INTRODUCTION:

The quadtree is a spatial data structure with a hierarchical structure. It's a tree with each level corresponding to a further refinement of the space in question. Though quadtrees come in a variety of shapes and sizes, they can be used in a variety of ways. The concept can be applied to any dimension, and it is always a recursive subdivision of space that aids in the storage of information and provides the most vital or interesting details regarding space.In quadtrees, we begin by adding pointers to the it's root node, which defines all potential space. The node divides into four child nodes when the number of points in the node reaches a predetermined maximum capacity. When any of those nodes has reached its full point capacity, it splits into four child nodes, and so on. Quadtrees have a range of applications; from internet services handling millions of requests every second, image compression, handling geo-location services, searching for nodes in 2-D areas, collision detection and more. In this paper, our focus will be on reviewing its use in collision detection.Collision detection is an essential part of most video games. Both in 2D and 3D games, detecting when two objects have collided is important as poor collision detection can lead to some very interesting results. Many games require the use of collision detection algorithms to determine when two objects have collided, but these algorithms are often expensive operations and can greatly slow down a game. In this paper we will be addressing quadtrees, and how we can use them to speed up collision detection by skipping pairs of objects that are too far apart to collide. We′ll be writing a general purpose, scalable and re-usable quadtree library in Typescript and importing it in a visualization tool to depict its internal workings.