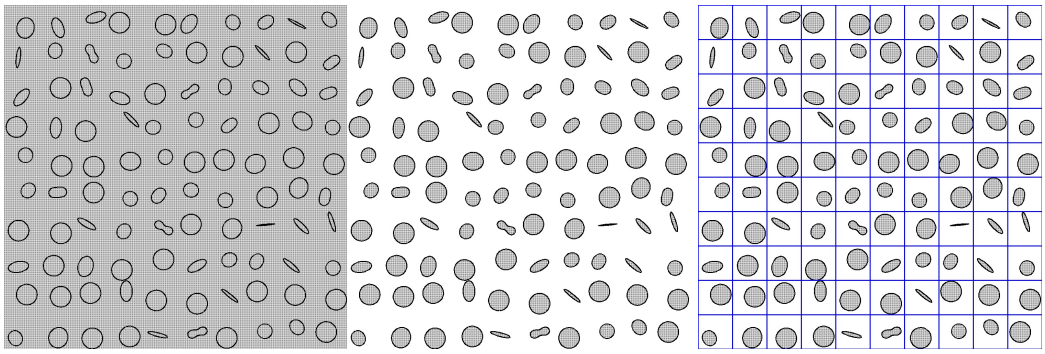
Tutorial 4 - Primitives subdivision with binary search trees

Computer Graphics

Kartic Subr and Martin Asenov

October 11, 2019

In this tutorial we explore a simplfied version of Bounding Volume Hierarchies (http://www.pbr-book.org/3ed-2018/Primitives_and_Intersection_Acceleration/Bounding_Volume Hierarchies.html) for ray intersection acceleration by using Self-Balancing Binary Search Trees. You have been given a set of random shapes (circles, elipses and Cassini ovals). You need to check whether the uniform samples accross the whole image are part of any of the shapes. You have been provided code, which already does this in a naive way - each sample is tested againt each shape.



(a) (b) (c)

Figure 1: Setup and overview of the task (a) Uniform samples across the whole image need to be checked whether they belong to any of the given shapes (b) Expected output (c) To make the problem easier each shape is in a seperate rectangle part of the image

You can compile the program by running:

```
g++ -I. -L. `pkg-config --cflags --libs opencv` subdivision.cpp selfbalance.cpp -std=c++11
```

You have been provided `subdivision.cpp` and `selfbalance.cpp` which contains most of the code to get you started. Implement and explore the following:

1. Search with binary trees - exploring the code, you will notice that we store the information about the shapes in a binary tree. Implement more efficient search by using the tree, rather than the naive way (Hint: the implementation should be no more than 3 lines of code)
2. Performance gains - you have probably already noticed that the program runs much faster now. Can you quantify what is the speed up - in terms of computation time; how many queries are performed (in the tree search and quering if a point is inside a shape)?
3. More realistic extensions

(a) Overlapping objects - how would you handle overlapping objects?

(b) Random positions - currently it is guaranteed that each object is inside one of the squares as seen in fig.1,c. What if that is not the case?