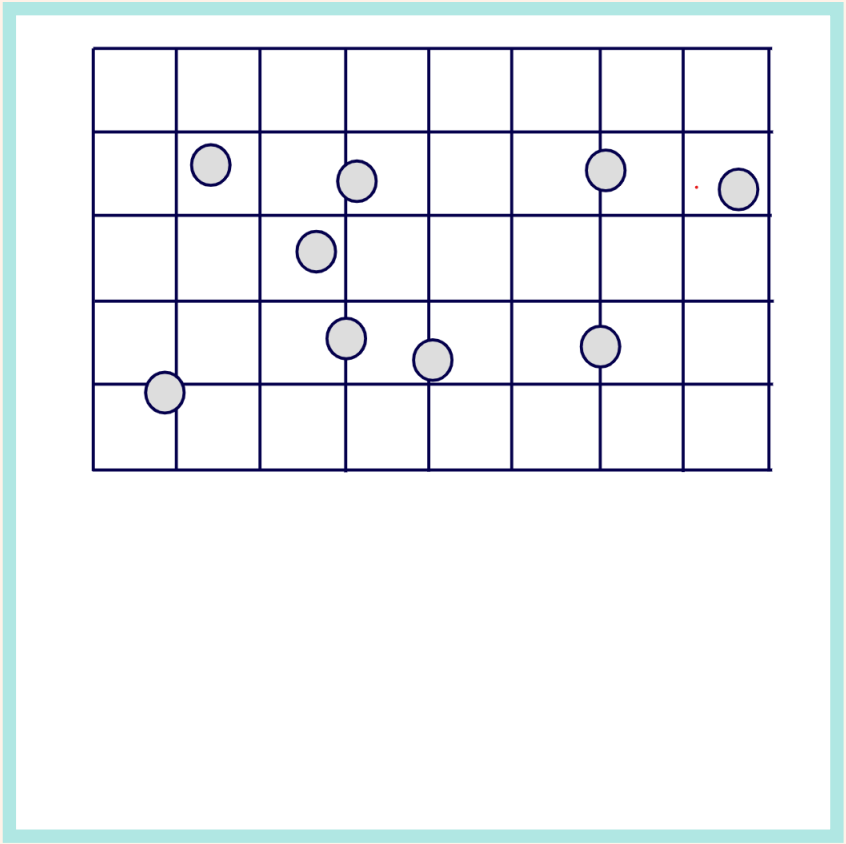
- **Asociază fiecărui obiect celula în care se află**
- **Verifică coliziunile doar dacă celulele se suprapun**





In 3D: regular grid of cubes (voxels):

Crearea unui grid normal



Pași:

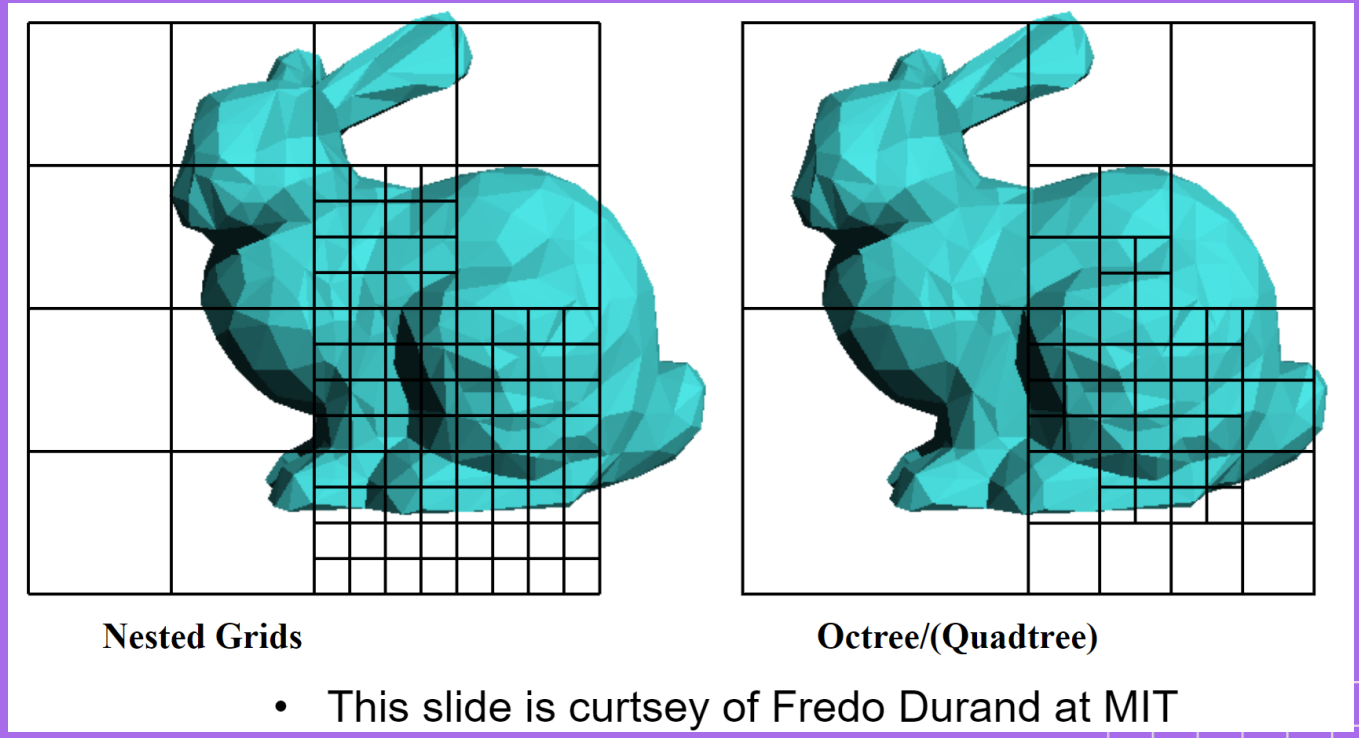
- Găsește bounding box-ul scenei
- Alege rezoluția grid-ului pe fiecare axă (x, y și z)
- Obiectele care ating mai multe celule sunt referențiate de toate celulele pe care le ating.

Discuție grid normal

- ***Avantaje?***
  - **Ușor de construit**
  - **Ușor de parcurs**
- ***Dezavantaje?***
  - **Poate fii rar**
  - **Obiectele se pot aduna doar în anumite regiuni (celulele conțin multe obiecte)**

Grid-uri adaptive

- **Subdivizează până când fiecare celulă conține mai puțin de n elemente, sau până când o distanță maximă d este atinsă.**



Bibliografie/ Resurse

CPSC 427 VIDEO GAME PROGRAMMING
https://www.cs.ubc.ca/~rhodin/2021_2022_CPSC_427/

http://www.realtimerendering.com/intersections.html

Curs Algoritmi Avansați