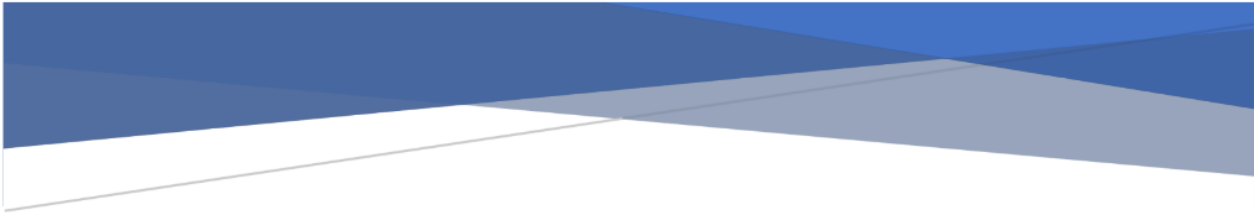


GamesArchitecture_MartinDunn_ 1705105

by Martin Dunn

FILE	4080_MARTIN_DUNN_GAMESARCHITECTURE_MARTINDUNN_1705105_2 34537_1408208470.DOCX (203.19K)		
TIME SUBMITTED	07-FEB-2020 05:13PM (UTC+0000)	WORD COUNT	1271
SUBMISSION ID	119768031	CHARACTER COUNT	6278



GAMES ARCHITECTURE AND MECHANICS

ACCA5017

Martin Dunn (1705105)
1705105@student.uwtsd.ac.uk

Contents

Introduction	2
Euler's Method.....	2
Vector Class.....	4
References	5

Introduction

This report documents different aspects of games architecture and game mechanics briefly. Euler's method is a large focus for the most part including what it is, how it works and its accuracy. A C++ implementation has been created to use Euler's in a practical application to approximate the velocity, acceleration and position of a spring powered model dragster as each timestep of 0.03 seconds. The pros and cons of this method are also addressed to outline where Euler's may not be a sensible solution for an equation that needs to be approximated.

Some light coding involving vectors has also been included to implement essential functions that would be needed within a game engine, the two main functions implemented in this work is distance between points and collision detection of two bounding sphere volumes.

Euler's Method

Euler's method is an iterative mathematical method used to solve differential equations that was created by an 18th century mathematician who is widely regarded as the best mathematician of all time [1]. It is used to solve differential equations where an initial position and velocity is given. This is the simplest iteration method to be used but can work very well in certain scenarios. One major downfall of this method is as time goes on it becomes less and less accurate, this can be countered by using a very small time-step (h). Although this helps improve the accuracy of the approximation it is not a viable solution as the number of steps needed to be taken takes a lot more computing power to execute them all individually [2]. As with all methods it has its advantages and disadvantages.

Pros	Cons
Direct	Limited time usage
Simple	Adds energy
	A lot of computing power for accuracy
	Less accurate than other methods

[3]

Using this understanding of Euler's method, a C++ program has been created to determine acceleration, position and velocity of a spring power dragster car at each time step of 0.06 seconds. The results have been plotted onto a graph to better show the outcomes of the simulation.

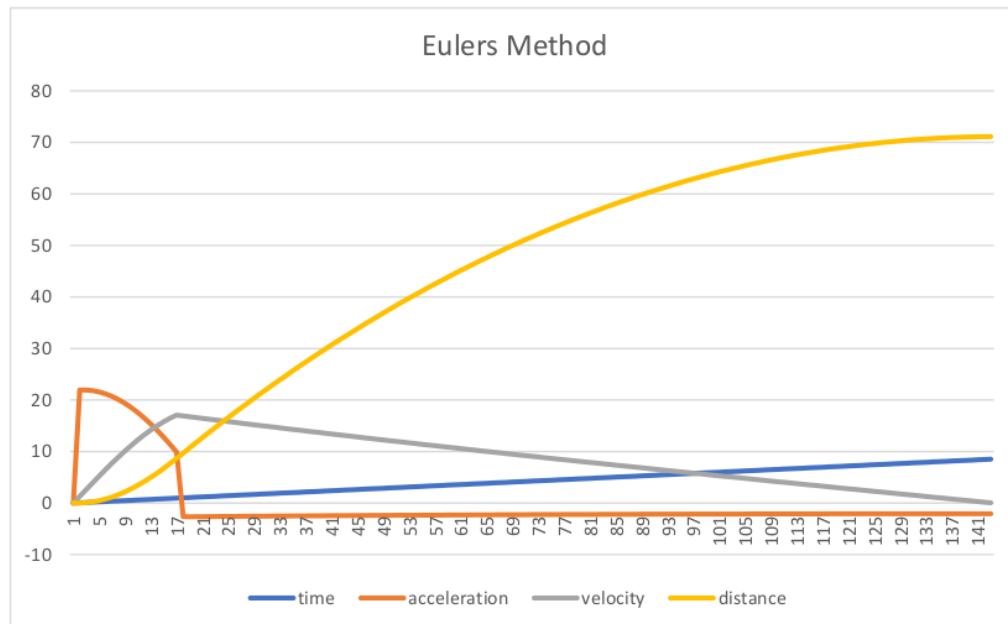


Fig 1 Euler's method front area = 0.01

The graph shows a reasonably accurate approximation of how the dragster simulation should perform if it were to happen in the real world. For example, The acceleration instantly shoots to a high value, as the spring is fully loaded ready to release all of its energy into the dragster, then as the spring extends and disconnects from the dragster it begins to decrease after around 8 metres as this is when the tractive force comes to an end. We then see the acceleration drop below 0, this tells us that the car is decelerating and slowing the vehicle down. We know this is accurate by comparing the acceleration to the velocity, the velocity continues to rise rapidly whilst there is a positive acceleration force on the vehicle. The moment the deceleration begins the velocity continuously drops over time until inevitably the velocity and the acceleration should both levels out at 0 at the same point on the graph, if it were to be extended.

The distance also fits in with the model and time shown for the car to stop. This is shown as the line begins to level out at the end, this indicates that there is no more distance being covered, therefore the vehicle has stopped. This should also level out at the same point the velocity and acceleration hit 0. If we were to extend this graph for a few more seconds, we would see the time continuing to rise at the same rate, as we would be using the same time step, but both the acceleration and velocity would be continuously flat at 0 and the distance would also be flat at the total distance travelled before the velocity and acceleration hit 0.

We know that this is an accurate simulation of a spring powered dragster as it is performing in the way that would be loosely expected in a real-world test. We can also test this further by changing the values in the formula to see how the approximation reacts. In the next test the front area has been drastically increased from 0.01m^2 to 1m^2 . Using our basic knowledge of forces, we should see

the vehicle travel for less time and loose velocity much faster as the bigger the area, the more drag force will be acting upon the surface.

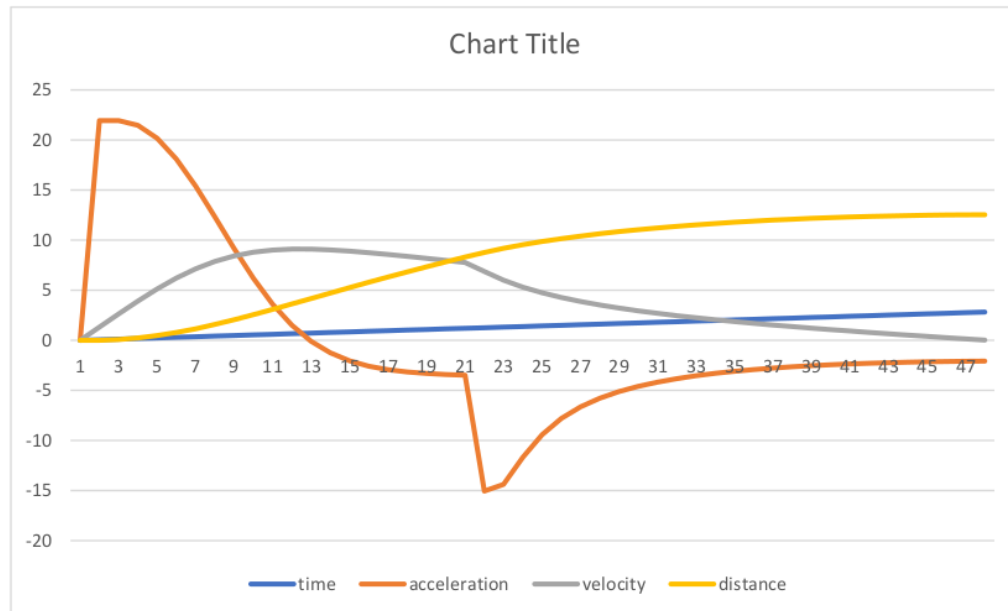


Fig 1.1 Euler's method front area = 1

As we can see when the area is increased it has large affect that we would expect to happen. We can see that the acceleration still begins to rise rapidly as explained before, but we can also see it decelerate at a much earlier point and much faster then previously, even before the 8 metre mark is reached the car is already decelerating due to the amount of resistance on the large area. Once the tractive force at 8 metres is no longer applied, the car very quickly decelerates, and this effect can also be seen at the velocity at the same point. Then as before both the velocity decreases and the acceleration increases until the car comes to a stop at 0, the same point at which the distance levels out.

As we can see with the model dragster simulation, Euler's method can be very useful for short, direct and simple scenarios to save time and computing power to get an approximation that is close to the numerical answer.

Vector Class

Using the guidelines provided a cVector.cpp and cVector.h were provided to have the basic functionality of a vector class within a game engine. It was then tasked to add and additional method to the class responsible for calculating the distance between two points in 3D space. The simple equation used to calculate the distance between points in either 2D or 3D space is Pythagoras' theorem.

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2}$$

Using the above equation, we are presented with a single floating-point number that is the distance between point one and point two in the equation. The equation shows that to find the distance between the two points we must first minus each value on the x, y and z axis of point two from point 1, the result of these must be squared then added together. We then take the square root of the final addition; this becomes our distance between the points. This has been implemented using C++ and tested using the test schedule below (Answered rounded to 1 decimal place).

Test Number	Description	Expected Result	Actual result	Pass/Fail
1	Negative values P1 = {-5, -10, -34} P2 = {-12, -37, -6}	39.5	39.5	pass
2	Positive values P1 = {6,8,24} P2 = {5,6,3}	21.1	21.1	pass
3	Negative and positive values P1 = {-5,-17,-23} P2 = {7,6,9}	41.2	41.2	pass
4	Large numbers P1 = {1006,3006,3569} P2 = {3258, 10253, 20158}	18242.4	18242.4	pass

The vector class, and more importantly, the above function that has been created were used to implement collision detection with two sphere bounding volumes, using the distance between the centre of both spheres and comparing this to the sum of the radii. If it is less than or equal to the radii there is a collision, if it is greater than there is no collision.

References

- [1] The Story Of Mathematics, "The Story Of Mathematics," [Online]. Available: https://www.storyofmathematics.com/18th_euler.html. [Accessed 04 02 2020].
- [2] jdickinsongames, "jdickinsongames," 22 01 2015. [Online]. Available: <https://jdickinsongames.wordpress.com/2015/01/22/numerical-integration-in-games-development-2/>. [Accessed 05 02 2020].
- [3] D. A. Geletu, "tu-ilmenau," Ilmenau University of Technology, 2011/12. [Online]. Available: https://www.tu-ilmenau.de/fileadmin/media/simulation/Lehre/div/Lec_Slides3.pdf. [Accessed 04 02 2020].

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

68/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

GRADING FORM: SMU GENERIC_GJ

**MARTIN
DUNN**

68

RESEARCH-INFORMED LITERATURE



Refs included and used appropriately. Use established texts and not on-line refs! Key areas covered.

5.5

KNOWLEDGE AND UNDERSTANDING



Essential topics covered. Light in places. Good discussion of physical properties of the car and how they affect speed and distance. Test results and output of code appropriate.

13

ANALYSIS



Results provided for a range of differing vehicle properties - well done.

16

PRACTICAL APPLICATION AND DEP.



Code covers tasks - Euler attempted. Results appropriate.. Separate .h and .cpp functions. Euler coded in main - use a class. Test results could be more detailed.

28

SKILLS FOR PROFESSIONAL PRAC.



Report strucutred. Could be more detailed in places. Refs on-line - use established texts such as Bishop and van Verth.

5.5