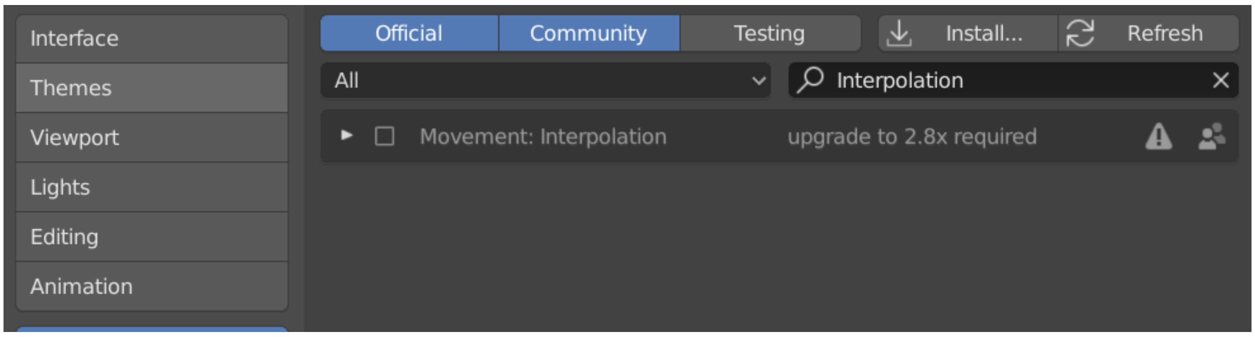
User manual for add-on "Interpolation"

Authors:
Marc Guerrero Palanca
Tatsiana Palikarpava
Alberto Pérez Abad

In the following manual we describe designed add-on, used for working with animated objects and modifying their trajectories according to user settings.

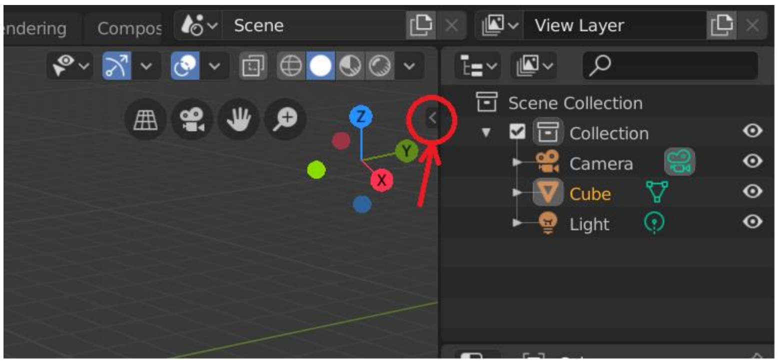
1. Installation & Use

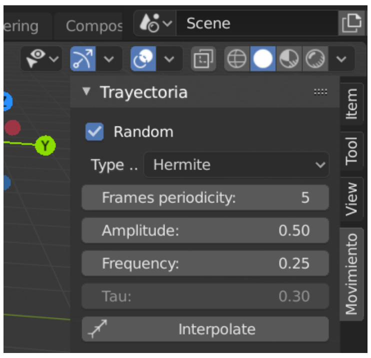
- Our add-on is located in archive **interpolation.zip.** To install it, it is necessary to choose it in user preferences for installation. Previously, you need to save the file.



Movement: Interpolation   upgrade to 2.8x required

- After installation, you will find our panel by clicking the arrow marked on the image or pressing "N". You will see a tab "Movimiento", which is actually what we will be using.





Trayectoria
Random ✓
Type .. Hermite
Frames periodicity: 5
Amplitude: 0.50
Frequency: 0.25
Tau: 0.30

- Next, you need to create some objects and define action for them by inserting keyframes.
- In the window you see the following properties:

Check-box **Random**	If not checked, movement will be produced accurately according to other properties such as type of interpolation, frequency, etc. If checked, for each keyframe coordinates of the object will be changed by defining some deviation from normal trajectory.
Dropdown list **Type of interpolation**	User can choose between 3 types of interpolation: *Linear, Hermite* and *Catmull-Rom.*
Integer property **Frames periodicity**	User defines the periodicity of inserting new keyframes.

Float property **Amplitude**	User defines maximal distance of object's deviation from normal trajectory. If **Random** option is not checked, property **Amplitude** is blocked.
Float property **Frequency**	User defines frequency of random movement. If **Random** option is not checked, property **Frequency** is blocked.
Float property **Tau**	User defines value of parameter τ, used for *Catmull-Rom* interpolation. If not *Catmull-Rom* option is chosen, property **Tau** is blocked.
Button **Interpolate**	This button performs modifying of the trajectory. It is designed with concern to prevent some errors, occurring because of lack of information about the object. To have this button enabled program checks that each object, to which user is going to apply interpolation, contains animation data and particularly action, with at least 2 keyframes.

- After defining all the required properties press **Interpolate.** For all selected objects trajectories will be modified. You can press "Ctrl+Z" to return to the previous state.
- If you are using Hermite interpolation, after interpolating you will see some arrows, which are designed to demonstrate velocities. User can scale the arrows to adjust the movement. Then it will be necessary to reinterpolate trajectory, so that the calculations will be done with edited values of velocities.

2. Action management

Let us consider our initial problem. We have several metaballs and some trajectory. What we need is to define some kind of movement within this trajectory, so that metaballs form something like a bubble, staying rather close to each other, but not having identical paths.

Taking into account the fact that the amount of balls is not limited, we try to avoid creating numerous trajectories manually. For example, we defined required trajectory for one ball. One possible solution is to copy this object a few times, so that newly created objects will have the same trajectories. In the early stages of developing of this tool, we used a function *CopRuta,* which aim was to copy the trajectory of one object to all objects of the list, which is the parameter of the function. So, we were usually creating one metaball, defining its trajectory and applying CopRuta to the list of other metaballs in the scene.

However, more wise solution would be to work a bit with *Action Editor.*

We have an object Mball with defined trajectory, called MballAction. In *Action Editor* we can press *Stash* to unlink it from the object Mball. Then we copy the trajectory (it is now called MballAction.001). Now it's time to create one more metaball Mball.001. Select it and in *Action*

Editor choose MballAction, then copy it (now we have MballAction.002). Repeat the same for the third metaball Mball.002 with action MballAction.003.

After these preparations, we can apply our add-on to the balls. For example, we select Mball.001 and Mball.002, then define properties (*Random* = True, *Type* = Lineal, *Frames periodicity* = 3, *Amplitude* = 0.3, *Frequency* = 0.25) and press *Interpolate.* Finally, actions MballAction.002 and MballAction.003 have changed.

Well, let us suppose that we changed our mind and now decided that it is necessary to interpolate initial trajectory for all the three balls with Hermite interpolation. In such case, we select Mball.001 and Mball.002 one by one, choose MballAction (it is still stored), and copy it (now we have MballAction.004, MballAction.005). We don't need to perform it for Mball, because its trajectory hasn't been changed. We define properties (*Random* = True, *Type* = Hermite, *Frames periodicity* = 3, *Amplitude* = 0.5, *Frequency* = 0.5) and press *Interpolate.*

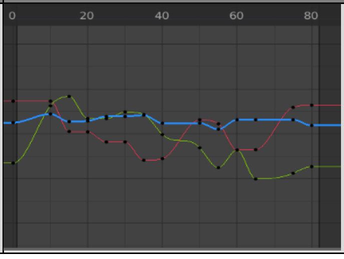
Therefore, the main idea is to store all changed trajectories separately and not forget to save the trajectory before modifying. This gives us the following opportunities:

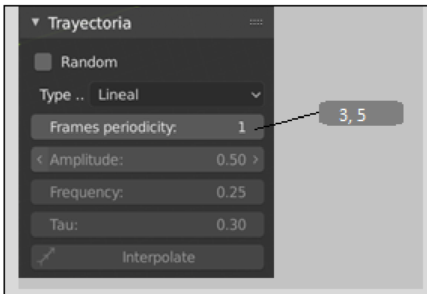
- ✓ We can always return to the previous state.
- ✓ It is possible to create different interpolations for one object using one trajectory as a base. We can also compare the behavior of the object with different trajectories (let them be called *Trajectory1*, *Trajectory2*). For this, we can copy our object and assign it *Trajectory2* in *Action Editor,* while for initial object it will be *Trajectory1.* In that way, user can see simultaneously two trajectories in real time.
- ✓ We can assign one object trajectory of other and then modify it.
- ✓ We can apply sequentially different types of interpolation.

Therefore, our program does not deal with storing the actions automatically, so once you have pressed *Interpolate,* current path is modified. But as we described above, you can store everything you will need manually.

3. Add-on in use

Below we will give you an example of use of our tool, showing the properties and the result of applying them to the objects. We create three metaballs in the scene and define the trajectory using **Circuito.png**. All the results will be obtained by undertaking interpolation with following properties for all the balls and will be stored in separate trajectories.

Properties	Description of the result
No interpolation	



Trayectoria

Random: ☐

Type .. Lineal

Frames periodicity: 1

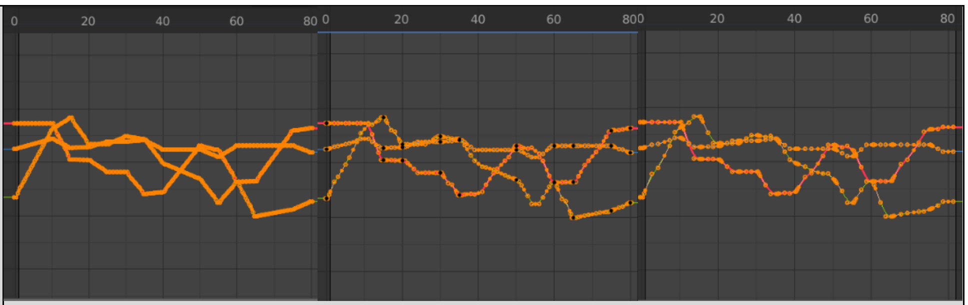
Amplitude: 0.50

Frequency: 0.25

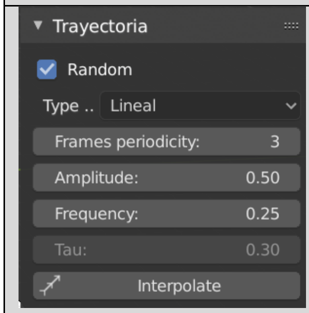
Tau: 0.30

Interpolate

3, 5



This is the easiest case. We create linear interpolation, calculating the position for each 1, 3 or 5 keyframes. Linear interpolation produces not very natural movement in this case, because the velocity between keyframes is constant. Because of unchecked *Random* option, all the balls have the same trajectory and move as one ball.



Trayectoria

Random: ✓

Type .. Lineal

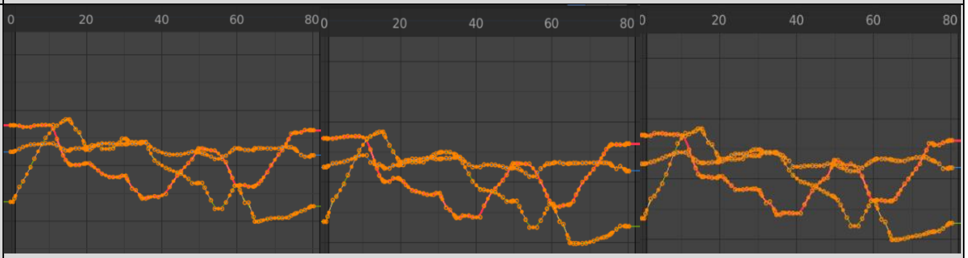
Frames periodicity: 3

Amplitude: 0.50

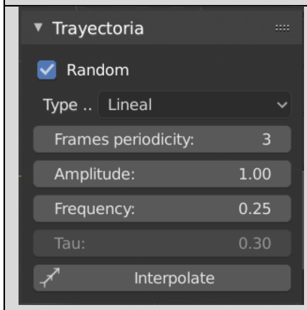
Frequency: 0.25

Tau: 0.30

Interpolate



Here we use *Random* option. In the picture there are trajectories of three balls which are very similar, but with little deviations.



Trayectoria

Random: ✓

Type .. Lineal

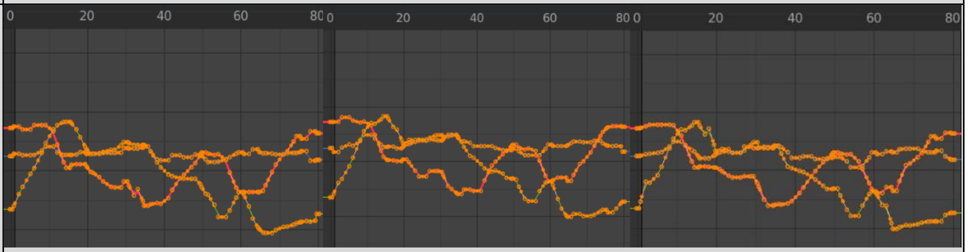
Frames periodicity: 3

Amplitude: 1.00

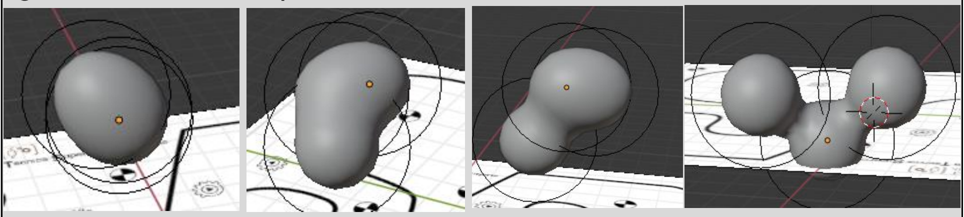
Frequency: 0.25

Tau: 0.30

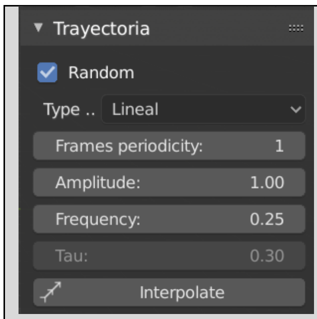
Interpolate



In this case, we use greater amplitude, which implies greater deviation. Concerning the image, we have a more dynamic bubble than in previous case. Compare in the picture below bubbles, generated with amplitudes 0.5, 1, 1.5, 2.



Metaballs will stay united until amplitude is not greater than double radius of the balls.


▾ Trayectoria

✓ Random

Type .. Lineal

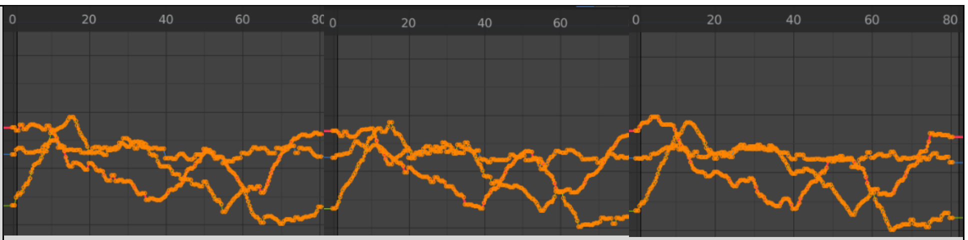
Frames periodicity: 1

Amplitude: 1.00

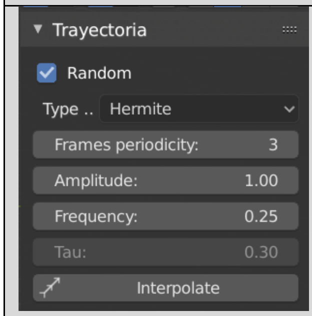
Frequency: 0.25

Tau: 0.30

Interpolate



When we use *Random* option the more is Frames periodicity, the more smooth “bubbling” will be produced. If we set it to 1 the bubble will be trembling, but for 5 it will be changing shape leisurely.


▾ Trayectoria

✓ Random

Type .. Hermite

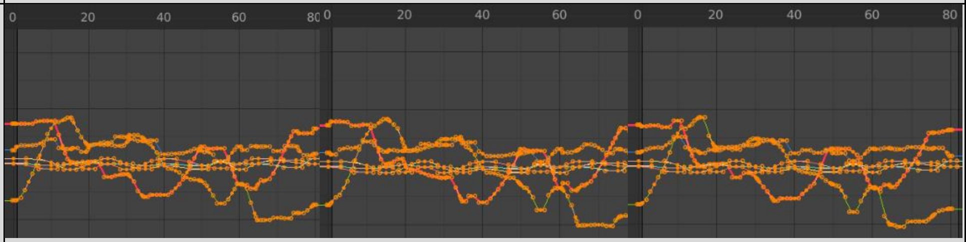
Frames periodicity: 3

Amplitude: 1.00

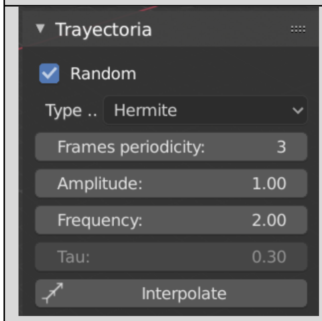
Frequency: 0.25

Tau: 0.30

Interpolate



As it was mentioned above, linear interpolation produces not very natural behavior, but with Hermite and Catmull-Rom we can overcome it.


▾ Trayectoria

✓ Random

Type .. Hermite

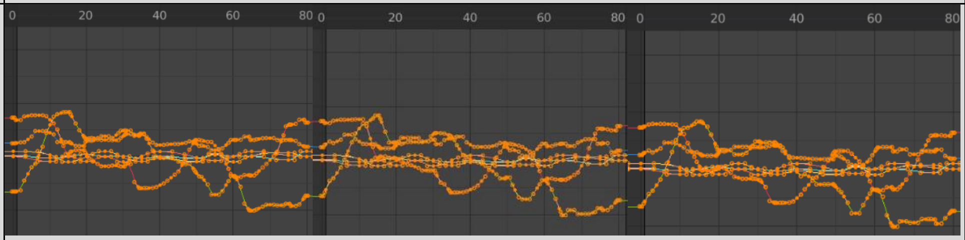
Frames periodicity: 3

Amplitude: 1.00

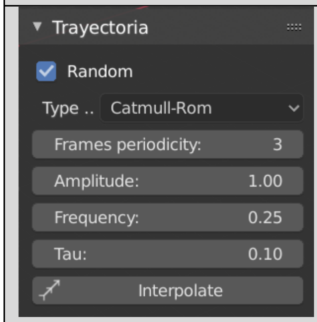
Frequency: 2.00

Tau: 0.30

Interpolate



Here the frequency effects the way how random deviation is generated, so that higher frequency values produce more dramatic changes between adjacent keyframes. However, it does not much effect on behavior of our bubble.


▾ Trayectoria

✓ Random

Type .. Catmull-Rom

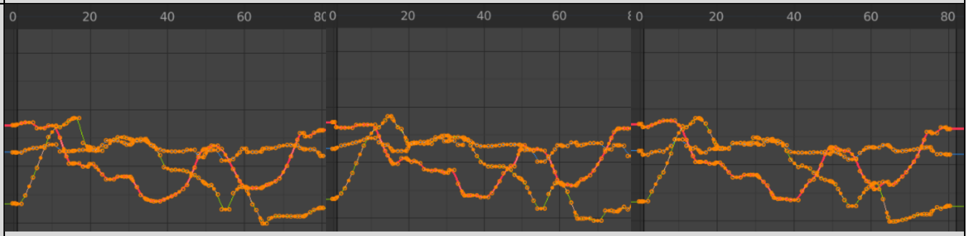
Frames periodicity: 3

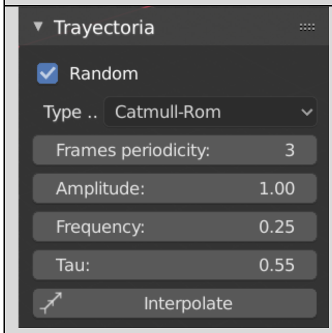
Amplitude: 1.00

Frequency: 0.25

Tau: 0.10

Interpolate




▾ Trayectoria

✓ Random

Type .. Catmull-Rom

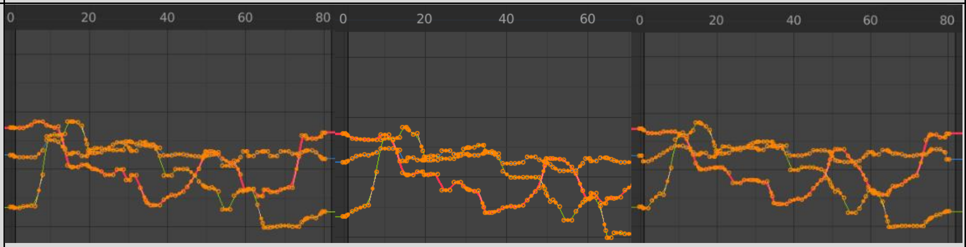
Frames periodicity: 3

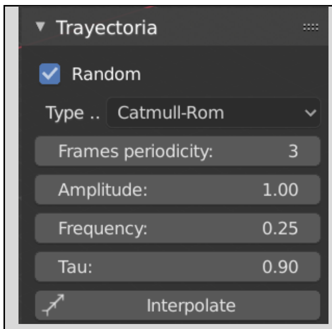
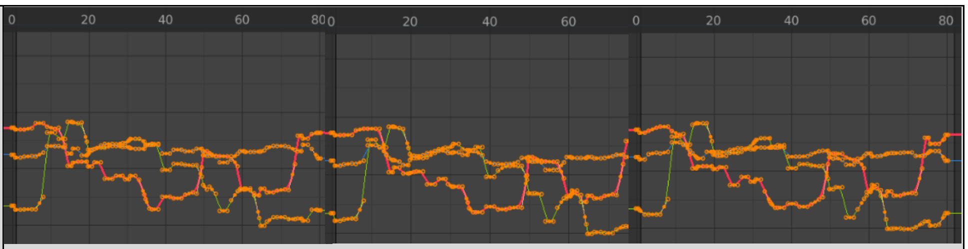
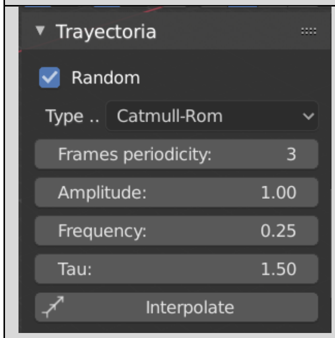
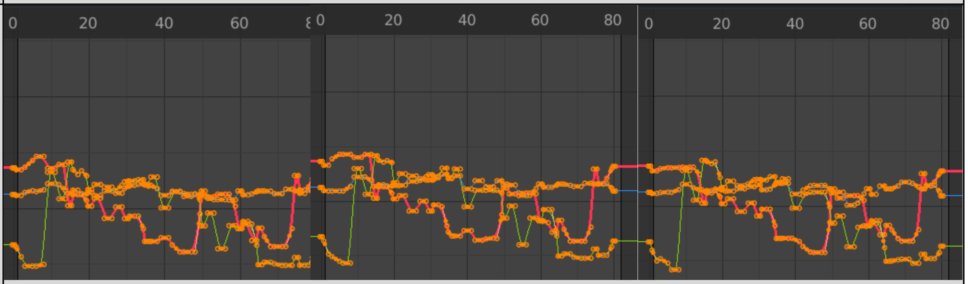
Amplitude: 1.00

Frequency: 0.25

Tau: 0.55

Interpolate



 ▼ Trayectoria ✓ Random Type .. Catmull-Rom Frames periodicity: 3 Amplitude: 1.00 Frequency: 0.25 Tau: 0.90 Interpolate	
 ▼ Trayectoria ✓ Random Type .. Catmull-Rom Frames periodicity: 3 Amplitude: 1.00 Frequency: 0.25 Tau: 1.50 Interpolate	 In the following pictures there are the results of Catmull-Rom interpolation with different values of τ. You can notice that greater values of τ produce more “frivolous” trajectories. For example, for $\tau = 1.5$ balls sometimes get really far from expected trajectory. One more crucial moment is that for high τ movement becomes very sharp, braking and accelerating very quickly.