Goal implement the Boid algorithm on a 2d plane including obstacle avoidance.

What is a boid?

<https://www.red3d.com/cwr/boids/>

<https://www.red3d.com/cwr/papers/1987/boids.html>

The rules

Emergent behavior

Swarm robotics

A Boid is a one creature in a flocking algorithm developed by Craig W. Reynolds. These Boid attempt to replicate flocking behavior produced by animals like birds and fish. This complicated motion is produced with out any over head leading the group but will all participants acting individual.

The behavior of flocking shown by birds and schools of fish is very prevalent in nature. The flocking behavior is very difficult to describe

Flocking in nature is incredibly impressive. Each member of the flock independently moves, but in the end the whole flock seems to behave as one creature. This behavior displayed commonly in birds and fish seems from the outsides as incredibly complex, but can be described very simply. A Boid is an artificial animal that moves as a flock developed by Craig Reynolds in 1987. Each Boid makes independent movements with only information about nearby Boids.

The three rules every Boid follows are Alignment, Separation, and Cohesion. Alignment states that every Boid should try and move in the same direction as it’s neighbors. This causes the flock overall to move in the same direction. Separation states that each Boids should move away from neighbors that are close to them. This prevents any Boid from crowding their neighbors. Finally Cohesion states that each Boid should try and move to the center of their neighbors. This causes the Boids to move closer to each other. When each Boid follows these rules the emergent behavior of flocking occurs.

When in comes to obstacle avoidance a common approach is to have each Boid fell pushed away from obstacles. This however seem to create some very unnatural behavior as Boids may make a u-turn when approaching obstacles. To create a more natural behavior we will use a steer to avoid method where Boids with trace the edge of object with from their perspective and move to adjust their heading to be outside of the edge of the object. This create a more natural movement pattern and has the added benefit of being more realistic as actual animal would only be able to see the objects nearby and in front of them instead of just sensing all obstacles in the world.

In order to implement steer to avoid we need to develop a method for detecting collisions between our obstacles and rays. For this we will use ray marching. This algorithm slowly traces the path of a ray forward one step at a time. Each step we move the distance from our location to the closest objects edge. That way after each step we are sure that the current point is not inside of an object. If at any point the minimum distance reaches 0 we know that a object has been reached. This method of detecting the environment around the Boids allows for a more natural flying behavior.

In order to support the above described functions it is necessary in implement some supporting classes a vector, graphics interface, and shapes. Most calculations for both the ray marching and Boid algorithm are described as vector formulas so a powerful vector class will be needed to simplify calculation elsewhere. In order to display the information in a pleasing way we need a graphics interface. The Shape class will provide the data of object that the Boids will steer away from hitting.

The vector class has two important design goals, first we want to minimize memory usage, second loading functionary into the Vector class to simply algorithms. In order to improve the memory efficiency of the program we attempt to create a few new vectors as possible and focus on modify vectors with other vectors. Because we need to do these calculations hundreds of time per second creating new vectors after each operation would be inefficient. Second inorder to make it esay to implement the bold and ray marching algorithm easily it is necessary to load functionality into the vector class so complex operations can be done on a single line making the implementation of the algorithm easier.

The graphics interface needs to translate the data of all the objects in the scene on to a single image. The graphics package will provide an interface to allow the object to be drawable. This interface should provide a system for object to be described relative to a center point to allow for easy translations and that one object could require more than one primitive shape or color. It could also have feature to control the perspective of the scene zoom and centering.

The shape class is a convenient storage type for both drawing and mathematical operations. It should provide support for circles, rectangles aligned on the x and y axis and a generalized polygon. It needs to be able to define it’s edges for the ray tracing and it’s drawing instructions for the graphics interface. And allow for outside functions to edit it’s data. This should also come with a class that can read in formatted data for object to initialize them so they don’t need to be done by hand.

Finaly it is possible that the program could run into some efficiency trouble and therefore it is prudent to divide our functions in a way the multithreading could be an option. Most operations only relie’s on the information for the pervious frame. Our functions should attempt to separate functions into two parts first calulate what the next frame will look like and then change the data to match. In this way it is possible to multi thread both steps.

Algorithms for obstacle avoidance(obstacle as forces/edge detection)

Edge detection is harder

It makes more sense than a force

When you look a wall you don’t feel pushed away from it you just turn to get out side the edge

Our ray tracing circle strategy (typhoon algorithm)

Algorithm is called ray marching

Concentric circles

Steps in the algorithm

Note try and find a source

Efficiency concerns

Boids \* Boids

Boids \* shapes \* avg distances

Circle estimation may be faster

Drawing time

Some solution ideas

Multithreading in the dead frame

To be possible need to maintain independence in steps ie frame 0 only relies on info from frame -1

Vector math memory

Want to make a few new vectors as possible to save memory

Simple outline of class/packages

Basic outline

Boid

Relies on vector

The object that calculates the boid algorithm

Vector

A class the acts as a vector to do calulations

Ray

relies on vector

interacts with shapes to implement the ray tracing

Shape

relies on vector drawable ray

the obstacles for the boids

could be circle rectangle polygon

Graphics

relies on vector

relies on drawable

a screen jframe that can draw subclasses of drawable

Drawable

Can discrip an object to the graphcis module

relies on vector

Xmlreadin

Used to inialize shape on the screen

relies on vector

relies on shape