Version 2.0.0

Project Dawn | Local Avoidance | Tool

Table of Contents

Overview

This package contains fast lightweight solution for local avoidance. It is developed with DOTS in mind, as result it takes advantage of Unity latest technology stack like SIMD mathematics, Jobs and Burst compiler.

Manual

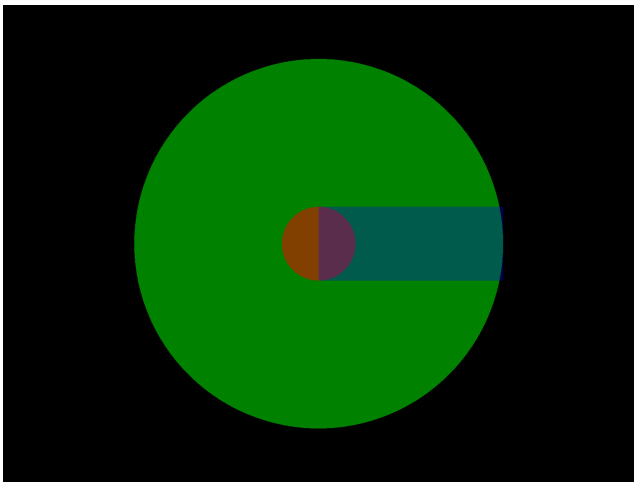
This local avoidance uses novel solution called `SonarAvoidance`. Idea is very simple. Interest point constructs sonar volume shaped in circle, that scans for nearby obstacles. Each obstacle cuts into the volume. Once all obstacles are scanned, the best direction can be calculated from it.

The solution is designed to be lightweight so it could be moved into any design. Thus the whole funcionality is contained of single struct called `SonarAvoidance`.

Constructor

Sonar can be created with simple constructor.

```
var sonar = new SonarAvoidance(position, direction, up, innerRadius, outerRadius, speed);
```



Position

Your agent position in world space.

Direction/Up or Rotation

Orentation of agent desired direction.

Inner Radius

Radius of agent.

Outer Radius

The maximum distance of obstacle.

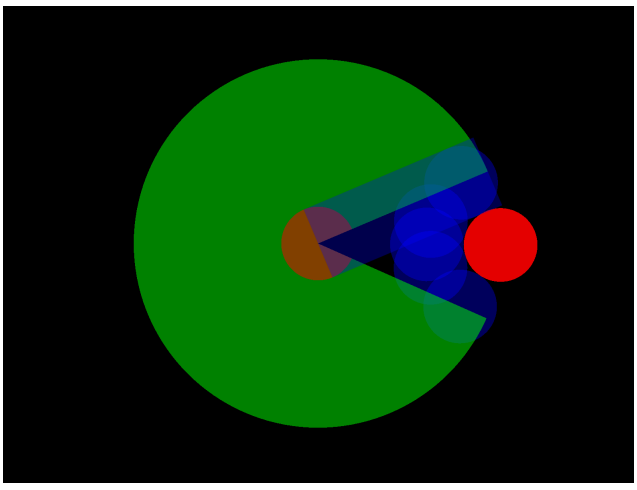
Speed

The current speed of agent.

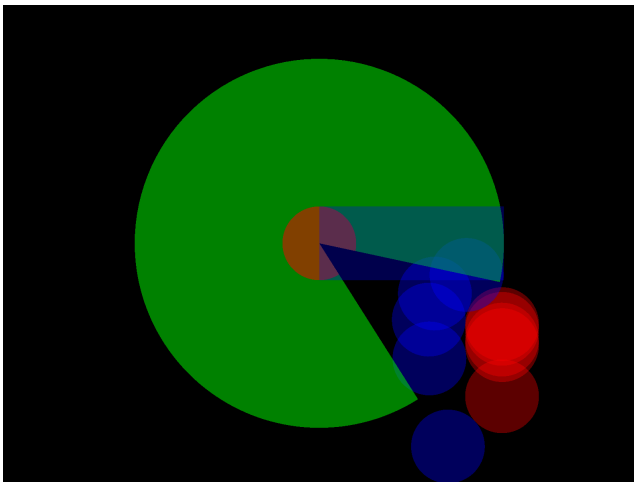
Circular Obstacle

Circular shape obstacles usually used for other agents.

```
bool success = sonar.InsertObstacle(obstaclePosition, obstacleVelocity, obstacleRadius);
```



Obstacle with 0 velocity



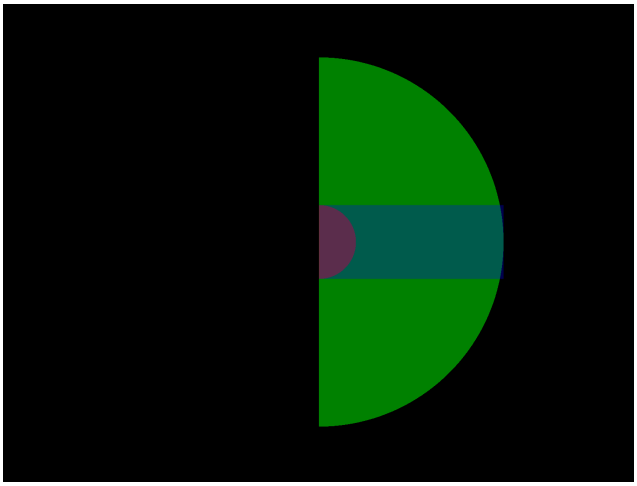
Obstacle with negative y axis velocity

> **Note:** Blue points shows where sonar would collide with collider red point at specific time. There can be multiple collision points.

Directional Obstacle

Directional shape obstacles usually used for limiting agent vision.

```
bool success = sonar.InsertObstacle(obstacleDirection, obstacleRadius);
```



With negative x direction and 180 degress

Cleanup

If you want to re-use the structure for next frame you can clear it.

```
sonar.Clear()
```

Or destroed with dispose interface.

```
sonar.Dispose();
```

Debug

There is few very handy methods for drawing sonar volume. These methods use gizmos so they must be called from MonoBehaviour.OnGizmos message and also editor only.

Drawing sonar shape

```
sonar.DrawSonar();
```

Drawing the closest direction with the inner radius

```
sonar.DrawClosestDirection();
```

Drawing obstacle and all its collision points

```
sonar.DrawObstacle(obstaclePosition, obstacleVelocity, obstacleRadius);
```

Samples

Samples contains few of possible implementations of agent navigation using `SonarAvoidance`. Keep in mind the code in samples are there just to show what API is capable of and give overall idea of its usage. For this reason code in there is not production ready and it is recommended to write your own implementation using `SonarAvoidance`.

Dependencies

- Tested with Unity 2019.4
- Package com.unity.mathematics@1.2
- Package com.unity.collections@0.9
- Package com.unity.burst@1.4

Support

If you have questions, bugs or feature requests use Discord.