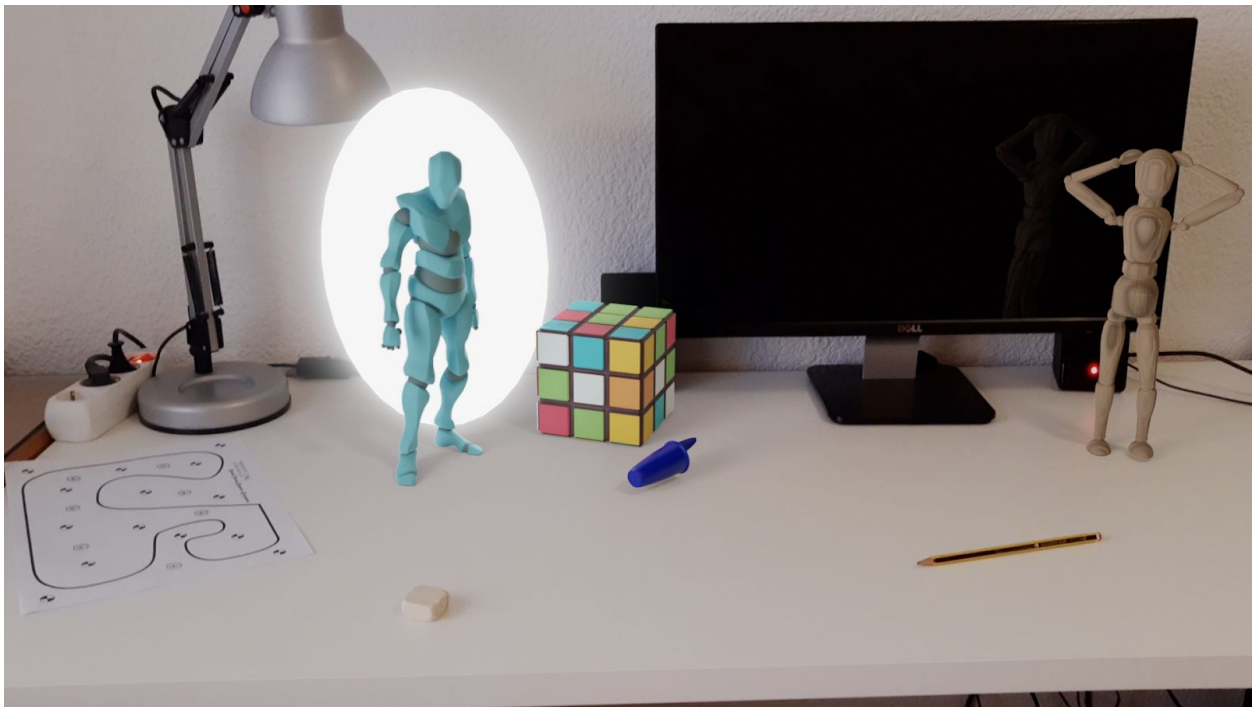
Escuela Técnica Superior de Ingeniería (ETSE-UV)

Animation
Final project



Guillem Barceló Orts

Ana Gloria Gálvez Mellado

Darío Rodríguez Hernández

Bekzat Mukhamedali

Index

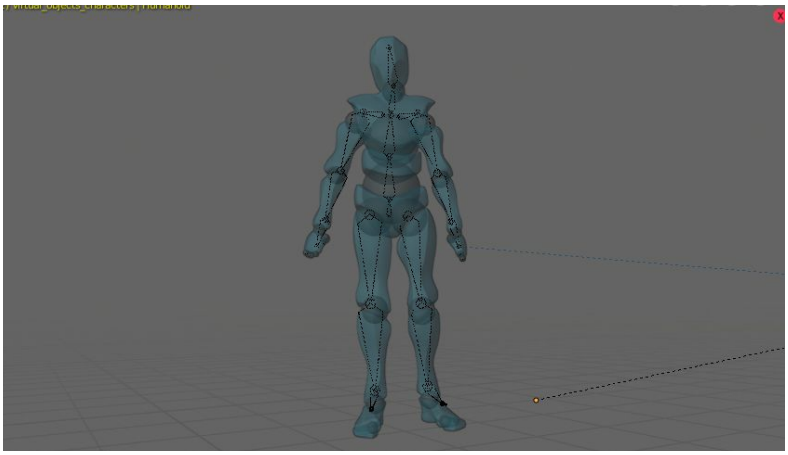
Video events

A small humanoid opens a portal on a desk and jumps to our reality. Due of his size, a bee scares the humanoid. He hides behind a computer monitor but eventually loses his fear and frightens the bee. Then, he weights with a pen cap and throws it. He has nothing else to do, so he opens a new portal on the ground and jumps into it. At the same time, a wood mannequin tries to get attention but he runs away when the humanoid is about to throw the pen cap.
Final Animation link: https://www.youtube.com/watch?v=aD6cgvClRKo
Animation Techniques: https://www.youtube.com/watch?v=KuA0a_tHry0&feature=youtu.be

Objects and movements

Humanoid

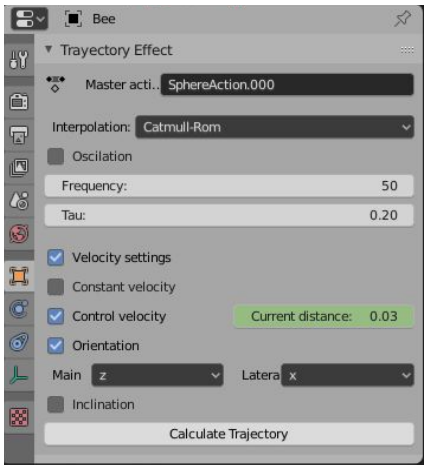
The humanoid model was retrieved from the Internet (see Resources), we deleted some parts of the original model because it had more articulations than the armature generated by the .bvh (the MOCAP file) and it was more realistic without them.
As the original model had finger articulations, on the Edit Mode, **we erased the fingers** and adjust the vertex and faces distribution **to avoid problems** like articulations getting unattached from the model.
To apply the recorded movements to the humanoid model we only had to **scale a few bones** (head, column, and feet) of the generated skeleton. It fitted the humanoid body quite well.



For correctly positioning the model at the beginning and the end of the scene (entering and going out of the portal), we used **location keyframes** (around frames 56 and 886).

Bee

We used our **add-on** to generate a trajectory the bee around the humanoid character. We inserted some **location keyframes** manually to set the trajectory.



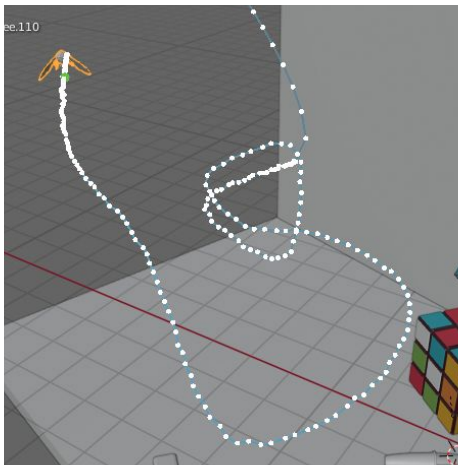
Then, with the add-on, we selected Catmull-Rom (default **Tau value 0.2**) on the Trajectory Effect panel. We checked "Velocity Settings" and "**Control Velocity**" options. To recreate slow movements (when the bee appears on the scene) and faster movements (when the bee is dodging the humanoid character), we inserted "**Current Distance**" keyframes.

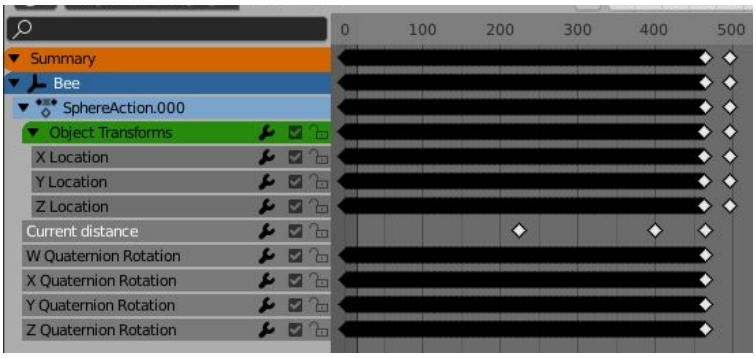
Then, we selected orientation with Z as "Main axis" and X as "Lateral axis".

With all this parameters and with maximum frequency value (50, 1 keyframe in each frame) we generate the first trajectory.

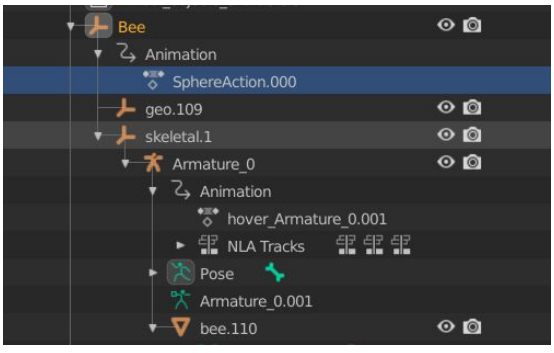
To recreate a more accurate movement, we applied the **oscillation** to the new generated trajectory, setting it as the master action, with 0.08 amplitude and maximum frequency.

The bee model was retrieved from the Internet (see Resources) and we have preserved its original animation of its armature, that moved the bee's wings.



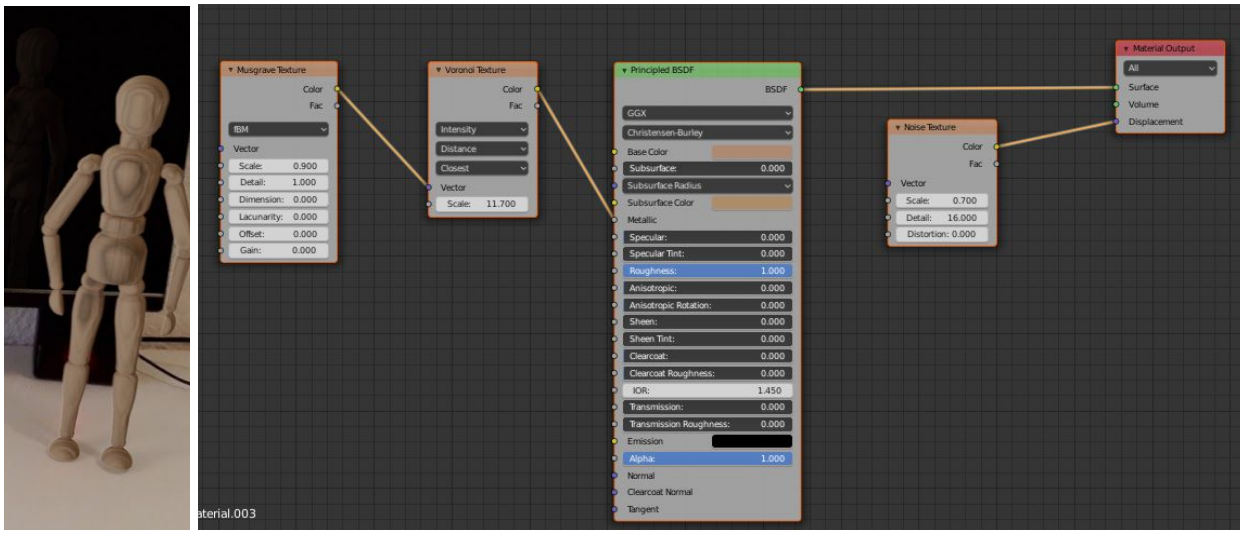


Our trajectory was applied to an empty object that contains the armature and the geometry.



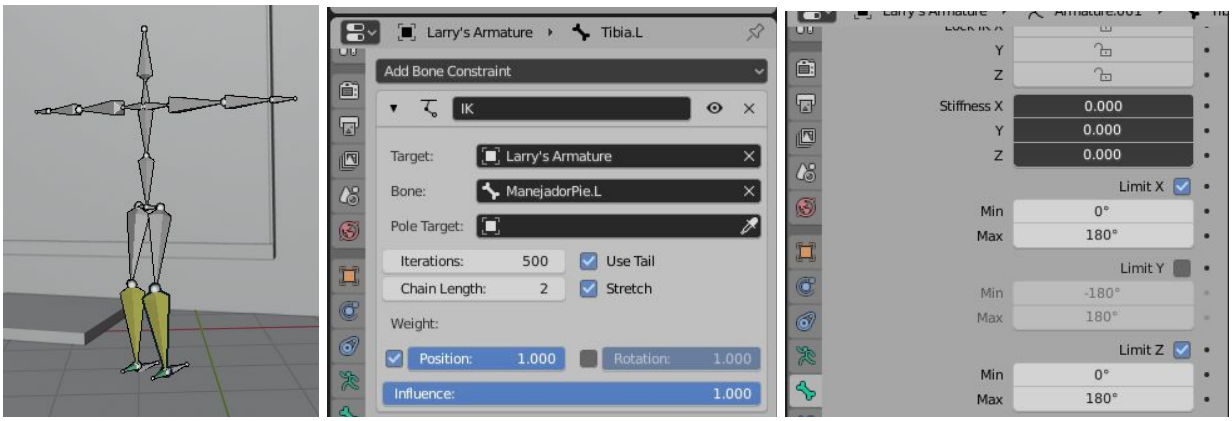
Wood mannequin

The wood mannequin was retrieved from the Internet and slightly modified (see Resources). It has a wooden material composed by Musgrave, Noise and Boroni Textures and Principled BDSF.



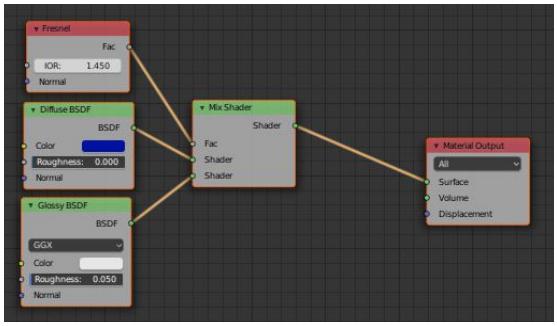
The model is **manually animated** from frame 1 to 750 when the mannequin leaves the scene.

To follow animation principles and achieve a realistic movement, arms and head are animated using **forward kinematics**, by only using rotations of the different parts (arc principle). Legs are animated with **inverse kinematics**; each foot has a handler on the ankle, and the feet have a **Copy Rotation constraint**, whose target is the handler. Each femur bone has an **IK constraint** and is limited from 0 to 180 degrees in both X and Z axis (that way, the knees could properly flex without doing inappropriate positions).

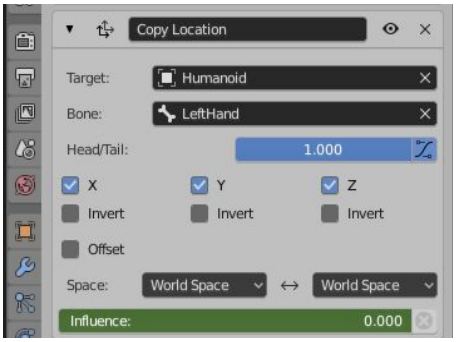


Pen cap

The pen cap is an object model retrieved from the Internet (see Resources). To make it realistic, we have applied to it a material that simulates blue plastic.
The material is a combination between Glossy BDSF, Diffuse BDSF and Fresnei nodes.

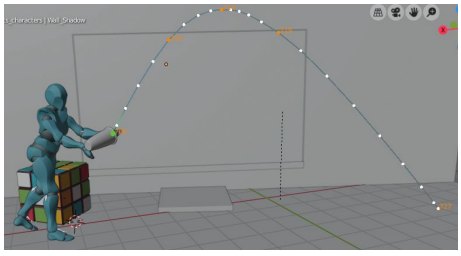


Between frames 574 and 705, the cap stays attached to the humanoid model. For recreating that action, we have used a constraint called **Copy Location**, whose target is the left hand bone of the humanoid's skeleton. During the action, the "**Influence**" value of the constraint is 1 (an influence keyframe has been inserted on frame 574), the rest of the time, the value is 0 (influence keyframe inserted on frame 705).



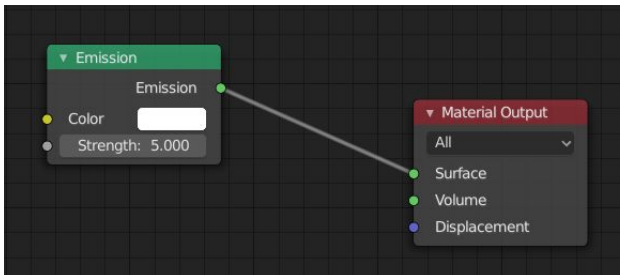
As the humanoid holds the cap with boths hands, some **rotation keyframes** have been inserted in order to assure that the cap seems to be holded also with the right hand.

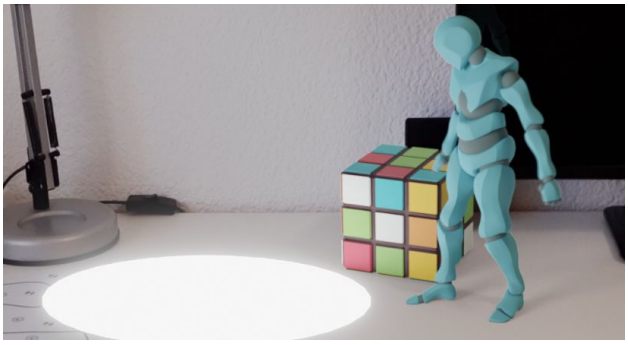
The fall of the cap has been done by manually inserting **location keyframes** to simulate a parabolic trajectory.



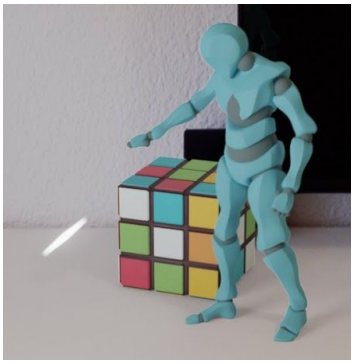
Portal and shot

To create the portals we used a sphere with non uniform **scale keyframes**. We tried a lot of different materials from the Internet to simulate a space or a time travel portal but none convinced us. Finally we create a white emissor material and a composite "**Bloom**" effect.



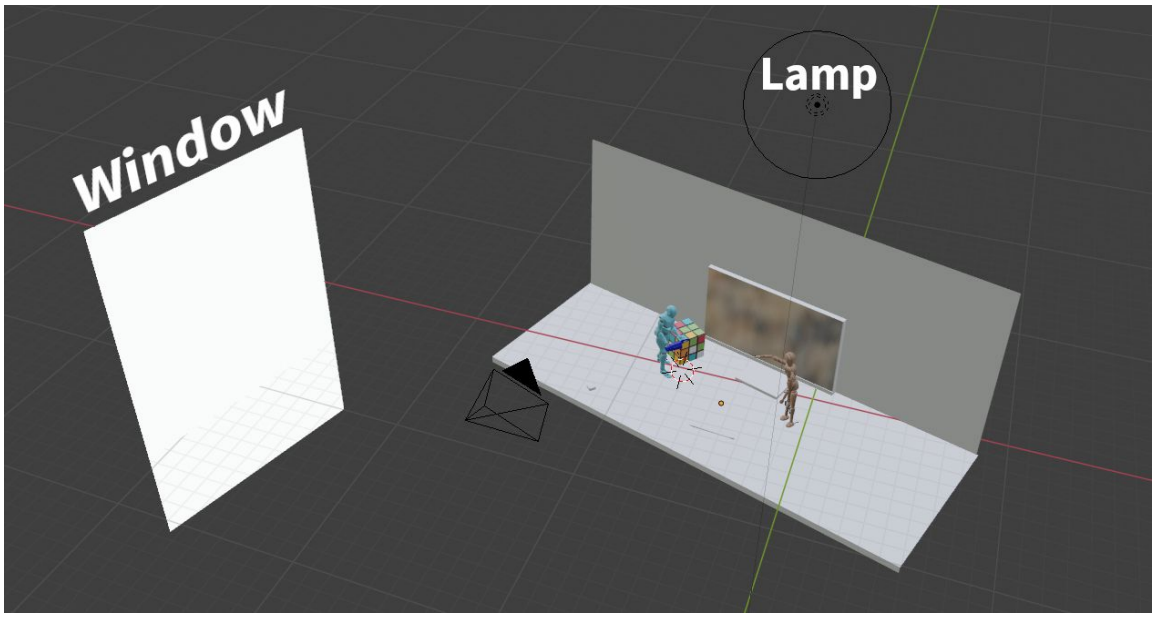


With this effect we achieved a "sci-fi" portal with a non complex material,
making the scene not too heavy.
Same technique was used with the "shot" that opens the portal. The shot is an scaled cylinder with location and scale keyframes between frames 837 to 851.



Illumination and composting

To make realistic lighting, we have created two light emitters and objects located like the real environment for capturing the lights and shadows. First, light is a **point light** that simulates the room lamp with yellowish color and a custom size (2 m.). Also, there is an **emission plane** to simulate a window/door with 8.000 strength and cold color.



All the objects are organized in collections to easily manage them in the layers.

Camera: camera.

Lights: Lamp and Window.

Screen_reflection: ScreenGlass.

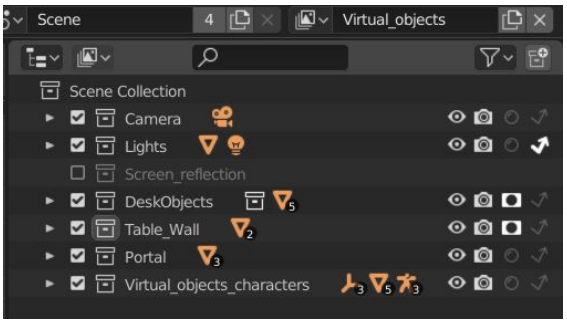
DeskObjects: real objects of the environment (screen, rubber and pencil).

Table_Wall: Real table and wall.

Portal: In and out portals and shot (bullet).

Virtual_objects_characters: All virtual objects that aren't in the real scene except the portals and bullet.

Layers

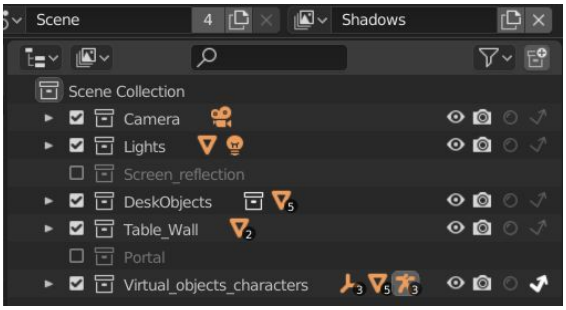


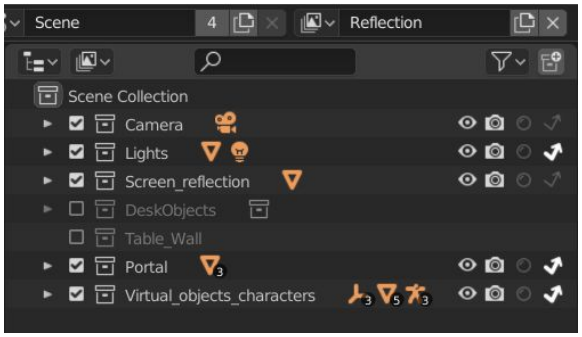
Virtual_objects

It shows the **Virtual_objects_characters** and **portals** without shadows.
The **DeskObjects** and **Table_Wall** are in Holdout mode so that virtual objects are not visible when they are behind them.

Shadows

DeskObjects and **Table_Wall** are active with the *Shadow Catcher* activated.
Virtual_objects_characters are *Indirect Only*.
The result is a layer with shadows only. Virtual objects make shadows on the table, wall, rubber, pencil and screen.



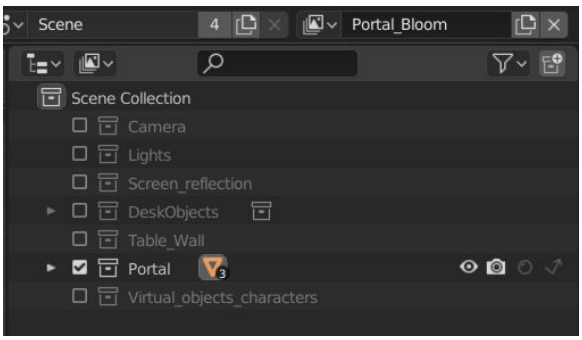


Reflection

Screen_reflection is activated. **Portal** and **Virtual_objects_characters** are *Indirect Only.*
The result is a layer that only shows the screen reflection of the virtual objects.

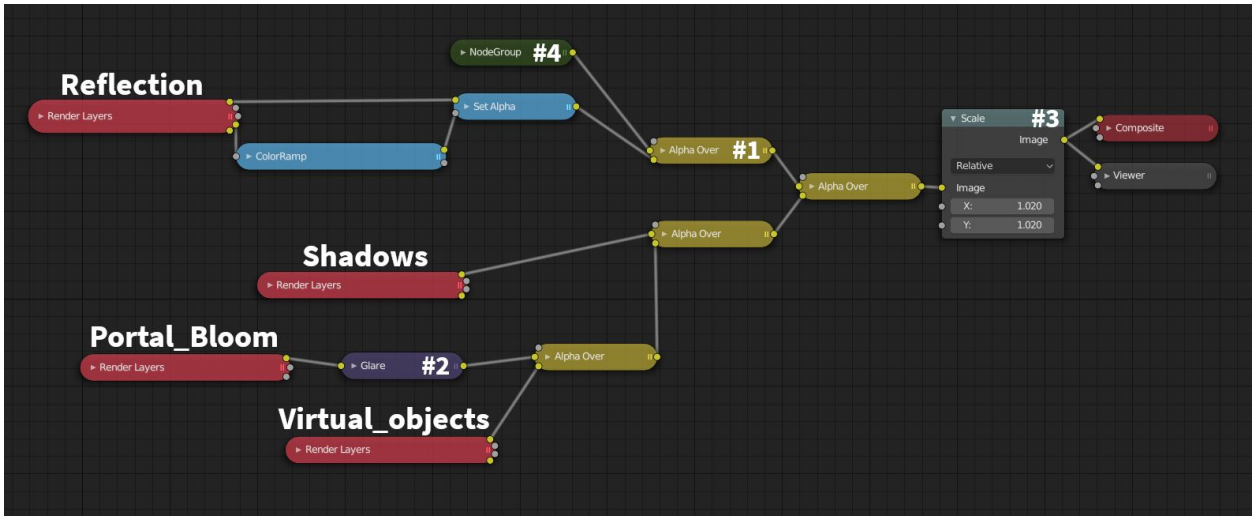
Portal_Bloom

Portal is the only activated collection.
The result is a layer that only shows the portals and the shooting bullet. We use it to create a bloom effect in the compositing.



Compositing

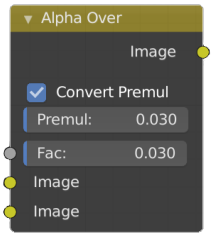
Now, it is necessary to mix the layers properly.



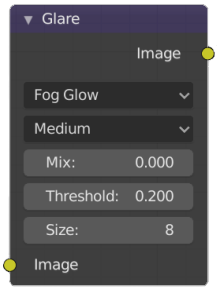
Reflection **>** Shadows **>** Portal_Bloom **>** Virtual Objects

NodeGroup #4 :
Background video. It is created automatically when doing the motion tracking and we have grouped it into a NodeGruop.

Alpha Over #1 :
0.3 value and convert premult to simulate the screen reflection.



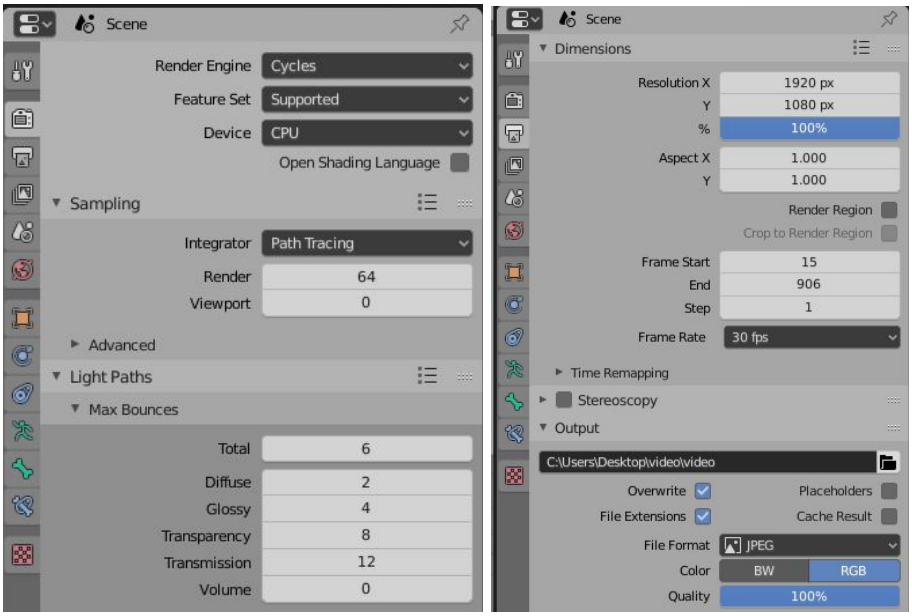
Glare #2 :
Glare is used to make gloom effect. We tested different parameters until we got a satisfactory result.



Scale #3 :
Due to focal length correction of the original video, the video was curved making small black borders. To fix it, we have add a final scale node with the value *1.02* in X and Y axis.

Rendering process

Once the scene was completed, we rendered the animation. We choose to render it as an **image sequence** so several computers could do it at the same time.
Those were the render parameters that give us a good time / quality result:



In Virtual_objects, Shadows and Reflection layer:

It took about **a day and 5 computers** to render the animation.

Difficulties

During the development of the project, we have faced some problems, explained below:

The first problem was that, as the motion capture movements were recorded on a small area and without having space references of the original video, **the movement** of the character **did not fit perfectly** on the scene. We wanted it to "hide" behind the computer screen, however, the foot of the humanoid **seemed to be over the speaker** of the real video. To solve that, we simply add the Rubik's cube **model covering the speaker**.

Another problem was the pen cap when the humanoid throws it. We tried to **add rigid body physics** to the floor and the cap, for simulating it. Nevertheless, they didn't work properly as it described a **non-realistic trajectory** even changing the weight of the cap. Finally, we decided to **do it manually**.

We were searching different ways to make a ***light catcher*** material to catch the portal light on the table and wall (it is not yet implemented a *light catcher* option in Blender); so it looked more realistic. There weren't easy solutions, so finally we decided to create and add a ***bloom* effect** to the portal. To do this using cycles, it's necessary to modify the Compositing.

The number of polygons in the scene slowed down the viewport, making it harder to render and work with the project. The most important problem was the bee, that had too many polylongs. To solve that, we add (to the bee and others objects) a ***decimate modifier*** that decreased considerably the number of polygons , keeping a good final result.

Add-on code structure

The code is divided into two files, **front-end** (interfaz.py) and **back-end** (script.py), and one extra file (__init__.py) that helps joining them correctly as a Blender tool.

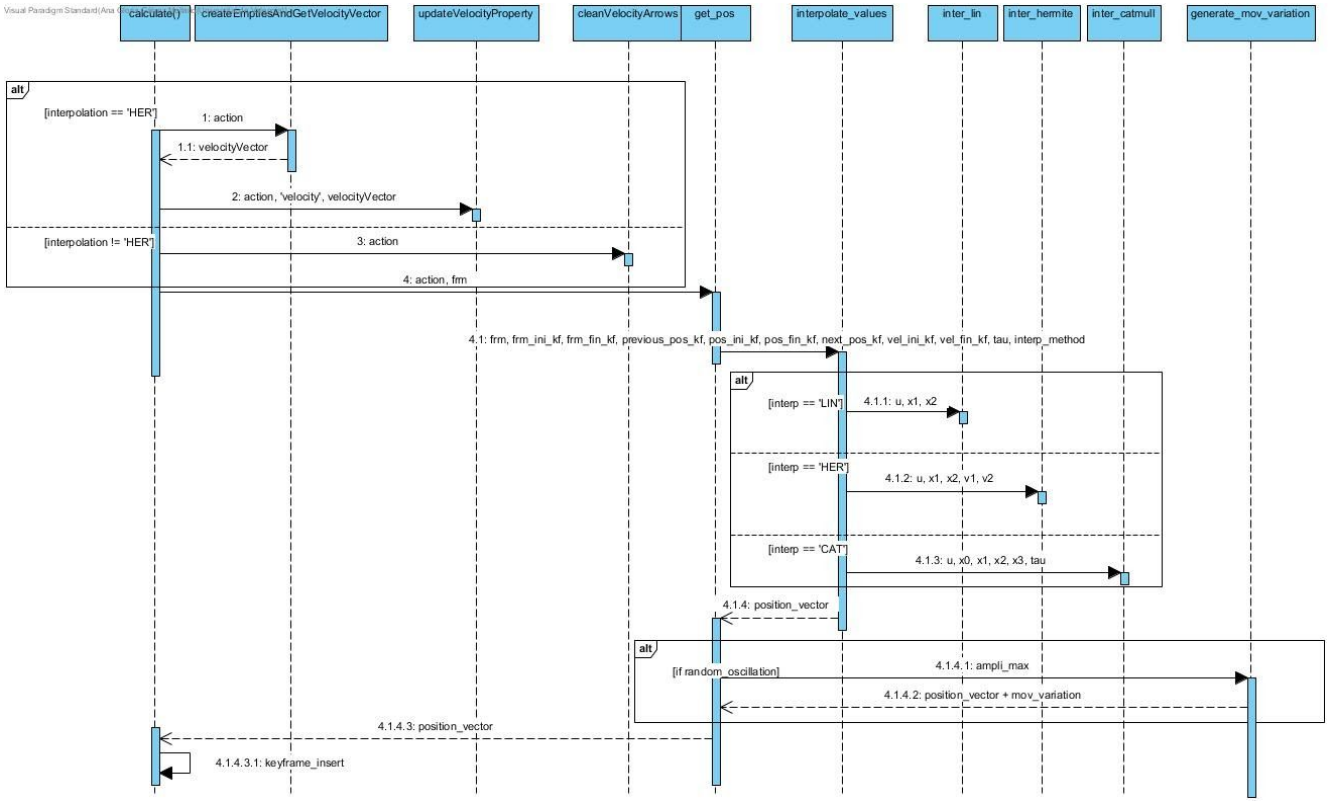
Script.py

That file contains all the functions related with calculations. In total, it has 17 functions.

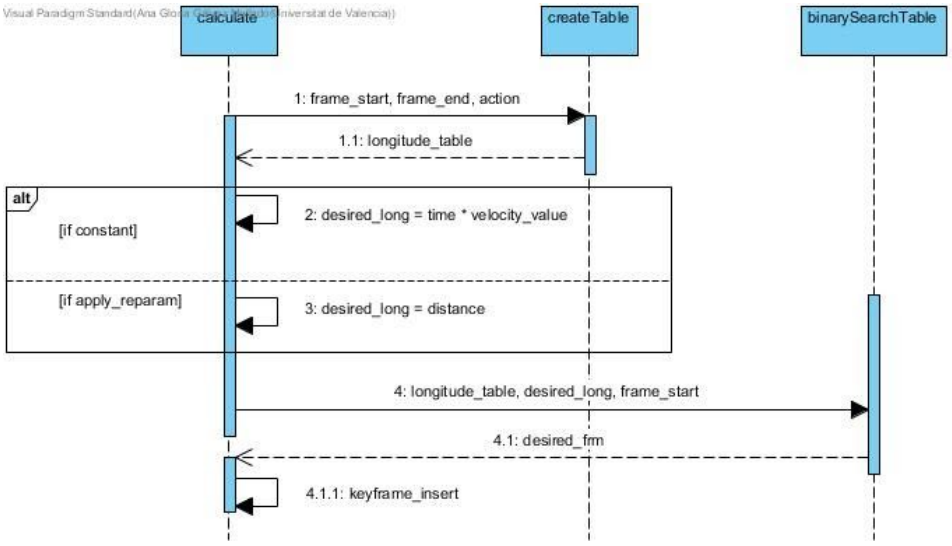
We can classify functions according to its utility:

Interpolation functions: interpolate_values, inter_lin, inter_hermite, inter_catmull, createEmptiesAndGetVelocityVector, updateVelocityProperty, cleanVelocityArrows.

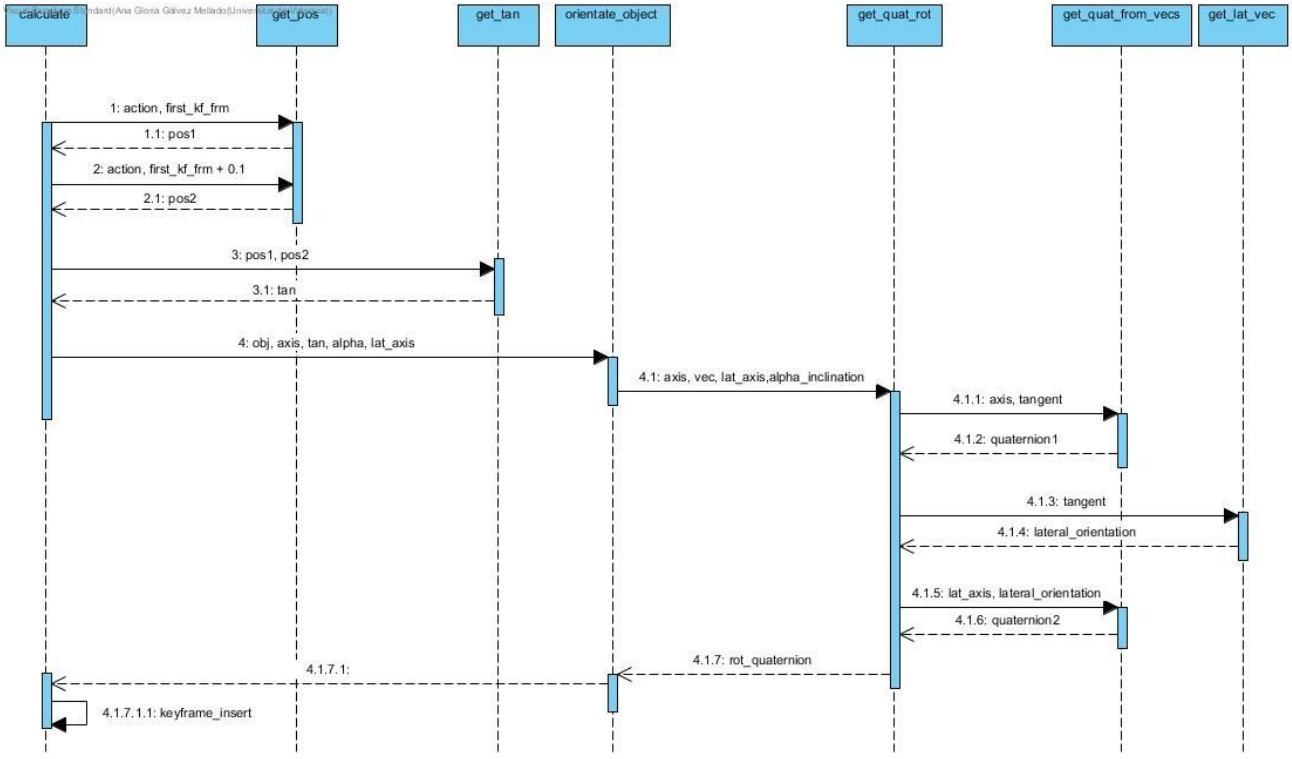
Amplitude function: generate_mov_variation.



Velocity functions: createTable, binarySearch_table.



Orientation functions: get_tan, orientate_object, get_quat_rot, get_lat_vec, get_quat_from_vecs.



General function:

The **get_pos** function [139] returns the position corresponding to the frame entered.

The get_pos function contains:

- Object declarations [155-184].
- Loop for get each location coordinate [186-237] (some parameters initialization and *interpolate_values* call).
- Random oscilation [254-255].

All that functions are invoked inside the **calculate** function [705], which is the main one.

The calculate function contains:

- Object declaration [711-729]: (parameters needed related to the scene and properties).
- Checks [731-744] of the interface properties selected (for avoiding incompatibilities).
- Creation of parameters [745-772]needed on calculations (longitude table, Hermite arrows...).
- Main loop [775-825] (in which functions are invoked and keyframes are inserted).
- A last check [829-843] for changing the Blender interpolation in case of constant velocity.

Interfaz.py

It contains the classes and methods needed to display the interface and pick up the data.

It has:

Calculate class: The operator (the “Calculate Trajectory” button) that calls the function calculate of *script.py.* It also has the poll method with all the needed checks.

Interpolation class: The panel in which all the properties are ordered to be displayed on the interface and its values are stored.

Register and Unregister functions: Definition of the properties and unregister.

Extra code

Some code (already mentioned) was not asked on the projects, but we considered it necessary to be implemented:

- The function cleanVelocityArrows, erases Hermite added arrows and ensured that the arrows get deleted when changing the type of interpolation or even the position of the keyframes.
- Each time *Calculate Trayectory* button is pressed, a new action is created to avoid lose original data and be able to modify parameters and recalculate the trajectory as many times as we want (implemented in *calculate* function).

Resources

Pencil model (used on for compositing). Retrieved from: https://sketchfab.com/3d-models/simple-pencil-9f82b214bc6d4f6eadb1d6d91cf7d983#download

Wood mannequin (subsequently modified). Retrieved from: https://free3d.com/es/modelo-3d/wood-dummy-79325.html

Humanoid (subsequently modified). Retrieved from: https://www.mixamo.com/#/?page=1&query=y+bot&type=Character

Pen cap. Retrieved from: https://free3d.com/3d-model/pen-cap-v1--995348.html

Rubik's cube. Retrieved from:

https://sketchfab.com/3d-models/rubiks-cube-692466217f764fbe9d5c969a30614f68#download

Bee. Retrieved from: https://sketchfab.com/3d-models/flying-bee-4e9ec9d9e4044e6ea6d47614f7daf533

Annex I: User manual

Trajectory Generator Blender Add-On Final Version

User Manual

Authors:
Ana Gloria Gálvez Mellado
Guillem Barceló Orts
Darío Rodríguez Hernández
Bekzat Mukhamedali

Index

Introduction

We have developed a Blender add-on that allows the user to generate a modified trajectory on any object, by introducing a previous one (master trajectory).
Once the user has introduced the location keyframes of the trajectory of an object, they can choose the interpolation method (linear, Hermite or Catmull-Rom + its parameters) and its frequency.
It can be chosen (if wanted) a random oscillation to be applied to the movement and its amplitude (how “exaggerate” it is).
Moreover, the velocity could be set as “constant” (1m/s) or be controlled by inserting distance keyframes.

On this manual it is explained how to use the full add-on.

How to install the Add-On

It is needed to open Blender and select the tab “Edit”;on its submenu select “Preferences ...”

When the Preferences window is opened, select the option “Add-ons” that appears on the menu of the left.

Press the button “Install” that is on top, and browse to your directories to find the addon.zip (**don’t unzip it!**).

A list with all the add-ons installed will be displayed. Enable the checkbox that its on the left side of the add_on and it will be be successfully installed.

How to use the tool

Once the add-on is activated, it should be displayed a new menu on the right editor (the properties editor); concretely, on the Object tab (see Figure 1).

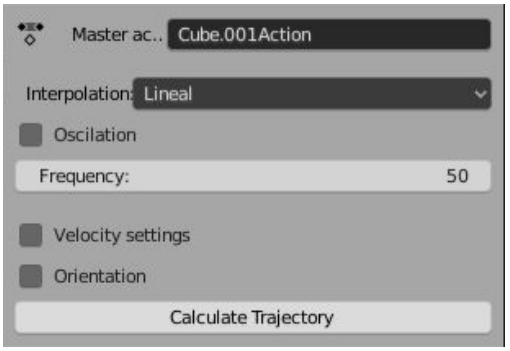


Figure 1

That menu is unfolded by default and it contains the main panel “Trajectory Effect”.

The Trajectory Effect Panel

The Trajectory Effect Panel contains all the modifiable variables (action name, interpolation method, oscillation, frequency, constant velocity and orientation), as well as the “Calculate” button, that generates the new trajectory.

To generate the trajectory correctly, it is important to take in account that **the name written in Action** - see Action- **must match an existing action with, at least, two location keyframes**. It is not necessary for the action to be associated with an object: the program automatically creates and links the new trajectory to the object each time the Calculate Trajectory button is pressed.

By default, Blender automatically interpolates the trajectory; by using this tool, new keyframes would be inserted along all the scene. Those keyframes will maintain the original positions but using the selected interpolation method. (see Figures 2 and 3). To see how to modify the trajectory with a random oscillation, go to “**Oscillation**”.

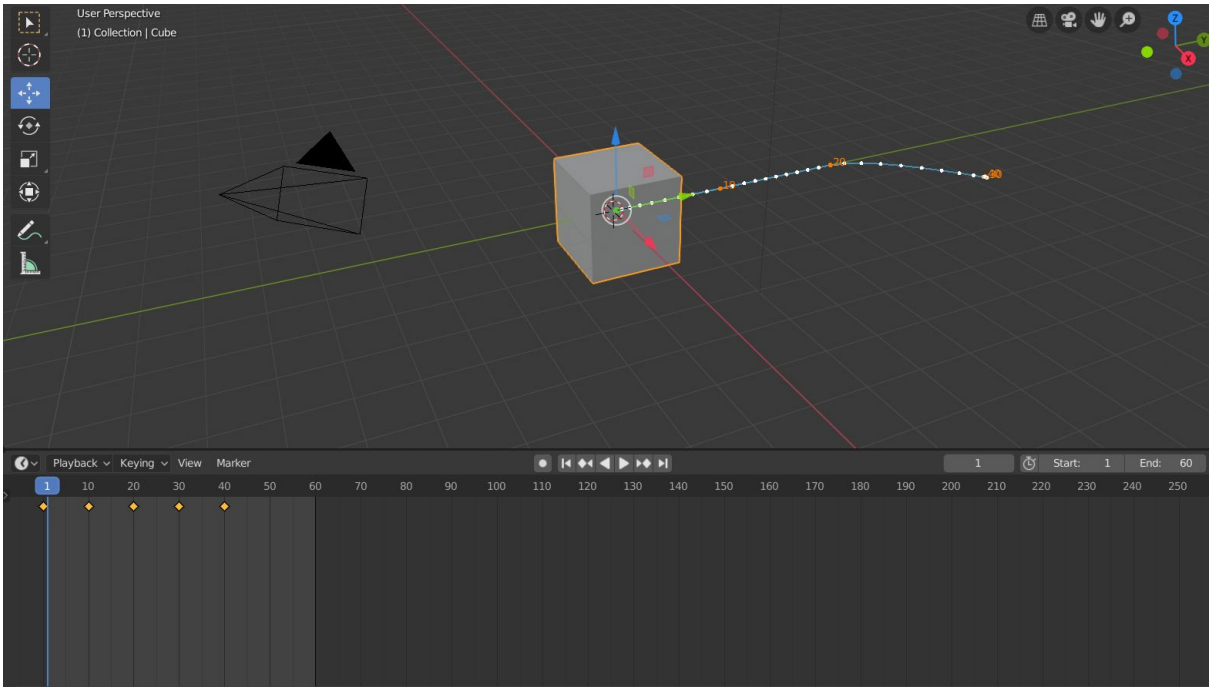


Figure 2

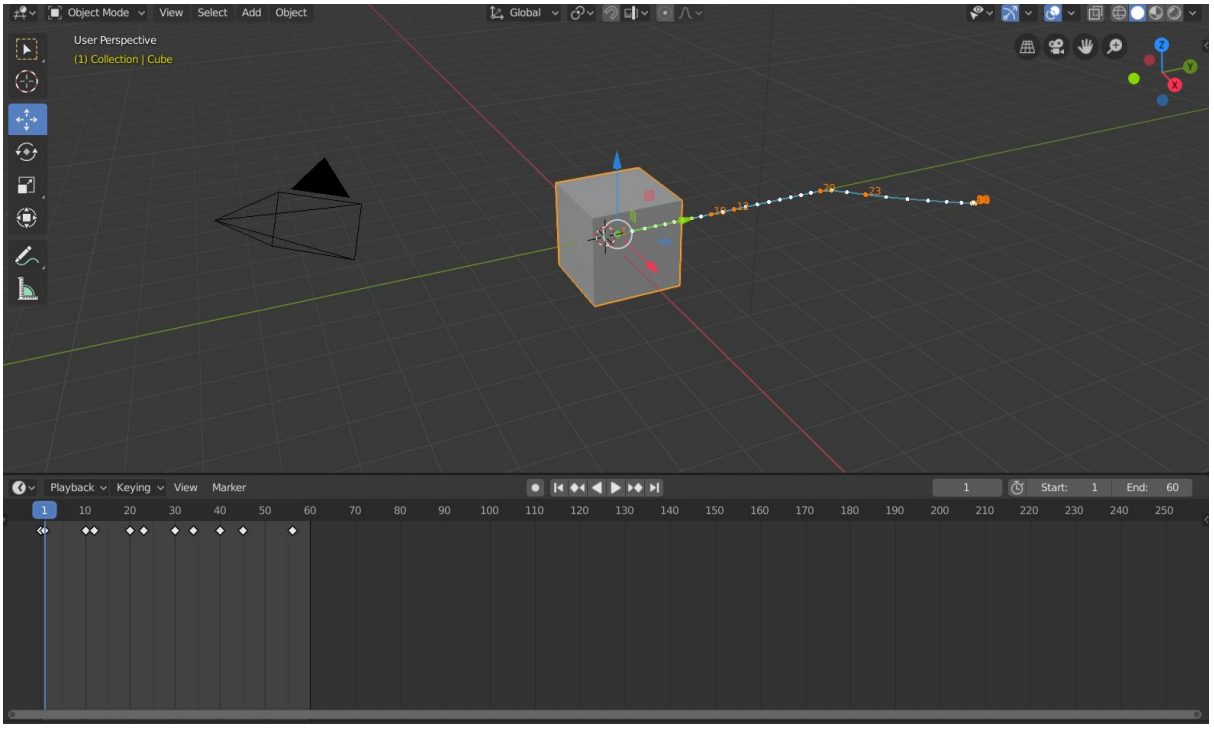


Figure 3

On the next sections we will see how to use each parameter:

Frequency

The frequency is the parameter that controls how many new keyframes will be inserted on the action. The bigger the number is, the more keyframes are inserted. Allowed values go from 1 to 50.
The frequency variation will visually affect the trajectory if it has a random oscillation applied or if it has velocity (see **Oscillation** and **Hermite**).

Interpolation

By clicking on the drop-down interpolation list, we can change the selected interpolation method (linear by default). The different options are explained below:

Linear

It's the basic interpolation method. With this option the only modifiable parameter is the Frequency (and the **Oscillation**).

Hermite

With Hermite interpolation method, a velocity is required for each keyframe. That way, the movement is more controlled. The velocity is modified through the **Velocity Arrows**. On the next section it is explained how to generate and use them.

Velocity Arrows

Once the "Calculate" button is pressed for the first time (having selected Hermite interpolation method), the Velocity Arrows will appear on the 3D viewport (see Figure 4). There must be an arrow for each location keyframe.

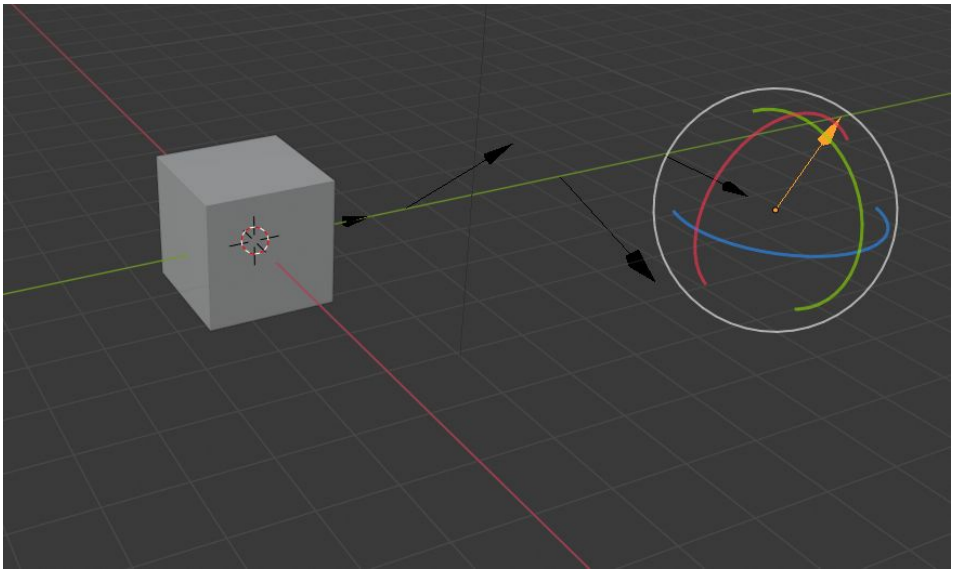


Figure 4

With the Blender commands, the arrows can be modified (scaled and/or rotated) in order to change the velocity value on each keyframe (by default, the velocity arrows value is 2, and they point to the

direction of the next keyframe of the trajectory). The bigger the arrow is, the bigger will be the value of the velocity.
Next times the button “Calculate” is pressed, the trajectory will be updated with the new velocity values.

Catmull-Rom

By using the Catmull-Rom interpolation method, the trajectory velocity will be modified (as in **Hermite** interpolation method) but, in this case, it is automatically assigned. The parameter **Tau** must be modified in order to control the velocity globally faster or slower (it is immediately displayed when Catmun-Roll interpolation is selected - see Figure 5).

Tau also can be used to locally control the velocity. For doing that, just insert Tau keyframes by setting the mouse above Tau value and pressing I key (on the keyboard). With two or more keyframes, a new fcurve will be created on the object and the Tau value will be interpolated between its keyframes.

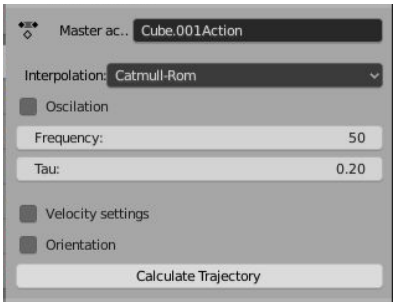


Figure 5

Tau

This parameter is a factor by which the velocity is calculated. To appreciate changes but not to critically modify the original trajectory, Tau values go from 0 from 0.5.

Oscillation

Although the interpolation methods respect the original trajectory, the Oscillation mode allows you to make little variations on the position. The inserted keyframes will be slightly moved on any x,y,z axis direction in order to simulate “random” displacements along the trajectory.
It is important to take into account that those variations will be applied on each new keyframe. It means that the “amount” of random movements is directly controlled with the **Frequency** parameter. (The higher the frequency is, the more oscillations it will be).

The oscillation is not selected by default. If we select the checkbox, it will be active, and a new parameter called “Amplitude” (see **Amplitude**) will appear below (see Figure 6).

Amplitude

As explained before, the oscillation is a random variation of the original trajectory on each keyframe. Through the amplitude it is possible to control how large (big values) or subtle (small values) we want it to be. The amplitude values go from 0 to 10.

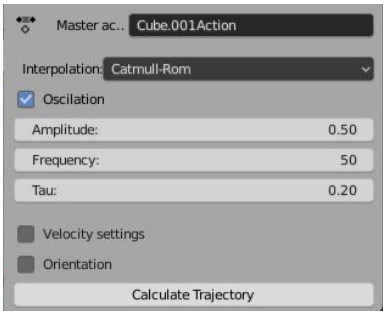


Figure 6

Action Field

On that field it must be written which action (master trajectory) must be used to generate the desired actions. **If it is not written or it does not contain location keyframes, the program won't be able to run.**

Velocity Settings

If the Velocity Settings checkbox is selected two new unselected checkbox will appear on the panel (see Figure 7) those checkboxes are the options provided for modifying the trajectory velocity (see Constant Velocity and Control Velocity).
If both Constant Velocity and Control Velocity are selected, by default, Control Velocity will be applied.

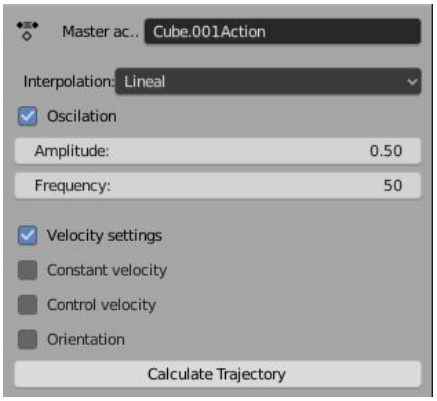


Figure 7

Constant Velocity

By selecting this checkbox (see Figure 8), the trajectory will be automatically rearranged over time to achieve a constant velocity. The value of the velocity (m/s) can be changed on the Value slider. (Values go from 1 to infinite).

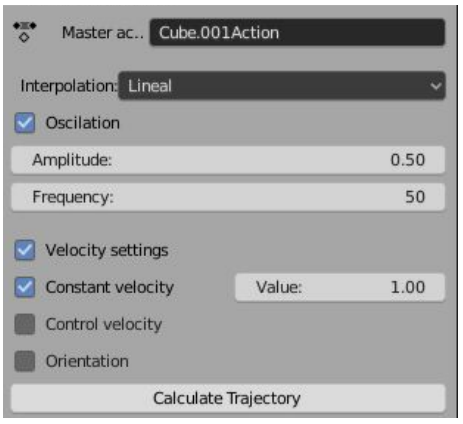


Figure 8

Control Velocity

If the user selects the checkbox “Control Velocity”, the slider “Current Distance” will be displayed (see figure 9).

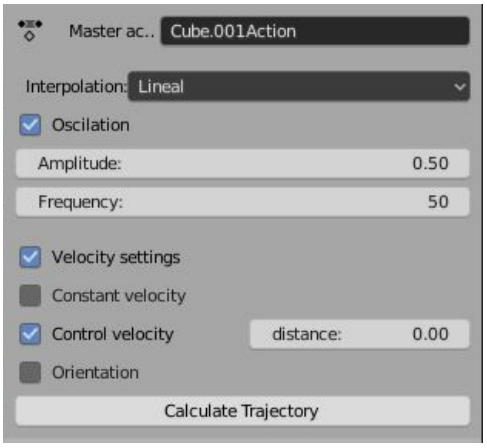


Figure 9

If it is the first time “Control Velocity” is selected, the values of distance will go from 0 to infinite. From the second time, the max value of “Current Distance” will be the total length of the trajectory.

By using **the main trajectory** (about which the interpolation will be applied), distance keyframes could be inserted (allocate the mouse above “Current Distance” and click I).

That way, it can be controlled the traveled distance among time (at frame frm, the Current Distance is the selected on the slider). So velocity (distance through time) is modified.

Orientation

With the “Orientation” function, the object could be oriented so it rotates by following the trajectory (e.g.: a car, its nose must be following the road all the time).
It also has a function that allows to set a “wrapping angle” and insert keyframes to control it (e.g.: when a plane rotates in the air, it’s not completely straight, it has a little inertial rotation)

If the Orientation checkbox is selected, two new selectors and an unselected checkbox will appear on the panel (see Figure 10).

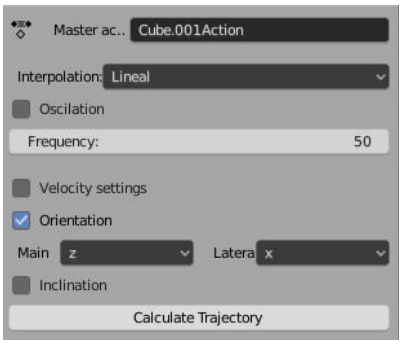


Figure 10

Those selectors are the “main axis” selector, it must be selected the matching front face axis of the axis (so it will be oriented face to the trajectory).
The “lateral axis” selector must match the lateral side of the object.
It is not possible to select the same (positive or negative) axis as the main axis AND the lateral axis.

If the orientation is selected, frequency always will be 50 (full frequency) and oscillation will be unselected. (Orientation and Oscillation didn’t make sense together).

Inclination

If the “inclination” checkbox is selected, a new field must be displayed (see Figure 11).
On this field, “Wrapping angle”, the desired inclination angle must be written **in degrees**, (values go from 0 to 360). It is possible to insert different wrapping angles as keyframes, just allocate the mouse behind the field, and press I.

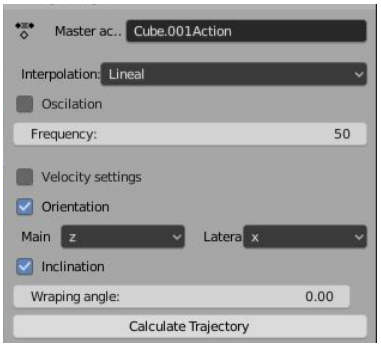


Figure 11

Object Actions

Each time “Calculate Trajectory” button is pressed, a new action will be generated and linked to the object. The rest of the actions must be unlinked but still exist (by using the Dope Sheet in Blender).

The program could be used as many times as wanted and the same trajectory can be applied to different objects on the same scene multiple times.

The master trajectory (the one that is written on Action Field) is never modified by our program. But the user can re-adjusts its parameters and it will still work.

Improvements in v.2.

-Hermite arrows are deleted when changing to other interpolation methods. (Now, they are associated to the object and not to each trajectory).
-They also automatically refresh when changing between different master trajectories.
-Tau keyframes can be inserted (see Catmull-Rom for more details).
-Now the program generates a new action each time. Moreover, the main action is never modified. (See Object Actions for more details).

Improvements in v.3.

(When orientation is selected):
-The object is originally orientated looking at the direction of the trajectory, although it is firstly still.
-Also, when it stops, it stills on the last orientation it had when moving (although the tangent is 0 because there is no movement).
-We have included negative axis (-X, -Y, -Z) so the user can easily orientate objects that have inverse axis (e.g.: the car object given has the nose pointing to the -Z axis).
-It is impossible to select the same axis (e.g.: X or -X) as the main axis and the lateral axis (the poll is implemented).

Use examples

Maximum and minimum values description		
Amplitude	**Frequency**	**Effect**
0 (min.)	50	If there's no amplitude, the trajectory is normally interpolated (there's no visible effect).
10 (max.)	25	With the max amplitude, the movement generated is so exaggerated (even near the start and end keyframes the position is visibly altered).
5	50 (max.)	The trajectory is completely altered. It seems like the object is moving in a very "nervous" speedy way. Sometimes, the movement is so wide that it costs to focus on the movement.

To see the demos of the following table, click on the link:
https://www.youtube.com/watch?v=b8hggLi2Tus&feature=youtu.be

Video demos				
Title	**Amplitude**	**Frequency**	**Interpolation method**	**Description**
#1 Bottle Oscillation in water	0.5	25	Catmull-Rom Tau: 0.4	It simulates the movement of a drifting bottle in the water.
#2 Phone vibrating	0.09	50	Catmull-Rom Tau: 0.2	With all the keyframes on the same position, it simulates an object vibrating.
#3 Sliding	0	40	Catmull-Rom Tau: 0.2	It simulates the movement of a object sliding on a sequence of slide planes.
#4 Dodging obstacles	5	25 (med.)	Linear	The object moves like it is avoiding obstacles on its way. It softly makes like "s" movements.
#5 A mosquito flying	1.5	48	Hermite. Default velocity.	The object moves like a flying insect (a fly, a bee, …). It is moving randomly but softly at the same time.
#6 Camera flash effect	10 (max.)	50 (max.)	Linear	The objects has a completely random movement. The movement is always around the same 3D space (where the original trajectory was). Iit seems like some lights effect (flickering).