Equation-based Movement on Unity

Cendhika Imantoro - 13514037
Informatics Engineering
School of Electrical and Informatics Engineering
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514037@std.stei.itb.ac.id

***Abstract*—Unity is one of several well-known tool that oftenly used for making an interactive application. Unity provides various feature which makes it's usage become easier. Each object in Unity have a component called "Transform" which used to define the position, rotation, and size of an object. Other feature that's commonly used in Unity is "Physics". The function of this feature is to emulate physical phenomenon than happen on an object. One example of such phenomenon is velocity. Now, if we want an object to move in the curve of an equation, we could do it either by changing the "Transform" of that object, or changing its velocity through the "Physics" feature. This paper explain the difference between each methods.**

Keywords—equation; velocity; position; delta time; curve

I. INTRODUCTION

Nowadays, in order to create a game, there is no need for us to create it from scratch. The reason of that is because there are a lot of tools that available to use. One of such tools is Unity. Unity oftenly correlated with creating a game application, but actually it could be used in creating various kind of software.

One of the software that could be created by using Unity is a software for visualizing an equation. One of visualization method that could be used is by moving an object along the curve of an equation. Such software would need an object to move in a certain track. Such movement could be implemented either by using "Transform" or "Physics" feature in Unity.

"Transform" is a component which define the position, rotation, and size of an Unity object. Basically, it define the "state" of an object in spatial dimension. Every object in Unity have this component. If we want an object to move along with a curve using "Transform", what we need to do is set the value of position based on the curve.

"Physics" is a feature that could emulate physical phenomenon on an object. It could emulate gravity, velocity, collision, and many more. To move an object along a curve by using "Physics", we need to set the velocity of an object based on the curve and current position.

While both method could be used for moving an object along a curve, they're fundamentally different and doing different thing on the object. In order to decide which method to use, we need to now the difference between both method.

II. METHOD USING "TRANSFORM"

This method is actually really simple. Let's assume an object has initial position of (x_0, y_0). We want that object to move in curve f(x) with constant horizontal speed v. It's obvious that y_0 should be equals to $f(x_0)$ because if it isn't, the object wouldn't be in the curve from the start. In order to move the object, we need to update its position for every "change of time". Unity has a unit of time called delta time which is basically a very, very small unit of time. We need to move the object based on this unit. For each delta time, the change on the object should be done as follows:

1) Determine the object's new horizontal cordinate: This could be done by using equation

$$X' = X + (v * \text{delta time})$$

with X' as the new horizontal position and X as the old horizontal position.

2) Determine the new vertical position: Since the object would move in the curve of f(x), the new vertical position should be

$$Y' = f(X')$$

with Y' being the new vertical position.

3) Change the position of the object to (X', Y')

Updating the object's position by doing those steps repeatedly will make the object to move along the curve.

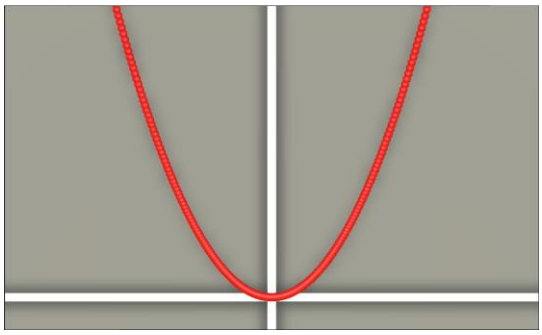


Figure 1 Track of an object moved by changing the position on $f(x) = x^2$

III. METHOD USING "PHYSICS"

Instead of updating the position, this method do an update on the velocity for each delta time. The question is, what value should be used for the update? The first idea would be using

the derivation of the curve equation which is supposed to be equals to the velocity at the moment. Let's assume an object has initial position of (x_0, y_0). We want that object to move in curve f(x) with constant horizontal speed v. The update should be done as follows:

1) Determine the new horizontal speed: In this case, the horizontal speed (V'_x) would be v.

2) Determine the new vertical speed: Since we are using the derivation of curve as speed, the vertical speed would be

$$V'_y = f'(X)$$

with V'_y as the new vertical speed, and X as the current horizontal position.

3) Change the velocity to (V'_x, V'_y)

Based on physics in real life, the steps above should make the object to move along the curve. But, that's not the case for Unity. If we use those steps in Unity, there would be a slight error in the vertical position for each delta time which would add up into a really big difference.

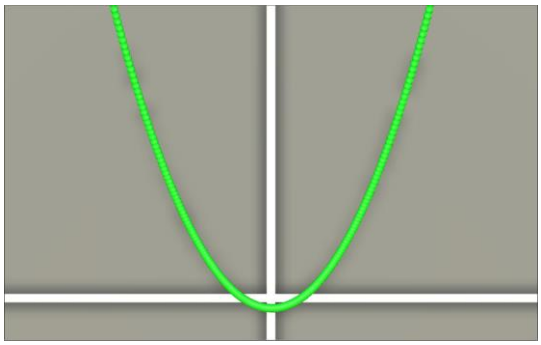


Figure 2 Track of an object moved by changing the velocity to curve derivation on $f(x) = x^2$. On x = 0, the vertical position supposed to be 0 based on the equation, but the image shows slightly lower vertical position.

What's causing the error? What make real life physics differ from Unity's emulated physics is the delta time. In real life physics, velocity acquired from derivating position equation by using a delta time which is really close to zero. Meanwhile, the delta time in Unity is nowhere close. While in span of Unity's delta time Unity only change the velocity once, in reality that much time should cause infinite change on the velocity.

Does it mean we can't use velocity if we want to precisely move an object along a curve? There is a limit of precision given the delta time used, but we can achieve better precision by changing the value used for update. Instead of using derivation, we should calculate "the velocity needed so that after delta time, the position would be at (X', f(X'))". To do this, the steps should be as follows:

1) Determine the object's next horizontal position: the position should be equals

$$X' = X + (v * \text{delta time})$$

with X' as the next horizontal position and X as the current horizontal position.

2) Determine the object's next vertical position: The position should be equals

$$Y' = f(X')$$

with Y' as the next vertical position.

3) Determine the object's horizontal speed: In this case, the horizontal speed (V'_x) should be equals v

4) Determine the object's vertical speed: Since we want the object to move through the distance of vertical difference in span of delta time, the speed should be equals

$$V'_y = (Y'-Y)/\text{delta time}$$

with V'_y as the horizontal speed and Y as current vertical position.

5) Change the object's velocity to (V'_x, V'_y)

The steps above should make it as if the object moving along the curve.

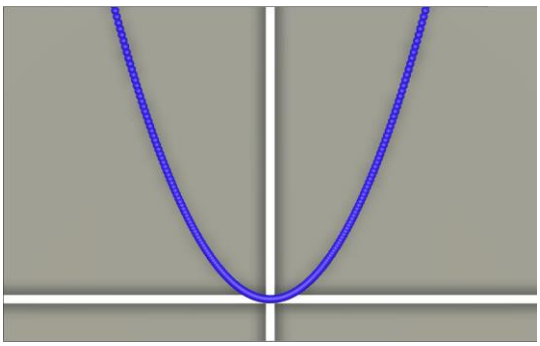


Figure 3 Track of an object moved by changing the velocity based on the current position and the next position on $f(x) = x^2$.

IV. Analysis

Based on the explanation above, both method could be used for moving an object along a curve. Based on the complexity of method, the one using "Transform" is less complicated than the one using "Physics". Both method calculate the new next position, but the later one include calculating the velocity. But, there might be times when we shouls use the later. The one using "Transform" might be more simple, but it didn't have velocity because it's just changing the position. The one using "physics" has it. If the aplication we want is "Emulating the collision of an object that move along the curve f(x) with an object that move along the curve g(x)", we shouldn't only changing the position, but also the velocity since momentum can only appear when the object has velocity. Thus, we should use the method that use "Physics". But, if what we want is only to view the shape of a curve, then the one using "Transform" should be enough.

V. Conclusion

There are more than one method to move an Unity object along a curve. Which method should be used, depend on the usage and purpose of the movement.

Acknowledgment

For the completion of this paper, the author thanks Mr, Rinaldi Munir as the lecturer of class 1 in the course IF3280 Socio-Informatika dan Profesionalisme. The author also thanks Mrs. Ayu Purwarianti and Mrs. Dessi Puji Lestari as the lecturer of combined class in the same course.

References

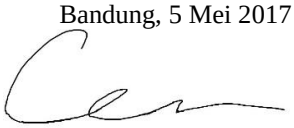
[1] https://docs.unity3d.com/ScriptReference/Object.Instantiate.html

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/IEEE-Paper-Template-Stima-2017.doc

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Mei 2017



Cendhika Imantoro - 13514037