Local Avoidance Techniques for Real-Time Crowd Simulation

Idris Miles
Bournemouth University SDAGE level I

May 4, 2015

Abstract

In this paper I demonstrate the use of Reciprocal Velocity Obstacle, RVO, as a viable local avoidance model for real time agent based crowd simulation.

1 Introduction

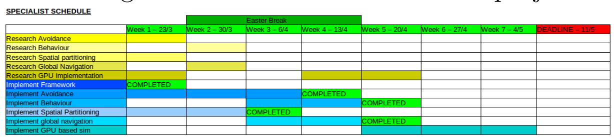
Crowd simulation is a growing niche for films in the animation and VFX world, where complete CG worlds are brought to life with their own CG inhabitants. Of course it would be too much to ask animators to animate the 300 pedestrians in the background, or the thousands of fans in a sports arena. So the use of crowd simulation tools were bought to life. There are many different aspects involved in crowd simulation: behaviour of the characters, also known as agents, global navigation of the agents, local obstacle avoidance, the animation system, so the characters don't glide around, physics if you want characters to react appropriately to various effects, the list goes on.

2 Local Avoidance in Crowd Simulation

Local avoidance is the avoidance of agents within a close proximity of each other, it is useful when there are multiple moving objects in the world that the global navigation system may ignore, due to the fact their position is constantly changing. Global navigation differs from local avoidance because it looks at finding a route from one location to another, while local avoidance just considers its surroundings and desired direction and aims to avoid collisions while staying on track to its goal. Artificial Intelligence also differs from local avoidance although the two can be linked together, AI focuses on creating behaviours of the agents such as forming groups or separating from other types of agent/objects in the virtual world. AI can indirectly incorporate local avoidance due to the behaviours it may create but often this isn't enough to satisfy a collision free system, hence local avoidance can be 'layered' on top. Boids flocking and social forces are examples of behaviour that can result in indirect local avoidance however they are not enough on their own to guarantee it. Reciprocal Velocity Obstacle, RVO, is not a behaviour but is purely a form of local avoidance. It is the RVO technique that is the main focus of my project.

3 Initial Research

The first call of action was to make a time schedule that I would follow in order to keep on track, as time management would be crucial in this project.

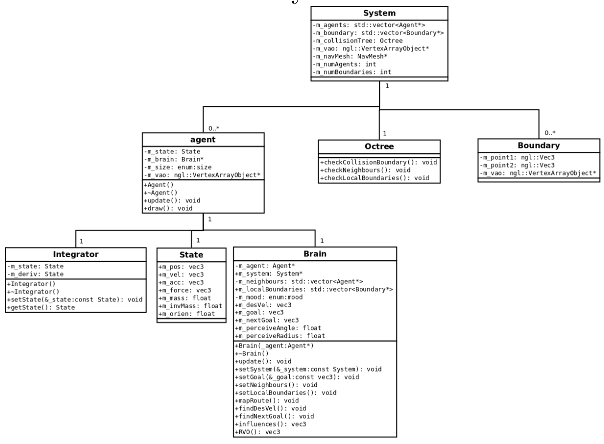


SPECIALIST SCHEDULE

		Easter Break						
	Week 1 – 23/3	Week 2 – 30/3	Week 3 – 6/4	Week 4 – 13/4	Week 5 – 20/4	Week 6 – 27/4	Week 7 – 4/5	DEADLINE – 11/5
Research Avoidance	🟨							
Research Behaviour		🟨						
Research Spatial partitioning	🟨							
Research Global Navigation		🟨						
Research GPU implementation	🟨			🟨	🟨			
Implement Framework	COMPLETED							
Implement Avoidance	🟦	🟦	🟦	COMPLETED				
Implement Behaviour			🟦	🟦	COMPLETED			
Implement Spatial Partitioning	🟦	🟦	COMPLETED					
Implement global navigation			🟦	🟦	COMPLETED			
Implement GPU based sim					🟦	🟦	🟦	

For my initial research I looked into the various methods used for local avoidance and what techniques current systems used. The three main methods I came across was Craig Reynolds Boids flocking system, Social forces and Reciprocal Velocity Obstacle. I felt flocking was not suitable as it was better for replicating animal behaviour such as birds rather than humans. Social forces is a very interesting technique that is better for producing human behaviours but not necessarily guaranteed object avoidance. Reciprocal Velocity

Obstacle, RVO, is a method not so focused on the behaviour of the agent but rather a method that aims to choose an appropriate velocity that will not result in a collision. RVO is a technique used in various systems including Goalem crowd plugin for Maya and Unreal Engine 4. After this initial research I decided to focus on the use of RVO as it seemed the most suitable for my project.
I then set out to create UML diagrams to see how my code should be structured and how the various components fit together, to do this I looked into design patterns. The use of aggregation seemed required as I planned to dynamically add agents and boundaries into the system.



4 RVO

RVO is an agent based method for collision avoidance and was developed by Jur Van Den Burg et all [5]. It is based upon a similar technique known as Velocity Obstacle (VO), and uses much of the same principle but with some critical tweaks, which was initially designed for motion planning of autonomous robots. However these techniques have been extended for use in crowd simulation.

4.1 How RVO Works

[1] [2] VO, which RVO is based from, works by the following:

$$VO^A_B(v_B) = \{v_A | (p_A, v_A v_B) BA = \} \text{ formula}$$

This translates to, the velocity obstacle $VO^A_B(vB)$ of agent B to A is the set of velocities V_A for A that will result in a collision with B moving at velocity V_B. The idea is to find a new velocity for the agent A that is outside the VO of B. The difference with RVO and VO is instead of finding a velocity outside the VO of B we find the average between the velocity outside and the current velocity. This produces smoother movement of the agent and guarantees oscillation free movement which can be a common problem for local avoidance.

4.2 RVO Implementation

pseudo code of some form can probably go here.

5 Optimizations

Changing the VO so not so conservative in dense areas. Using hash table and Moving heavy computation to a compute shader. [3] The first optimization I implemented was incorporating a hash table. The hash table speeds up the neighbour search considerably, rather than being of $O(n^2)$ it becomes $O(n)$ this is because each agent is put into a "bucket" that is associated with its position, the hash table consists of an array of these "buckets", and to find the agents neighbours you simply have to query the "bucket" the agent belongs to and the neighbouring "buckets". A further optimization is to have the agent only search half the necessary neighbouring "buckets" and when ever it finds an agent, have each agent add each other to their neighbour list. This means that half the number of neighbour queries have to be completed.
The next optimization was not for speed increase but rather refining RVO. During times when the crowd became dense the agents could become very conservative with their movement, this is due to the shape of the VO cone produced that designates the undesirable velocities. The tip of the cone can be cut so that we have 3 edges to check against rather than 2[2], although increasing the number of calculations it also increases the available velocities for the agent. Even still this is not enough at times, so we introduce a penalty factor for all velocities tested. This penalty is weighted upon the time to collision, smaller time = larger penalty, and also the difference between the tested velocity and desired velocity. The velocity with the small-

est penalty is chosen and is also multiplied by t, the time to collision, this means we can slow down the agent without having to sample too many velocities, only velocities of a different direction not magnitude. This is not perfect as it means some optimal velocities may be overshadowed however it is a quick method which is necessary for real-time applications.

6 Conclusion and Future Work

I am pleased with the outcome of the project as I feel I was successful in investigating local avoidance techniques focusing on real-time requirements. My implementation is proof off the success of the research.

References

[1] David Cherry. Rvo collision avoidance in unity 3d. 2013.

A students project to incorporate RVO avoidance into a Unity based crowd simulation.

[2] Stephen J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2009.

optimizations of existing RVO library, tweaks RVO implementation and describes parellizing implemenation.

[3] Wentong Cai. Nasri Bin Othman., Linbo Luo. and Michael Lees. Spatial indexing in agent-based crowd simulation. *6th International ICST Conference on Simulation Tools and Techniques pages 92-100*, 2013.

spatial partitioning

[4] Dinesh Manocha . Ming C. Lin Sujeong Kim ., Stephen J. Guy . Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. *12 Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Pages 55-62.*, 2012.

Use of stress modelling to drive agent motion in crowd simulation, taking into account psychological phenomena. Different stressors expressed as functions of space and time.

[5] Jur van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE, International Conference on Robotics and Automation (ICRA)*, 2008.

Velocity Obstacle and many other avoidance models inspired by VO explained and prooved.