

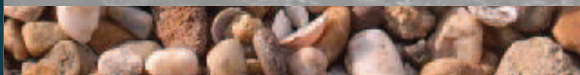
Duna-kavics

A Dunaújvárosi Főiskola online folyóirata 2015. III. évfolyam XII. szám

Műszaki-, Informatikai és Társadalomtudományok

STÉGER CSILLA

A tanulási környezetek átalakulása – Hazai és nemzetközi irodalmi áttekintés (2. rész)

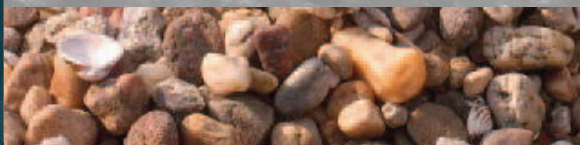


PETER KUNGL–JÓZSEF KATONA
Trajectory calculation using modified Euler method, presented in a platform game



HORVÁTHNÉ FÁBIÁN MYRTILL

A népességfogyás és előregedés hatásai regionális és települési szinten



Dunakavics

A Dunaújvárosi Főiskola online folyóirata 2015. III. évfolyam XII. szám

Műszaki-, Informatikai és Társadalomtudományok

MEGJELENIK ÉVENTE 12 ALKALOMMAL

SZERKESZTŐBIZOTTSÁG

András István, Király Zoltán, Kukorelli Katalin, Palotás Béla,
Rajcsányi-Molnár Mónika.

SZERKESZTŐSÉG

Ladányi Gábor (Műszaki)
Nagy Bálint (Informatika és matematika)
Szakács István (Gazdaság és társadalom)
Klucsik Gábor (technikai szerkesztő)

Felelős szerkesztő Németh István
Tördelés Duma Attila

Szerkesztőség és a kiadó címe 2400 Dunaújváros, Táncsics M. u. 1/a.

Kiadja DUF Press, a Dunaújvárosi Főiskola kiadója
Felelős kiadó Dr. habil András István, rektor

A lap megjelenését támogatta TÁMOP-4.2.3-12/1/KONV-2012-0051

„Tudományos eredmények elismerése és disszeminációja
a Dunaújvárosi Főiskolán”.

<http://dunakavics.duf.hu>

ISSN 2064-5007

Tartalom

STÉGER CSILLA

A tanulási környezetek átalakulása – Hazai és nemzetközi irodalmi áttekintés (2. rész)

5

PETER KUNGL–JÓZSEF KATONA

Trajectory calculation using modified Euler method, presented in a platform game

29

HORVÁTHNÉ FÁBIÁN MYRTILL

A népességfogyás és előregedés hatásai regionális és települési szinten

41

Galéria

(Fekete Fanni fotói)

69

Trajectory calculation using modified Euler method, presented in a platform game

Abstract: Nowadays most of the software developers have the knowledge and resources with the use of they are able to implement a simple computer game. We have to try for the minimizing of the counting demand in the case of computer games too such as implementing other applications. The article presents the execution of a simple martial, gunnery, simulation computer game where to the counting out the appropriate trajectory a modified Euler method was used with the purpose of the smaller counting demand. After the completion of the necessary calculations the game was implemented in high-level Java programming language.

Keywords: Euler method, Midpoint method, trajectory, software development, Java.

Introduction

During the development of a game, the designers need to think about lots of different elements that can be implemented to the game to make it more and more exciting. One of these features is the way we are fighting against our enemies in the game. In a triple a title where the budget is huge the world is fully 3D and dynamic the options are almost endless. But the reality is not this for most people who have the ambition of developing a game. [1, 2]

Most of us can only try and develop a way to simpler game, and usually that means 2D or 2.5D. (2.5D means a world where the game is designed in a 3D manner but the character can only move to two directions.) [1, 2] In these environments our planning-pool is way smaller. In this game the fight system can be very similar, and one of the ways of how to make ours different is creating a trajectory for our projectiles. This gives our game a

* *Dunaiújvárosi Főiskola,
Informatikai Intézet*
E-mail: peter.m.kungl@gmail.com

** *Dunaiújvárosi Főiskola,
Informatikai Intézet*
E-mail: katonaj@mail.duf.hu

[1] Jonathan F. Raper (1989): *The 3-dimensional geoscientific mapping and modelling system: a conceptual design, in Three dimensional applications in Geographic Information Systems*. Philadelphia: Taylor and Francis Inc. Pp. 11–19.

[2] Pile Jr. John (2013): *2D Graphics Programming for Games*. New York: CRC Press.

[3] Attila Kővári (2010): Real-Time Modeling of an Electrohydraulic Servo System. *Computational Intelligence in Engineering*. Springer. Pp. 301–311.

[4] Attila Kővári (2009): Hardwer-in-the-Loop Testing of an Electrohydraulic Servo System. *10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics*. CINTI 2009. Pp. 631–642.

[5] Kővári Attila (2015): Effect of Leakage in Electrohydraulic Servo Systems Based on Complex Nonlinear Mathematical Model and Experimental Results. *ACTA POLYTECHNICA HUNGARICA* 12. Pp. 129–146.

[6] Attila Kővári (2009): Influence of cylinder leakage on dynamic behavior of electrohydraulic servo system. *Proceedings of the IEEE SISY 2009: 7th International Symposium on Intelligent Systems and Informatics*. Subotica, Serbia. Pp. 375–379.

[7] Attila Kővári (2009): Hybrid Current Control Algorithm for Voltage Source Inverters. *Proceedings First IEEE Eastern European Conference on the Engineering of Computer-Based Systems*. Novi Sad, Serbia. Pp. 65–70.

realistic feeling to it and it widens our ways of making and outstanding in the game. New weapons with different types of shooting for example, new environments with different gravity and many more. The article will be about the implementing of a trajectory to a platformer game. For the calculations a modified Euler method was used.

Simulations

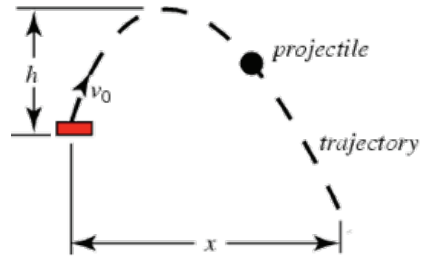
In this current state our goal was not to create a physically accurate environment, but to make it looks like that. Simulations are widely used in different areas of engineering [3, 4, 5, 6, 7], this game uses similar calculation methods to simulate the physical environment. During the game development we do not have to make everything according to the real world, there is no reason for doing that. What we rather want to achieve is an environment that makes the player believe that those things that we are showing him/her could really happen in real life. Since this is what we are aiming for, our calculations are very off, but it can be changed. It takes a lot of work, but with certain added rules and environmental objects the whole project can take a new turn and become something more serious, and turn into a simulating software for ballistic calculations.

New Game Features

Since our main build is a plat former game where certain calculations has been implemented, it is logical to stay in this path and make this game even more interesting. For this our new trajectory system gave a lot of options. Every game has to have a thing called addictive factor put into it to keep the gamers interested. These factors can be character levels, a very detailed story line, or upgrades. With this system we can make exciting things in upgrades segment. We could implement a character development cycle where as you are advancing in the game you get to unlock new kind of weaponry. If we are talking about a long game we can go from stone kind of a things like throwing

stone to the infinity and beyond, shooting lasers and all that kind of new age equipment. Of course every weapon should have new trajectory lines and different ballistic features, such as damage, armor penetration, attack speed etc.

Figure 1. Example of a trajectory.



Expectations, Requirements

What we need to keep in mind that during the development of the game the reality can be transformed such way to fit our needs. This means the solution that I have created not even remotely close to real world physics. For example in my little world the gravity is -0.01 N (newton).

All we can expect of our game is to look like as it was real life. Even though we live in a world where it is not a rarity to see mobile phones with as much as 3 GB of RAM, but still have to care about to optimization of our games. Since our current project will only feature one kind of projectile the performance really should not be a problem but during later improvements it is a thing to keep in mind. In this game, our goal is to achieve a state where if we press the ctrl button, our character shoots out a laser beam that drops accordingly to our calculations. Everything in the software has to be written in a way that leaves room for future upgrade. We need to keep that in mind when planning out our calculations and designs.

Figure 2. Software life cycle.



[8] Stoer, Josef–Bulirsch, Roland (2002):
Introduction to Numerical Analysis.
 (3rd ed.).
 Berlin– New York.

Planning

For the convenience let us assume that we already have a game that has every basic functions that we need. The planning process starts with the creation of the projectile class. First we need to initialize the keylistener, and write the object. I chose to shoot with the ctrl button. Restrictions have been put in place that prohibits the player to shoot while ducked or while the robot is jumping. To achieve this state we have used multiple logical variables. Yet another logical variable was used that assures a little time between shoots. This means the gun will work rather as a pistol not like a machine gun or some kind of a beam weapon. In our projectile.java class we are paying attention to two details. Let me start with the less important one. We have to handle the erasing of the already shot projectiles. Sooner or later it has to disappear. To insure this we have put three rules in place. First, if the X coordinate of the projectiles position goes over 800 then we have to erase it. The second one does the same for the Y coordinate, the game erases those projectiles that hit the ground. Here lies yet another potential for upgrade. For example if our character were to handle a bow, we could keep those arrows on the ground, and that would give the environment a bit of excitement and dynamism. Finally our third restriction goes with the collision detection. If we shoot an enemy our projectile must not go through the enemy only to disappear because of the earlier two things. If it hits the target it has to go.

Planning of the modified euler (midpoint method)

The Midpoint method has its name because we take a step in a given moment with not the valid derived but with the valid derived at the required half of the middle of the steps: the steepness of the wanted curve is chosen from the half way of the two ends of the step. [8] Therefore:

$$y(t + \Delta t) = y(t) + f\left(t + \frac{\Delta t}{2}, y\left(t + \frac{\Delta t}{2}\right)\right)\Delta t. \quad (1)$$

The only problem is that we do not know the value of the ‘f’ function in the moment of $t + \Delta t/2$, only in t moment. However we can approximate the value with the

vital Euler method. So the outcome:

$$y(t + \Delta t) = y(t) + f(t + \frac{\Delta t}{2}, y(t) + f(t, y(t)) \frac{\Delta t}{2}) \Delta t. \quad (2)$$

Mistake calculation:

We rewrite the previous term into simpler form similar to Taylor-line.

$$y(t + \Delta t) = y(t) + \frac{d y(t + \frac{\Delta t}{2})}{dt} \Delta t. \quad y(t + \frac{\Delta t}{2}) \quad (3)$$

Let us shell one after the other to primal order

$$y(t + \Delta t) = y(t) + \frac{d}{dt} \left(y(t) + \frac{d y(t)}{dt} \frac{\Delta t}{2} \right) \Delta t, \quad (4)$$

than open the term:

$$y(t + \Delta t) = y(t) + \frac{d y(t)}{dt} \Delta t + \frac{d^2 y(t)}{dt^2} \frac{(\Delta t)^2}{2} \quad (5)$$

Compared the term to the Taylor row shelling of the unknown function we can see that to the second rate the central method is exact, its mistake is third-rate. The advantage of the central method: It goes with more calculation like the vital Euler but it is second rate. We gain compared to the second-rate Euler that we do not have to calculate the second derived of the unknown function, it is enough to use the first derived so we can collectively say that it requires less calculation. The method can be already used in practice but it is not enough to serious calculation while where punctuality is not important it works mostly with acceptable stability. [9]

Since we are talking about a rather more difficult formula we have started to write it in the MATLAB software because there I could test myself instantly with the plot command. MATLAB is a widely used environment for several fields of engineering simulation for example mathematics, mechatronics, electronics or physics. The modification of the method means that we are not using the time variable that is given in the general Euler's method. [10]

[9] Lakoba, Taras I. (2012): Simple Euler method and its modifications (Lecture notes for MATH334, University of Vermont).

[10] Liangyu Peng–Lei Bao–Manchi Huang (2011): Application of Matlab/ Simulink Software. In: *Physics Communications in Computer and Information Science*. vol 163. Pp 140–146.

[11] Georges Gauthier Merx–Ronald J. Norman (2006): *Unified Software Engineering with Java*. P. 201.

[12]Griffiths, D. V.–Smith, I. M. (1991): *Numerical methods for engineers: a programming approach*. Boca Raton: CRC Press. P. 218.

Implementation

In this part we would like to talk about the most interesting part of the game design, and that is the implementation. It consists of three different parts. One is the shooting itself, then how I have created the script in MATLAB and finally the piece where the MATLAB script got translated to Java code. [11, 12]

A. First the shooting

The shooting consist of three separate pieces. First of all we have to handle the input. Our robot is going to shoot it is hyper-super-deadly laser blaster if we press the ctrl button. Here you can see the bit of code that handles this

```
public void keyPressed(KeyEvent e) {
    switch (e.getKeyCode()) {
        case KeyEvent.VK_CONTROL:
            if (robot.isDucked() == false
                && robot.isJumped() == false
                && robot.isReadyToFire()) {
                robot.shoot();
                robot.setReadyToFire(false);
            }
            break;
    }
}
```

As you can see the game does not allow shooting if the robot is ducked or is jumping at the moment. For this procedure we are using a very simple object:

```
public boolean isReadyToFire() {
    return readyToFire;
}
```

```
public void setReadyToFire(boolean readyToFire) {
    this.readyToFire = readyToFire;
}
```

```
}
```

So, if `isReadyToFire` is true we can call the object called `shoot` in the robot class.

```
public void shoot() {  
    if (readyToFire) {  
        Projectile p = new Projectile(centerX + 50, centerY - 25);  
        projectiles.add(p);  
    }  
}
```

All it does, it puts a new projectile to end of our gun which is located at the coordinates 50;25. The rest is being done in the `start` class.

```
ArrayList projectiles = robot.getProjectiles();  
for (int i = 0; i < projectiles.size(); i++) {  
    Projectile p = (Projectile) projectiles.get(i);  
    g.setColor(Color.YELLOW);  
    g.fillRect(p.getX(), p.getY(), 10, 5);  
}
```

Now this few lines handles the painting, the size, the color. And all this is being done with the help of an `ArrayList`. That arraylist is being filled accordingly to the input.

B. Creating the MATLAB script

Our goal with the MATLAB script is to get an equation which if we take a look at it with the use of the `plot` method, resembles a trajectory line.

First we need to declare a time variable let us call it `dt=0`; Our next step is to create the positions and speeds.

```
pos_x = 0; % The x coordinate of the position  
pos_y = 10; % The y coordinate of the position  
spe_x = 4; % The x coordinate of the speed  
spe_y = 0; % The y coordinate of the speed
```

Then we need to create the gravitational force. Here I have set it to 0.5 which fits our need in Matlab, but as I said before the optional value is -0.01 N (newton)

```
g=0.5
hold on
grid on
while (T<20)
```

```
plot(pos_x,pos_y, 'x')
xlim([-20 100])
ylim([-100 20])
The status vector is
```

```
vec=(pos_x,pos_y,spe_x,spe_y)
```

The differential equation is
 $d/dt(\text{vec})=f(\text{vec})$; where $f=(f_1,f_2,f_3,f_4)$ vector components.

The size of one step is dt so the modified method equation is

```
vec_new=vec_old+dt*f(t_old+1/2*dt,vec_old+1/2*dt*f(t_old,vec_old))
```

And because of does not rely on the final formula is

```
vec_new=vec_old+dt*f(vec_old+1/2*dt*f(vec_old))
```

To simplify things $\text{vec_old}+1/2*dt*f(\text{vec_old})$ can be called a helper dummy variable. Now we are ready to code, and here it is:

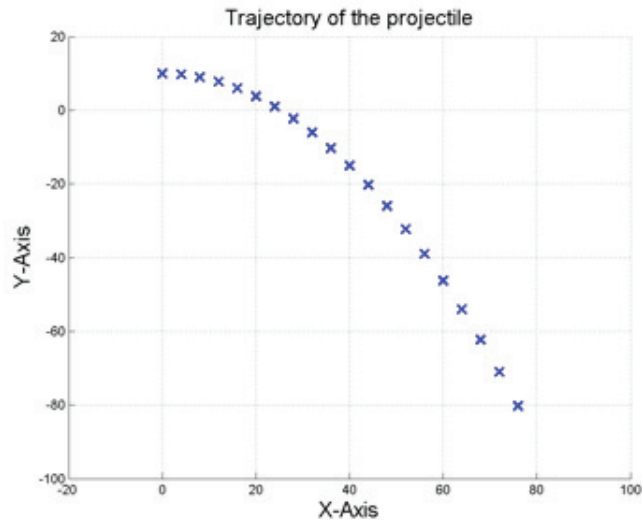
```
pos_x_d = pos_x + 0.5 * dt * spe_x;
pos_y_d = pos_y + 0.5 * dt * spe_y;
spe_x_d = spe_x + 0.5 * dt * 0;
spe_y_d = spe_y + 0.5 * dt * (-g);

pos_x = pos_x + dt * spe_x_d;
```

```
pos_y = pos_y + dt * spe_y_d;  
spe_x = spe_x + dt * 0;  
spe_y = spe_y + dt * (-g);  
T=T+dt;  
end  
hold off
```

Here we can see the plot of this calculation:

Figure 3. Trajectory MATLAB plot.



C. Code in Java

Luckily the code can be quite easily implemented to the game. All I had to do were some int to double conversions. Instead of a loop we have already an update object that does it is purpose quite well.

Here is the final product:

```

public class Projectile {
    private int x,y,T;
    private double pos_x,pos_y,spe_x,spe_y,g,dt,
    pos_x_d,pos_y_d,spe_x_d,spe_y_d;
    private boolean visible;
    private Rectangle r;
    public Projectile(int startX, int startY) {
        T=0;
        dt=1;
        pos_x = startX;
        pos_y = startY;
        spe_x = 3;
        spe_y = 0;
        visible = true;
        g= -0.01;
        r = new Rectangle(0, 0, 0, 0);
    }
    public void update() {
        pos_x += spe_x;
        r.setBounds(x, y, 10, 5);
        pos_x_d = pos_x + 0.5 * dt * spe_x;
        pos_y_d = pos_y + 0.5 * dt * spe_y;
        spe_x_d = spe_x + 0.5 * dt * 0;
        spe_y_d = spe_y + 0.5 * dt * (-g);

        pos_x = pos_x + dt * spe_x_d;
        pos_y = pos_y + dt * spe_y_d;
        spe_x = spe_x + dt * 0;
        spe_y = spe_y + dt * (-g);
        if (pos_x > 800) {
            visible = false;
            r = null;
        }
        if (pos_x < 800) {
            checkCollision();
        }
    }
}

```

```
}  
T=(int) (T+dt);  
}
```

Conclusion

In the project a basic platformer game was given as a base to my concept. We have upgraded this with a shooting method, and a projectile trajectory system. To create this system I had to use the Euler method that was modified in a way to fit all my needs and nothing more. The goal of this work was to widen the ways of creating a game that really stands out from the crowd. For this project to develop there are two ways, one of them is the more self-explanatory one that is a better more advanced game. The other is two create a trajectory simulator system which works according to real life physics.

Finally all have been seen and done, we can close the project. It successfully satisfied all of our expectations hence why it can be considered a success. Though as I have covered it in the appropriate segment there are lots of ways to go from here. The program can be implemented as web or cloud application also [13].

[13] Dukan, P.–
Kővári, A. (2013):
Cloud-based smart
metering system.
*Proceeding of the 14th
IEEE International
Symposium on Com-
putational Intelligence
and Informatics*. Bu-
dapest. Pp. 499–502.