Detecție de coliziuni

Tema 1

Responsabili: Rareș Tăerel și Ștefan Rușeți

1 Introducere

Scopul acestei teme este de a vă familiariza cu programarea orientată pe obiecte și cu limbajul Java. În același timp, aveți ocazia de a vă familiariza cu o structură de date folosită în practică.

Detectarea coliziunilor este o componentă esențială a jocurilor video. Atât pentru jocurile 2D cât și pentru cele 3D, este foarte important detectarea faptului că două obiecte se află în coliziune, viteza de rulare a algoritmilor generând un impact major în cadrul performanței generale a jocului.

2 Detectarea coliziunilor în spațiul 2D

Presupunând că există 100 de obiecte care trebuie verificate pentru coliziune. Comparând fiecare pereche de obiecte, rezultă un total de 10000 de operații. O soluție pentru creșterea vitezei este scăderea numărului de comparații. Pentru aceasta, este utilizată o structură de date numită Quadtree.

Un Quadtree este o structură de date, utilizată pentru diviziunea unei regiuni 2D în mai multe zone de dimensiune mai mică. Acesta este asemănător cu arborele binar, având patru copii în loc de doi.

3 Cerințe specifice

Se consideră coliziune între două figuri geometrice atunci când dreptunghiurile încadratoare se intersectează. Pentru detectarea coliziunilor a două figuri geometrice în spațiul 2D se va implementa un Quadtree pe baza căruia se vor realiza următoarele operații:

- Inserarea unor figuri geometrice (Figura 1 și Figura 2)

  În cazul în care într-un cadran, sunt mai mult de două figuri sau părți din acestea, se va diviza în 4 subcadrane de dimensiune egale (după cum de observă în Figura 1) iar

figurile se vor insera în funcție de apartenența geometrică a dreptunghiului încadrator la aceste cadrane (Figura 3). ATENȚIE! Pot exista cazuri speciale, cum ar fi o figură în interiorul altei figuri, caz în care diviziunea nu mai poate fi realizată.

- Ștergerea unor figuri geometrice (Figura 2)

Se caută figura în arbore până se ajunge la frunzele care o conțin. Se începe recursiv din aceste frunze și dacă un nod părinte are numai frunze ce conțin o singură figură geometrică de același tip (o singură frunză sau toate), cadranele se vor șterge, părintele transformându-se în frunză.

- Testarea de coliziune

Se parcurge arborele pornind din rădăcină, realizând teste similare cu cele de la inserare returnându-se id-ul sau id-urile figurilor cu care intră în coliziune pe baza dreptunghiului încadrator (Figura 3).

- Testarea apartenenței unui punct la o figură geometrică

Similar cu testarea de coliziune, cu precizarea că testarea apartenenței se face diferit pentru fiecare tip de figură (deci nu pe baza dreptunghiului încadrator).

Atenție! Figurile din frunză nu se intersectează neapărat cu punctul / figura. Pentru fiecare figură găsită, trebuie testată separat intersecția.

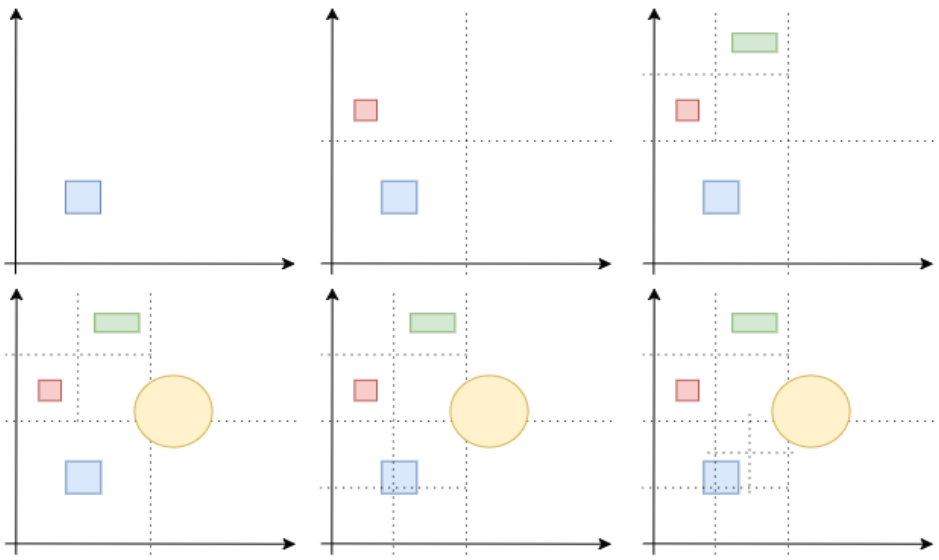

Figura 1

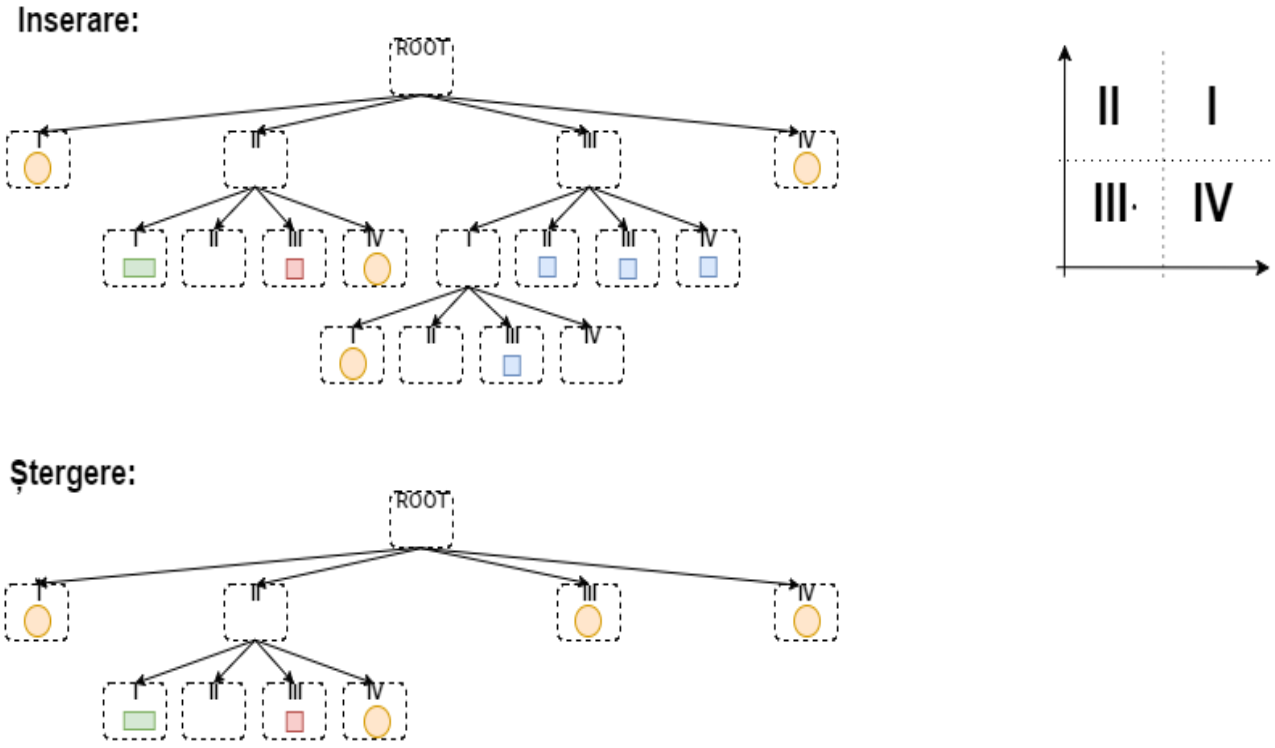

Figura 2

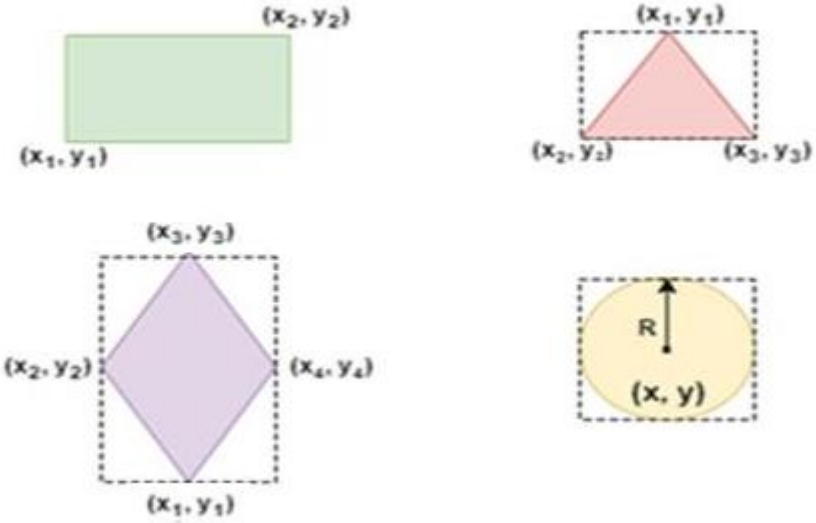

Figura 3

Tema va fi testată pe **VMChecker** (va apărea în curând), astfel încât va trebui să aveți și un **makefile** cu o regulă **„run”** care să ruleze clasa și care conține metoda **„main”**, pentru a ușura testarea automată.

4 Testare

Fișierul de intrare se va numi *quadtree.in* iar cel de ieșire *quadtree.out.* Prima linie va conține dimensiunea ecranului, sub forma a 4 numere separate prin spațiu (x_{min} y_{min} x_{max} y_{max}). Următoarele linii vor conține câte o comandă pe linie, în formatul definit mai jos.

quadtree.in	
11 1 ID x_1 y_1 x_2 y_2 11 2 ID x_1 y_1 x_2 y_2 x_3 y_3 11 3 ID R x y 11 4 ID x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4	- Inserare dreptunghi - Inserare triunghi - Inserare cerc - Inserare romb
22 ID	- Ștergere **Atenție!** Este recomandată folosirea unui vector pentru maparea între ID și figura geometrică
33 x y	- Detectarea de coliziune dintre punct și figurile din arbore
44 1 x_1 y_1 x_2 y_2 44 2 x_1 y_1 x_2 y_2 x_3 y_3 44 3 R x y 44 4 x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4	- Detectarea de coliziune dintre o figură și figurile din arbore

Fișierul de ieșire va conține câte o linie pentru fiecare interogare de intersecție din fișierul de intrare. Această linie va conține id-urile figurilor găsite, sortate crescător, separate printr-un spațiu, sau cuvântul *NIL*, dacă nu există nicio figură.

Tipul de element grafic

- **1** – Dreptunghi
- **2** – Triunghi
- **3** – Cerc

- **4 –** Romb

În Quadtree nu se vor insera puncte!

ID – Id-ul după care va fi identificată ulterior figura geometrică

x y x1 y1 etc. – Punctele ce reprezintă figura geometrică

Id-ul operației ce va fi efectuată

- **11** – Operația de inserare în Quadtree
- **22** – Operația de ștergere a unei figuri geometrice cu id-ul **ID** din Quadtree
- **33** – Operația de detectare a unei coliziuni a punctului (x, y) cu figurile din Quadtree
- **44** – Operația de detectare a unei coliziuni a figurii **T_FIG** definite de punctele **x y x1 y1 etc.** cu cele din Quadtree

Exemplu:

quadtree.in	***quadtree.out***
0 0 18 18 11 1 111 3 3 6 6 11 1 112 2 10 4 12 11 1 113 5 15 8 16 11 3 114 2.5 11 9 33 5 4 33 19 1 44 1 5 15 8 16	**111** **NIL** **113**

5 Constrângeri

- Numărul maxim de figuri nu va depăși 100000.
- Elementele din Quadtree trebuie să fie generice! Este interzisă apariția unor clase de genul Circle, Square în cadrul elementelor Quadtree-ului.
- Pentru realizarea temei trebuie folosită o versiune de Java nu mai recentă de Java 7, aceasta fiind versiunea care rulează pe VMChecker.
- Limita de timp pentru fiecare test în parte va fi publicată mai târziu, odată cu postarea temei pe VMChecker

6 Punctaj

Punctajul pe tema va consta din:

- 90% testare automată
- 10% JavaDoc
- -100% inspecția codului

7 Resurse

[1] Wikipedia Quadtree

[2] Definirea comentariilor în cod