Application of vector algebra and physics in designing steering behaviours of autonomous agents for realistic display in two dimensions*

Richard Čerňanský
Dominik Zaťovič

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xcernansky@stuba.sk
xzatovic@stuba.sk

November 3, 2022

Abstract

In this article we will focus on the problematic of animating steering behaviors of software-based autonomus agents in games. As we want the player to have the best realistic experience from the movement of the objects possible, we need to design their behavior according to the physics that describes it. The objective is to explain my ideas on how to calculate the steering force in the seek and arrive steering behavior and create an algorythm that animates such motion. The algorythm will be written in p5.js library of JavaScript language.

1 Introduction

Development in animation and games forces the creators to make the games more and more realistic. One of the most important aspects of realistic perception of a game is the movement of objects and the animation itself. Jonathan Cooper in the chapter **The 12 principles of animation in video games** [2] of his book mentions 12 basic principles of a well-animated game. The sixth one named Slow In and Slow Out says:*"Objects that burst into full speed immediately can look weightless and unrealistic, so it is here again that there is conflict between the gameplay desire to give objects ability to move immediately versus the artistic desire to give weight to a character."* That is why I came up with some ideas on how to design steering motion animations that look as realistic as possible.

In this article, we will call the objects autonomous agents. More about them will be explained in part 2.1. After understanding the concept of autonomous agents, we will state a problem in part 3.2. In part 4 we will have a closer look at the problem and derive some formulas necessary for the simulation. Finally, we will simulate the steering behavior, stated as a problem in p. 3.2 by writing the code in p5.js (in p. **??**).

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2022/23, vedenie: Mirwais Ahmadzai

2 Autonomous agents

2.1 Definition of autonomous agents

The autonomous agents generally refer to an entity that chooses how to act in its environment without any influence from a global plan or a leader. In games, these agents are not controlled by a player, but they are important part of the game because their action can significantly influence the flow of the game. For example, a villain who runs away from police decides on his own that he wants to escape and starts an action. There are three key components of autonomous agents we want to keep in mind [4].

1. An autonomous agent has limited ability to perceive its environment. It makes sense that if autonomous agent must decide on its action, it should be somehow aware of the environment it is located in. The question here is how much limited the ability is. If we wanted the object to be all-knowing creature aware of everything else around it, we need to give it an access to information about everything. On the other side if we wanted it to have just very narrow view of the environment, let's say just a few pixels around it.

2. An autonomous agent processes the information from its environment and calculates an action. The action is represented as a force that influences the object. For example, a police officer sees a thief and is attracted to him. The attraction is represented by a force pointing towards his location.

3. An autonomous agent should have no leader. This is not something that defines every autonomous agent. Sometimes you need to state some global rules that it must follow, but mostly you want the object to decide on its own[1], calculate its own actions.

2.2 Types of behaviors according to the number of objects involved

Craig Raynolds in his paper from Game Developers Conference **Steering Behaviors For Autonomous Characters** [3] introduces some types of behaviors that could appear talking about autonomous agents and how they behave. They divide into two main groups:

- Simple behavior for individuals and pairs
  Containing only one or two autonomous agents.
- Combined behaviors and groups
  Containing more than two autonomous agents.

In this article we will focus on the first mentioned because the combined behaviors are just more complicated kinds but the basic ideas are derived from the simple behaviors for individuals and groups.

3 Problem statement

Now as we have explained what it takes for the object to behave like an autonomous agent we should understand more specific self-operating concept of a vehicle to be able to grasp on a problem we state later in this chapter.

[1] however, the designer can include some specific atributes if needed

3.1 Braitenberg vehicle

Braitenberg vehicle is an entity that is a hypothetical self-operating machine that can make decisions about how to behave in an environment based on its sense perception. Valentino Braitenberg explains his concept in the book **Vehicles** [1]. Here is an example of that type of vehicle from the book:

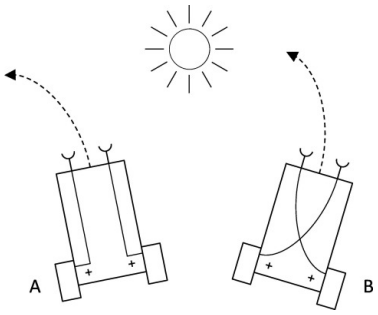

Figure 1

Vehicle A steers away from the light source and vehicle B steers towards the light source. It is not just about steering away or towards the sun. We could say that these vehicles feel emotion about the object. Vehicle A feels fear from the sun and vehicle B is attracted to it. We want the term vehicle to be clear because in the paper when we mention vehicle, we will be referring to the concept of Braitenberg vehicle and its characteristics.

3.2 Problem to solve

Our goal is to animate simple steering behavior. The best way to represent it is by steering a car. It is also a frequently occured situation in many games to steer self-controlled car realistically. Let's simulate a simple behavior for a pair (p. 2.2). The car that is expected to arrive to a specific position (f. e. a parking lot). We could say that in the car there is a driver or a device that decides on its own. It perceives the environment with its "eyes" and sees the location of the parking lot. Processes its distance from the place and calculates and action 2.1. It starts **steering** towards the parking lot (attraction towards the parking lot is based on the same emotional principle as vehicle B in part 3.1). Notice the word steering here which is very important. We do not want to simulate an unrealistically moving car that is immediately able to turn around and head towards the target in a maximum speed. We want the animation to be smooth and realistically looking. What rules do we have to follow if we want to simulate a situation like this?

4 Seek and arrive – a pursuit of a static target

In the language of steering behaviors, the problem can be translated as a designing a seek and arrival behavior on one object and a target. If we want to write an algorythm we need to come up with some formulas to calculate the motion.

4.1 Simple vehicle model

To describe an agent using physics, it is important to have some mathematical variables that describe it. Craig Raynolds in his paper [3] presents a simple vehicle model which

needs to be introduced. There are six attributes that our vehicle will possess.

Simple Vehicle Model: {mass- scalar; position- vector; velocity- vector; maxForce - scalar; maxSpeed- scalar; orientation- N basis vectors}

4.2 Characteristics of seek and arrive

4.2.1 Seek

Seeking steers a character towards a specified position in a space. This behavior aligns the vector of a velocity towards the target. But how do we calculate the steering force if we do not want the vehicle to steer completely right after seeing the target? We can diagram our problem from 3.2.

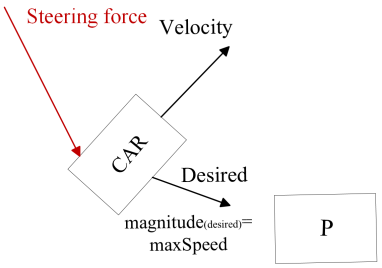

Figure 2

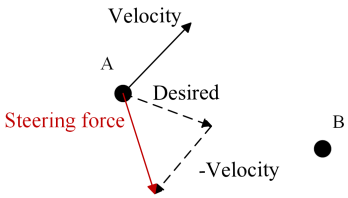

Figure 3

The steering force is then calculated with this formula * as shown in the Figure 3.

$$desired = positionA - positionB \quad (vectors)$$
$$*steeringforce = desired–velocity$$

As we are designing the animation, the steering force is updated with each frame and having lower effect next time every time it is applied. The vector of velocity is being aligned with the desired one more and more with every frame.

4.2.2 Arrive

When we want to simulate arrival, we must think about gradually decreasing the speed of the vehicle and eventually stop it once the vehicle passes some imaginary boundary. The following diagram describes the situation.

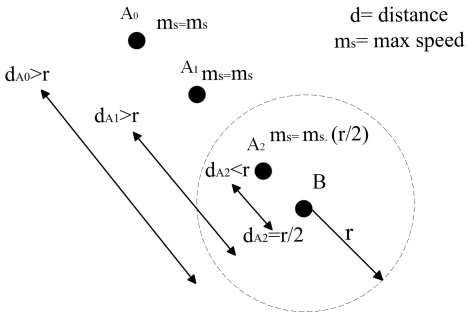

Figure 4

From the Figure 4 we can assume two things:

$$d > r \Rightarrow speed = maxSpeed$$
$$d \leq r \Rightarrow speed = (d/r).maxSpeed$$

From the second equation we can see that once the object passes through the circle of the radius r, the speed is decreasing in a fraction of the distance d and radius r. Once we have the new and **lower** speed calculated, then the magnitude of a vector of desired velocity is set to the size of that lower max speed. The closer to the place of stopping, the lower the speed is.

5 Simulating the behavior in p5.js

The objective of this project was to simulate the behavior. This chapter briefly explains the tool that was used and describes the main parts of the code.
For the link to the source code of the animation **click here**.

5.1 What is p5.js?

p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, or educators. It is a collection of pre-written code and provides us with tools that simplify the process of creating interactive visuals with code in the web browser. p5.js is free and open source. We will use it for creating the animation from 3.2.

5.2 Main part of the code - class Vehicle and its functions

5.2.1 constructor() function

This function creates all the variables for values that were stated in the chapter 2.2. As an input it gets updated x and y position of the vehicle.

5.2.2 seek() function

Seek function gets as an imput vector Target representing x and y position of the target. First, it calculates the steering force by subtracting velocity vector from desired velocity vector (4.2.1). An if statement included in a function checks if the distance is lower than 100 pixels. If yes, then it sets the magnitude of the desired velocity to some portion of max speed accordingly to the distance. Otherwise, it sets the magnitude to the size of max speed.

5.2.3 update() function

Update function applies the steering force combined with velocity to the vehicle every frame. The result is our realistically looking steering animation.

6 Research method and literature review

The source article [3] is what introduced me to the topic. From the beginning of my research there were a lot of questions. The idea of autonomus agent was mentioned in Raynold's article, but to completely understand it I had to search deeper. A part of the book Vehicles [1] and the article from Verhangen [4] gave me the complete understanding.

Raynold's article is what helped me the most with coming up with formulas for calculation of steering force. The explanation of his ideas is clear and understandable for anyone with a knowledge of coordinate geometry and physics of forces.

7 Results

8 Conclusion

Acknowledgement

I would like to thank my friend Lukáš Častven for introducing the problem to me and arousing my interest in the topic...

References

[1] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. Bradford Books, 1986.

[2] Jonathan Cooper. *Game Anim: Video Game Animation Explained*. A K Peters/CRC Press, 2019.

[3] Craig Reynolds. Steering behaviors for autonomous characters. `https://www.red3d.com/cwr/steer/gdc99/`.

[4] Henricus JE Verhagen. *Norm autonomous agents*. PhD thesis, Stockholm Universitet, 2000.