TAMPERE UNIVERSITY
OF APPLIED SCIENCES

Creating Motion Graphics in Blender with Animation Nodes

Rolands Tiss

Bachelor's thesis
November 2017
Degree Programme in Media and Arts

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media and Arts

TISS, ROLANDS:
Creating Motion Graphics in Blender with Animation Nodes

Bachelor's thesis 75 pages, appendices 5 pages
November 2017

In the past decades motion graphics has evolved dramatically. From analog signal synthesizing and processing to keyframing and digital computer graphics. Among the evolution of technology also motion graphics development methods have changed. If 40 years ago there was a Scanimate, an analog computer animation system, then now there exist dozens of computer graphics development software.

As a rule of thumb, paid software should deliver higher quality content and offer more technical features than Open Source Software but is it possible to deliver competitive results with a free software? The goal of this thesis is to examine open source software - Blender's and Animation Nodes' potential in motion graphics.

Blender has been an Open Source Software since October 13, 2002 and keyframe animation features within it. Even though node based procedural animation techniques were known way before keyframing, Blender was lacking these features until 2015 when Animation Nodes appeared. Two animation techniques, but what differs them from each other in the same working environment? The purpose of this thesis is to compare these techniques; to find their strengths and weaknesses.

To compare Animation Nodes features to the already well known keyframe animation method, two by appearance similar animations were created using two mentioned techniques.

The case study revealed differences in motion graphics development pipeline when using two different animation methods. Also, it was found that almost the same outcome can be achieved using different techniques from which each has its cons and pros.

The findings verify that Animation Nodes is a great and powerful add-on for Blender. Instead of replacing already implemented animation features in Blender, Animation Nodes expands their capabilities to create competitive motion graphics for free.

Keywords: motion graphics, animation nodes, animation, blender, 3d

CONTENTS

GLOSSARY

3D	Three dimensional
n	Non-negative integer
AN	Animation Nodes
Active node	The last selected node in Node Editor
Active object	The last selected object in Object Mode and is shown in a lighter orange color than other selected objects
Add-on	A piece of software that enhances and customizes software's applications
Blender	Full feature 3D creation software
CGI	Computer Generated Image
Interpolation	A method of constructing new data points within the range of a discrete set of known data points
OSS	Open-source software
Python	A High-level programming language
Shader	A program that runs on a graphics card which processes 3D scene during the rendering pipeline, controlling lighting and shading effects, before image is drawn on a computer's screen
Vector Algebra	Algebraic operations in vector spaces
VFX	Visual effects

1 INTRODUCTION

In the recent decades motion graphics has conquered a significant role in the animation industry. The 3D animation market is predicted to grow from USD 12.01 Billion in 2017 to USD 21.05 Billion by 2022 (MarketsandMarkets™, 2017). Nowadays one cannot imagine commercials, TV shows, movies, music videos, video games or other digital medium which does not contain a piece of digitally made animation.

Because of the tremendous demand from the animation industry, software developers have created dozens of digital animation production tools. Regardless of the developer and software suites they offer, most of them share similar features like 3D modelling, Animation, Rendering, 3D Tracking etc (Slick, J. 2016).

Roy Weinstock, Scanimate animator, tells that the origins of computer motion graphics started in 1960 in Lee Harrisons' attic in Blue Bell, Pennsylvania, United States (Scanimate: The Origins of Computer Motion Graphics – Film 2016). According to HistoryofInformation, the first hybrid graphic animation computer – ANIMAC was built by Lee Harrison III in 1962, which was predecessor for Scanimate. Scanimate was analog computer animation system designed and built by Harrison and the Computer Image Corporation in Denver. From around 1969 up to computer era in mid-1980's it was the main motion graphics creation tool in television and film industries. (HistoryofInformation)

Scanimate became a standard solution in 1970's and 1980's. According to Scanimate: The Origins of Computer Motion Graphics – Film (2016), Dave Sieg and Roy Weinstock tell that Scanimate was used for show openings, commercials, TV shows and other animations. The principle of Scanimate system is analog signal processing. Altogether there were 8 systems built and only one is still in working condition owned by Scanimate engineer Dave Sieg. This system was a significant turning point for motion graphics. Engineers who made Scanimate not only built the whole system, they also created formulas of video effects that nowadays are used in CGI by every motion graphics artist. (Scanimate: The Origins of Computer Motion Graphics – Film 2016)

Nowadays there is more than just one motion graphics creation tool. Despite the amount and variety of offered animation development software, there have been set a few standards in animation industry. The most frequently used software suites in motion graphics studios are 3ds Max and Maya from Autodesk Inc., Adobe After Effects from Autodesk Inc., Cinema4D from Maxon Computers GmbH or Houdini from Side Effects Software Inc. Without a doubt, all these tools are powerful and do their tasks at the top quality but for small companies and freelancers they might not be the most cost-efficient solutions for commercial purposes because of their relatively high licensing cost (Autodesk Inc.; Maxon Computers GmbH; Side Effects Software Inc.; Adobe Systems Inc.). Table 1 lists most popular 3D software licensing cost on 24.10.2017.

TABLE 1: Most used 3D software price comparison on 24.10.2017.

No.	Software Package	Company	License Cost Options
1	3ds Max	Autodesk Inc.	242.00 €/month 1936.00 €/1 year 5808.00 €/3 years
2	Maya	Autodesk Inc.	242.00 €/month 1936.00 €/1 year 5808.00 €/3 years
3	Cinema 4D Broadcast R19	Maxon Computers GmbH	1547.00 € Standalone
4	Adobe After Effects	Adobe Systems Inc.	Starting from 24.19 €/month
5	Houdini Indie	Side Effects Software Inc.	199 $/1 year

According to Slick (2017), among all above mentioned expensive industry standards there also exist several 3D software suites that are available for free: Blender from Blender Foundation, SketchUp from Trimble Inc., Sculptris from Pixologix Inc., Daz Studio from Daz Productions Inc. etc. It is important to point out that most of mentioned software above are not full feature 3D creation suites except Blender from Blender Foundation. It is one of the best full feature open source 3D software packages at the time. It has gained popularity among CG artists rapidly in past few years with its equivalent features that other developers offer for excessive cost. (Slick 2017)

It is completely free of charge for any kind of commercial and/or non-commercial purpose with strong community and developer support all over the world. Blender is a full feature 3D creation tool, which offers 3D modelling, rigging, shading, animation, particle simulation etc. (Blender Foundation: About)

Thanks to Jaques Lucke, a Blender developer from Henningsdorf in Germany, Blender has gotten its own add-on for procedural effect creation – Animation Nodes. This extra feature is also an OSS which does not cost anything and is freely available online for everyone.

The goal of this thesis is to investigate AN in depth. Most of the new companies, artists or freelancers do not have funding for expensive software. Therefore, the main question is – what Animation Nodes is capable of?

In this thesis, the basics of Animation Nodes will be covered. There will be given an overview of general AN principles and implementations. Also included is a practical case study to compare two different animation techniques – Animation Nodes and keyframe animation method.

This thesis is aimed for everyone who has an interest in 3D visualisation, motion graphics and details related to procedural animation effect creation. The focus is put on Animation Nodes add-on; therefore, basic knowledge of Blender and Vector Algebra is required.

2 WHAT IS MOTION GRAPHICS

Motion Graphics is a specific animation discipline which combines different media forms such as graphic design, animation or/and visual effects depending on the desired outcome. All these disciplines share common methodologies and techniques to be created but there are major differences that help to distinguish motion graphics from classical animation or VFX.

Asset translation on screen over time does not define what is motion graphics and how it differs from classical animation or visual effects. Motion graphics is derived from graphic design, inheriting simplicity and abstraction but is not limited to use any other visual elements. The most significant difference between motion graphics and animation is an extra layer of information. If animation and visual effects are created more for storytelling and viewing experience then motion graphics' purpose is to carry information that helps to explain an idea or concept. (Crook & Beare 2015, 10)

According to Crook and Beare (2015), classic animation and motion graphics share a lot in common: the tools they are created with, development methods, animation principles and many others. The attribute that differs these two disciplines is the purpose. If animated film's or short film's main purpose is to tell a story, send a message and entertain, then motion graphics purpose is to explain information. It is important to keep in mind that animation is derived from illustration but motion graphics comes from graphic design which is meant to deliver information rather than emotions. (Crook & Beare 2015, 12)

Crook and Beare (2015) emphasise that, the difference between motion graphics and VFX is that in VFX footage and images are composed in a way that they look like one piece while in motion graphics it is not always necessary. Both can share the same tools and methods to be created, such as particle generators and physics simulations but again the attribute that differentiate these disciplines is the purpose. (Crook & Beare 2015, 14)

3 NODES

3.1 Nodes in Blender

In computer science nodes are known as an approach of organizing, structuring and combining data network (TechTerms). Also, nodes are known as a visual scripting method which is more artist friendly for complex mathematical function creation. This technique allows one to avoid traditional scripting until a certain point. Nodes can be reconsidered as coding machines, which process raw data and output the final product (Steinmetz & Gottfried 2014, 9). Simplified node operating principle is shown in figure 1. Official Blender releases have already implemented node-based systems for material, texture and compositing data. They all consist of nodes and links.

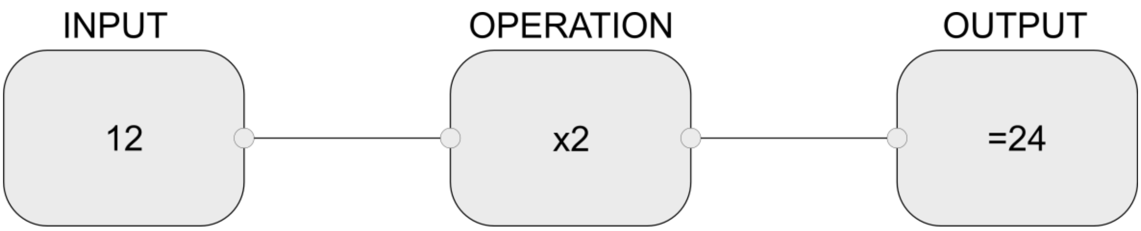
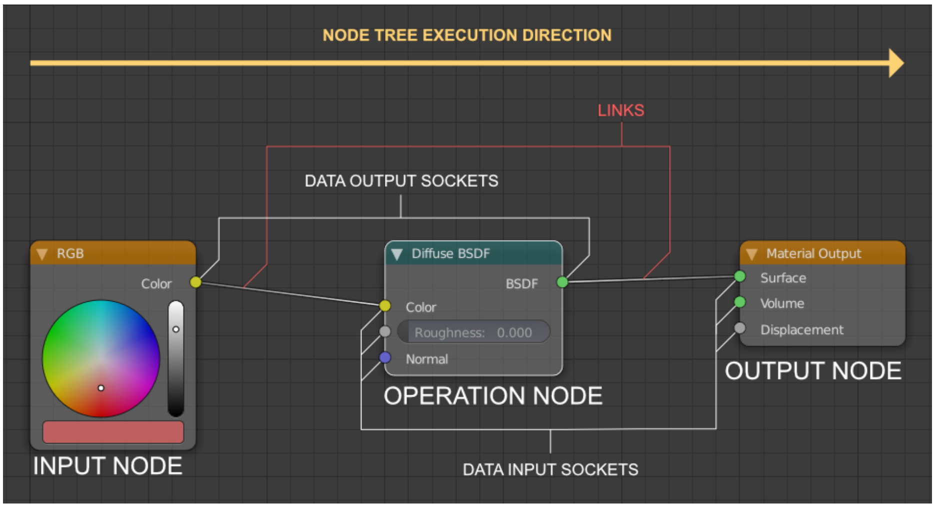


FIGURE 1. Simplified node operating principle (Based on Lampel 2015, 39)

Lampel (2015) explains, that each node represents a block of functions it performs on data. There are three types of nodes: input, operation and output nodes. (Lampel 2015, 39.) Input nodes provide the node system with data from various sources. Operation node executes operations based on data it receives from previous nodes or predefined values and outputs the result to the next node in the graph. Output nodes receive data from the rest of the node system and assigns them to linked dependencies. Link is a connection between two nodes and is represented as a line. An example of a Material node system in Blender is shown in picture 1.



PICTURE 1. Simple Material node system in Blender (Tiss, 2017)

Picture 1 represents a Material node tree in Blender that creates a simple diffuse material. The RGB node defines the color data, which is passed to the Diffuse BSDF node. Diffuse BSDF node process color data and outputs a shader based on the color data it receives from the RGB node. From the Diffuse BSDF node, data is passed to the Material Output node which creates final look of shader basing on three inputs – Surface, Volume, Displacement. Node system as any other program has its command execution sequence. Unlike in written scripting languages where code is structured from top to bottom, node graphs in Blender commonly are composed and read from left to right. It is also possible to make collapsible node groups to keep node trees more structured. In Blender node systems are converted into scripts which are given to render engine to do all the mathematics in the background. Using nodes takes away the obligation to know programming languages and their libraries which helps artists significantly in producing art. Therefore, visual scripting in CG is becoming more popular. Node systems have been already implemented in such video editing, compositing and 3D software as:

- Natron from Natron project
- Houdini from Side Effects Software Inc.
- 3DS Max from Autodesk Inc.
- Substance Designer from Allegorithmic
- Nuke from The Foundry Visionmongers Ltd.

3.2 Animation Nodes

In official Blender releases, there are few implemented tools for creating procedural animation effects - modifiers. Modifiers are very flexible in adding randomness and automating different processes but at the same time they are fairly limited in feature adjustment. Therefore, Jacques Lucke, a Computer Science student and a Blender add-on developer from Henningsdorf in Germany decided to take an effort. In 2015, he started the development of Animation Nodes to expand Blender capabilities.

Lucke (2017) reveals that he had several ideas for motion graphics add-ons for Blender, but many of them would share the same base, therefore he decided to combine all the ideas into one, besides maintaining one add-on seemed more efficient than many. Making Animation Nodes as a node based system seemed the most logical way how to implement all the features that were. (Lucke, 2017)

Animation Nodes is an Open Source node based scripting system designed for motion graphics in Blender. It is designed for procedural animation effect creation in Blender. The add-on is mainly used to create motion graphics but is capable of a lot more – it is as capable as Python scripting language is. In general, Animation Nodes is visualizing mathematical functions. (Lucke, 2017)

Currently (October 2017) Animation Nodes v2.0.4 is officially released. Since this add-on is based on executing mathematical functions, the only limitations are the users' creativity, knowledge of Math and problem-solving skills. According to Lucke (2017), it is very hard to enumerate cons and pros of AN because of the fact that it works very similar to programming. Many artists find themselves very confused in-front of node graphs while others highly appreciate the approach of Animation Nodes. It is each artists' individual preference. (Lucke, 2017)

It is important to keep in mind that Animation Nodes is developed continuously. Blender community and artists affect this open source project significantly. That means that suggestions and criticism is highly valued during the development process and is implemented into upcoming releases. This tight cooperation between the supporting community and Jacques Lucke ensures that the add-on will become user friendlier and more powerful in the future.

4 ANIMATION NODES

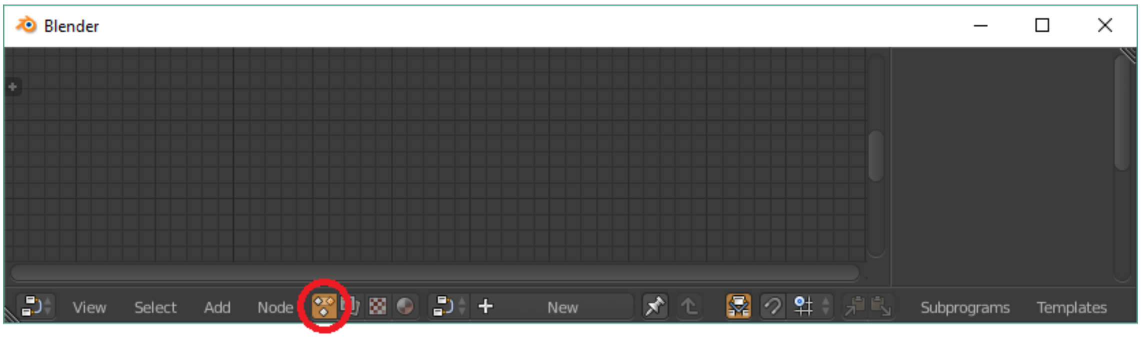
4.1 Installing Animation Nodes

Animation Nodes v2.0.4 is not included into original Blender releases, its latest version must be downloaded from an external source, provided by the developer (https://github.com/JacquesLucke/animation_nodes/releases/tag/v2.0). Installing an external add-on can be done by following the standard procedure described in the official Blender User's manual (https://docs.blender.org/manual/en/dev/preferences/addons.html) or following instructions in official Animation Nodes User Guide (http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/install/install.html).

.

4.2 Workspace of Animation Nodes

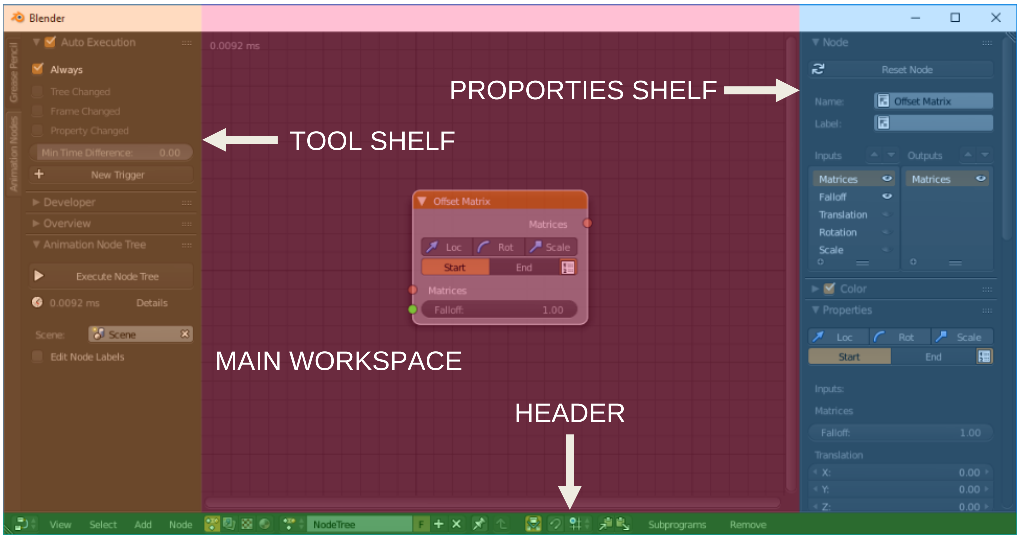
When Animation Nodes is successfully installed, in the Node Editor header new button appears, highlighted in picture 2.



PICTURE 2. An empty Animation Nodes workspace in the Node Editor window (Tiss, 2017)

"The Node Editor is used with node-based workflows. The node tree type can be changed using the buttons in the Node editor header." (Blender Foundation: Node Editor - Introduction).

The user interface of Animation Nodes is withheld like other node tree types found in Blender. It consists of four main components, shown in picture 3: tool shelf, properties shelf, header and main workspace where to place nodes. Also, keyboard shortcut combinations, such as toggling the tool shelf or properties shelf, are kept the same as in other node tree types to maintain consistent workflow over different node system modes in Blender.



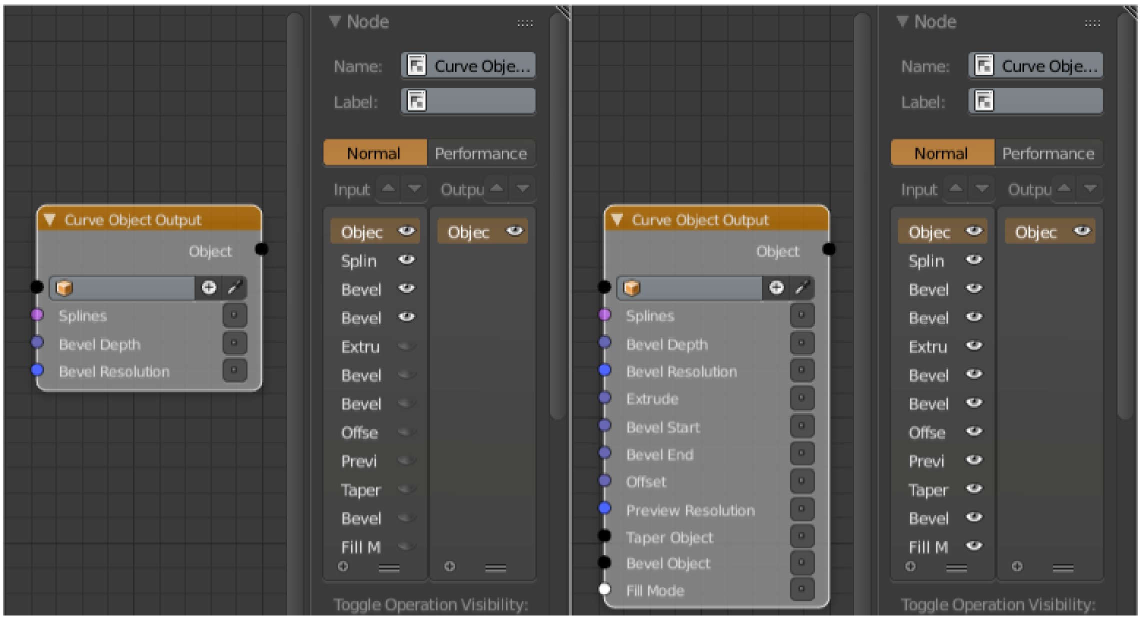
PICTURE 3. Structure of Animation Nodes node editor (Tiss, 2017)

4.3 Nodes

Nodes in Animation Nodes do not differ visually from nodes used in other types of node systems found in Blender, like Material, Texture or Compositing nodes. Despite the common node features among other node systems, there are several vital details to pay extra attention while working specifically with nodes in AN.

4.3.1 Hidden node sockets

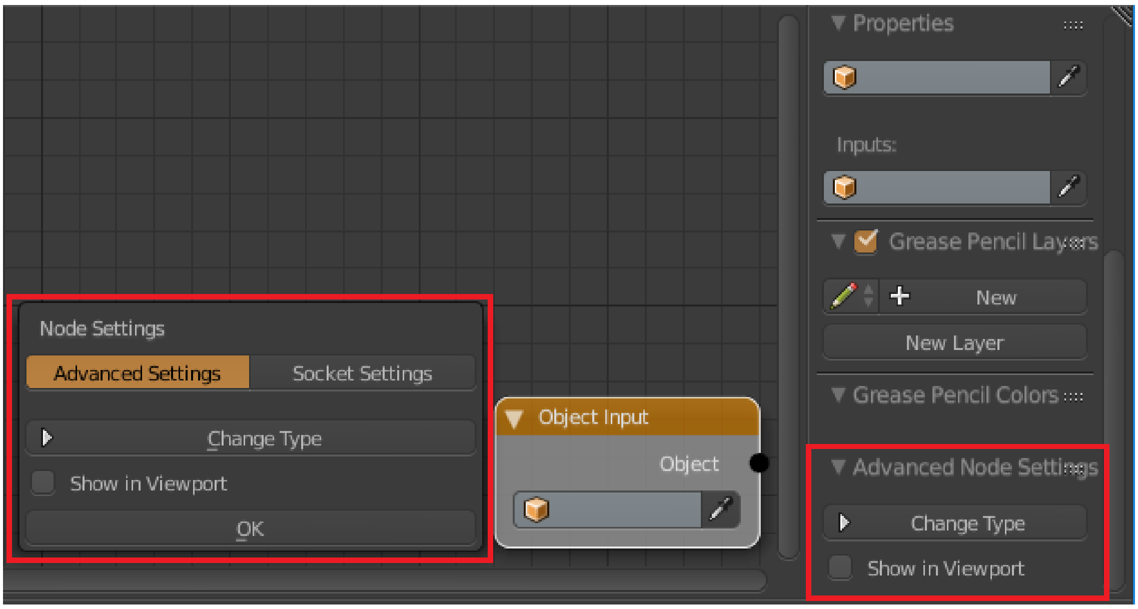
There are nodes that can handle many different data and therefore some nodes have hidden data input and data output sockets. This is for keeping node graphs in order without unused node sockets. The difference between the same node with hidden and unhidden data input sockets is represented in picture 4. Available node sockets of active node only can be found in the Properties Shelf or by pressing the keyboard shortcut key “u”. By default, most used node sockets are unhidden.



PICTURE 4. The difference between nodes with hidden data input sockets on the left and unhidden data input sockets on the right (Tiss, 2017)

4.3.2 Advanced Node Settings

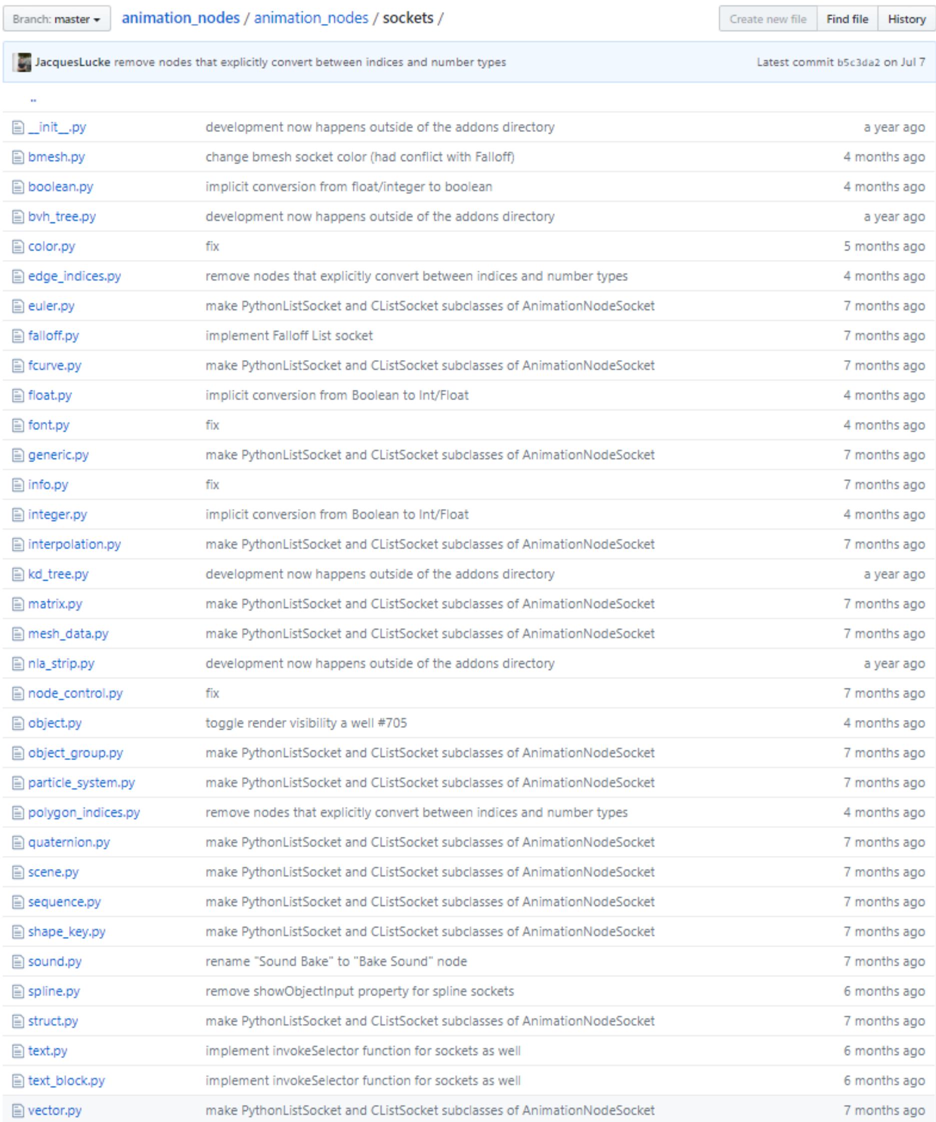
Several nodes have advanced settings that can extend or change functionality of the node but are not used frequently (Animation Nodes: Interface 2017). Therefore, these settings are placed outside the node to maintain clean structure of certain nodes. Advanced Node Settings of an active node can be found in the Properties Shelf or by pressing keyboard shortcut “u”, these settings are highlighted in picture 5.



PICTURE 5. Advanced Node Settings (Tiss, 2017)

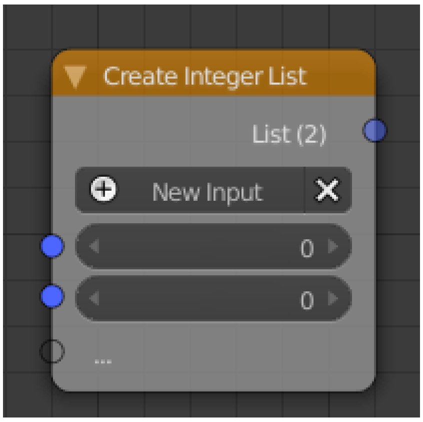
4.3.3 Socket types

Sockets serve as connection points between two or more nodes. Even though Animation Nodes is a visual scripting system and contains different data types already known in programming, for users it is more important to understand socket types. According to Animation Nodes manual, there are 36 different socket types (Animation Nodes: Socket Types 2015). The information presented in Animation Nodes manual is outdated, in AN v2.0.4 socket type list has changed and all socket types can be found in Animation Nodes' GitHub repository. Most of them have list option but currently (24.10.2017.) they are not listed in the manual. Picture 6 represents the latest socket type list.

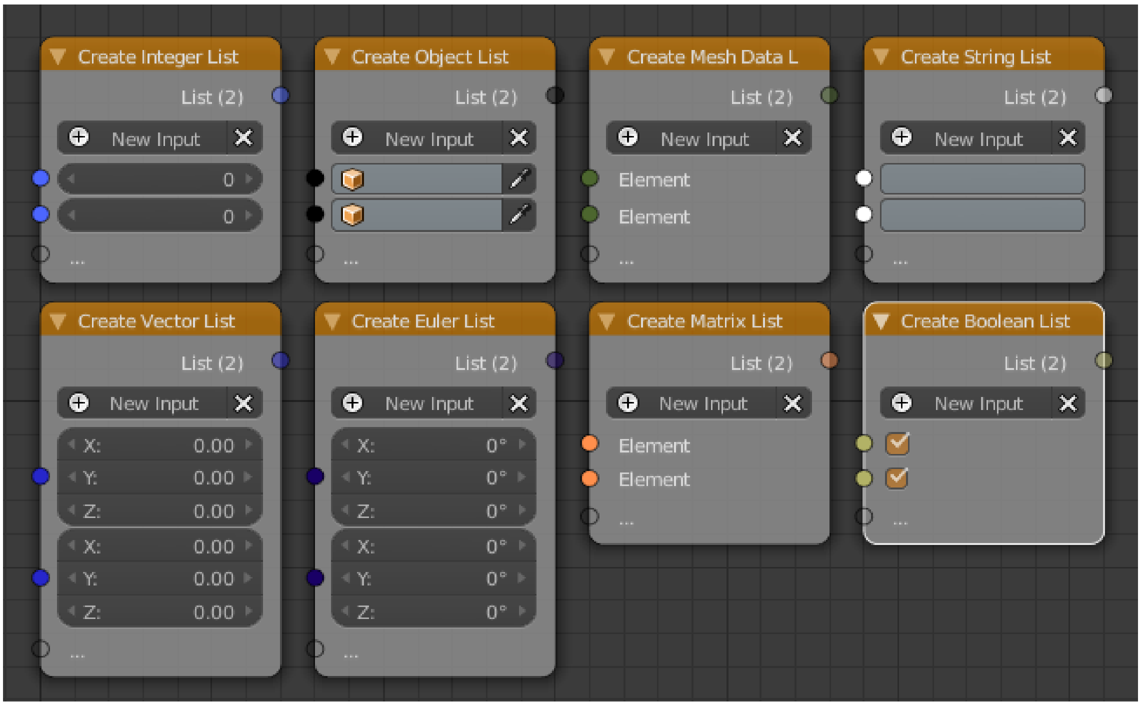


PICTURE 6: All the socket types in Animation Nodes v2.0.4 (Screenshot from Animation Nodes' GitHub Repository, 2017)

Each socket type represents collection of data type it returns in the data output or requires in the data input socket for correct node functionality. Socket types are colour coded and named to ease the recognition of their data type. Many socket types have two options: single data and certain data list option. In picture 7 is shown the Create Integer List node. All sockets are coloured light blue to indicate that they are Integer type sockets but input sockets are solid light blue while the output socket is also light blue but partly transparent. Transparency means that the socket requires or outputs certain data list. This rule applies to all socket types which are eligible for list option. In picture 8 are shown several nodes with different socket types.

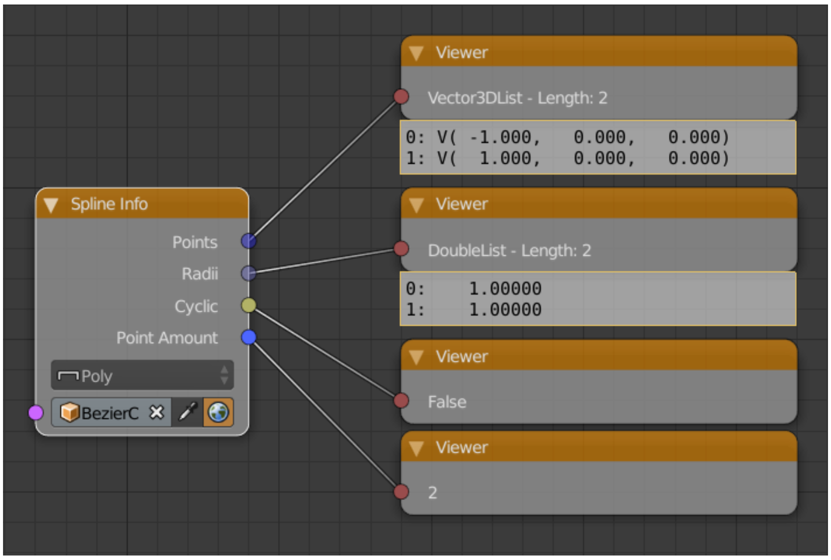


PICTURE 7. The Create Input List node (Tiss, 2017)



PICTURE 8. Nodes with different socket types (Tiss, 2017)

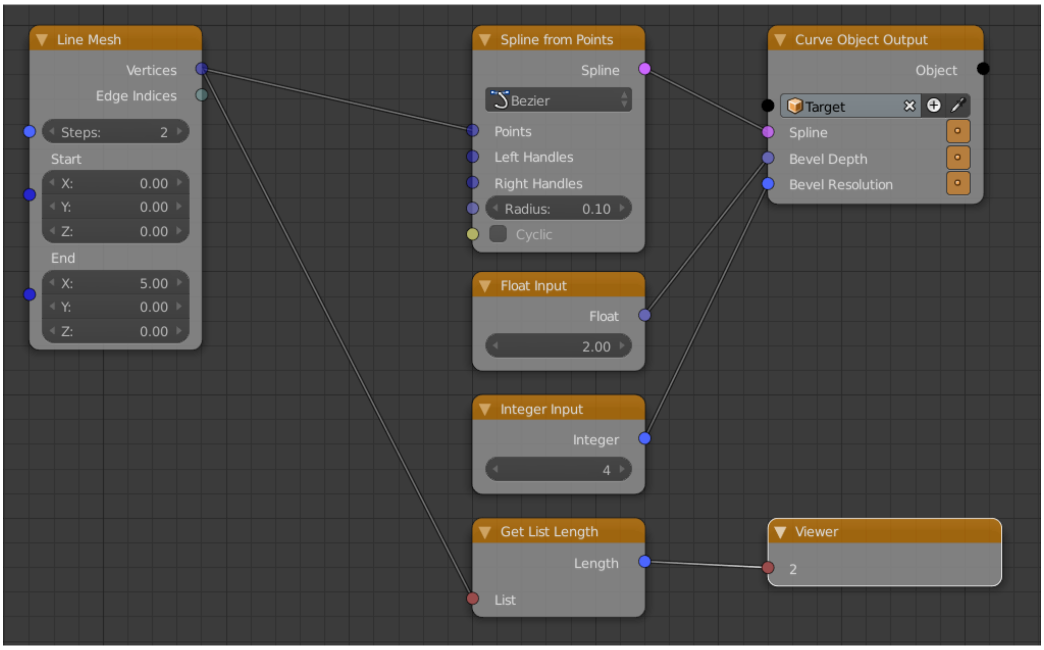
One node can contain different socket types as well. In picture 9, Spline Info node has only one Spline input socket and four outputs: Points, Radii, Cyclic and Point Amount. By their colours, it is possible to recognize what kind of socket type they are and what type of data they handle. Spline input socket gathers data from Spline object and separates information into four different data types. Points socket return spline's point location vector list, Radii returns each points radius as float list, Cyclic socket outputs Boolean value weather the spline is cyclic or not and Point Amount returns a single integer number with point amount in the spline, all these data are shown in Viewer nodes in picture 9.



PICTURE 9. Spline Info node with different socket types (Tiss, 2017)

4.3.4 Connecting nodes

Thanks to the fact that node socket types are colour coded it is easy to connect different nodes together. The main rule is that the input socket must receive data from the same colour output socket that comes from previous node. In picture 10 is shown an example of connected node tree. The Get List Length node is an exception. It is because the Get List Length node does not process data from the list but returns the amount of entries in it (Animation Nodes: Get List Length 2017).



PICTURE 10. Node connection example (Tiss, 2017)

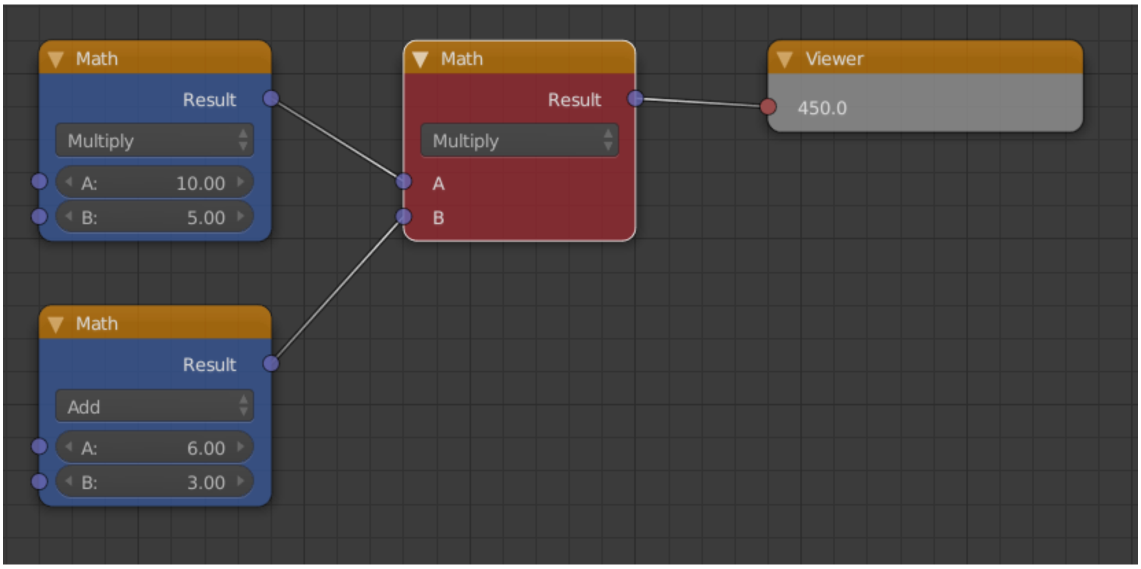
4.3.5 Input, function and output nodes

Nodes in Animation Nodes cannot really be divided into specific data input, data output or data function node types because their role varies depending on their location in the node tree.

In picture 11 is shown mathematical equation

$$(10 \times 5) \times (6 + 3) \ = 450$$

made in Animation Nodes. Three Math nodes are placed in various locations in the node tree. In this case, the blue Math nodes serve as data input nodes because the user manually defines their values. Meanwhile the red Math node processes the data provided by blue Math nodes. Therefore, it is not needed to divide nodes into different node types in Animation Nodes. Even though there are several unambiguous output nodes their purpose is comprehensible, such as the Viewer nodes.



PICTURE 11. Math nodes as different node types (Tiss, 2017)

4.4 Animation Nodes execution in Blender

Blender, like any other computer program, consists of many different script units which are responsible for certain task flawless execution like viewport and UI drawing, animation execution etc. All these script units are structured in a specific order to ensure proper and most effective program operation. This data execution structure has its routine and each necessary data block, it consists of, is called and accomplished once per cycle execution. This approach provides that the latest data is represented on the computer's screen in the shortest time.

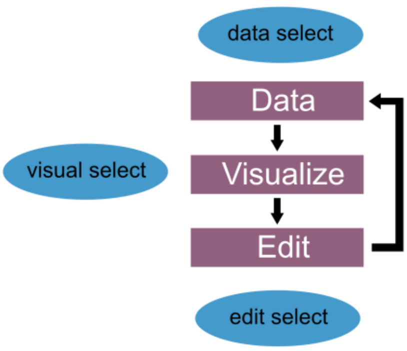


FIGURE 2. Blender's most basic structure (Roosendaal, 2011)

"This diagram depicts Blender's most basic structure. The "data-visualize-edit" cycle is in its core, and separated from the GUI (blue) which works on three distinctive levels." (Roosendaal, 2011).

According to Roosendaal (2011), the cycle represented in figure 2 shows three main execution blocks of Blender – Data, Visualize and Edit block. The "data-visualize-edit" cycle is executed repeatedly while Blender is running. It starts at the same time when Blender is launched. It starts with Data block, reading all the data from pre-defined start-up file and sends the information to Visualize stage where the gathered data is sent to the computer's screen. In the following Edit section, all changes made by user are applied. Then updated information is passed back to the Data block and the cycle starts all over again. The "data-visualize-edit" cycle is repeated until Blender is closed. (Roosendaal 2011)

Animation Nodes is nothing else but collection of script units in the hierarchy of Blender. It has its own place in the program's structure, from where it gets called and compiled once per each "data-visualize-edit" cycle execution, if default AN execution settings are not changed. Otherwise Animation Nodes is executed as defined by user.

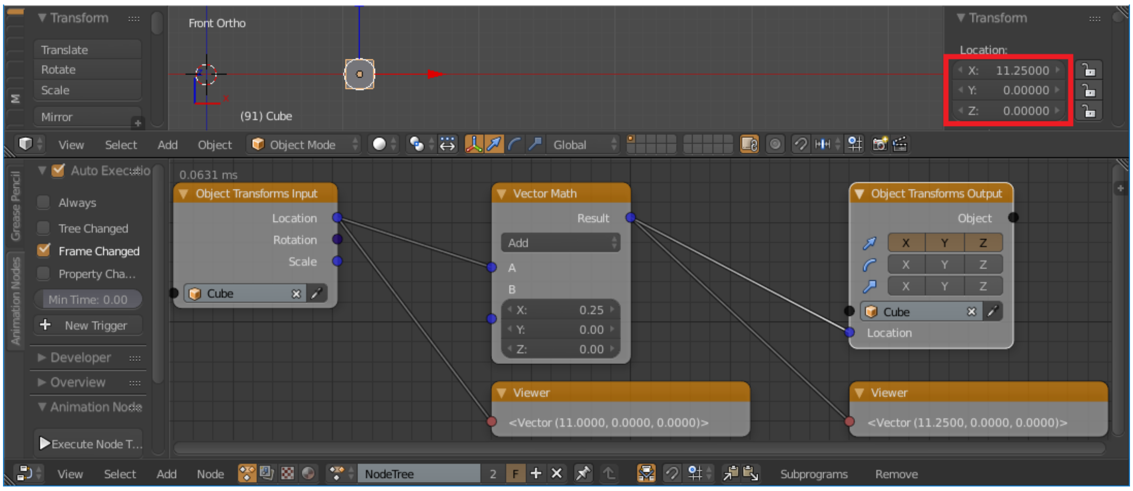
To be sure about Animation Nodes execution in the "data-visualize-edit" cycle, simple feedback loop can be created in the node tree editor, shown in picture 12. The node tree consists of five nodes:

- Object Transform Input node
- Vector Math node
- Object Transform Output node
- Two Viewer nodes.

Object Transform Input node reads the location of "Cube" and returns three-dimensional vector value $\vec{u} = (11.0, 0.0, 0.0)$ which is displayed in the first Viewer node from the left. Then the read location value ($\vec{u}$) is passed to the Vector Math node, which in this case does vector addition and returns the value of $\vec{u} + \vec{v}$, where $\vec{v} = (0.25, 0.0, 0.0)$ is defined value in the Vector Math node by the user.

$$\vec{u} + \vec{v} = \left(u_x + v_x,\ u_y + v_y,\ u_z + v_z\right)$$
$$\vec{u} + \vec{v} = (11.25, 0.0, 0.0)$$

Vector Math node returns value $\vec{u} + \vec{v}$ which is displayed in the second Viewer node from the left. After mathematical operation execution, the updated value is assigned as location vector value to the same "Cube". If AN execution settings stay unchanged ("Always" by default) then Cube's location value along x axis will be incremented by 0.25 (as defined) per each "data-visualize-edit" cycle execution.



PICTURE 12. AN placement in the Blender's structure by example (Tiss, 2017)

4.5 Main programs and Subprograms

In general, each node system which is made with Animation Nodes is an executable program in visual form. Ideologically by isolating/separating AN from the rest of Blender's structure, it is easy to comprehend it as an independent program which loops and executes its functions infinitely. Node graphs in Animation Nodes are built up from two main building blocks: main programs and subprograms. Visually all main node trees are coloured neutral grey while each subprogram has its own color which is assigned randomly. In figure 3 is shown the ideological execution process of Animation Nodes in time.

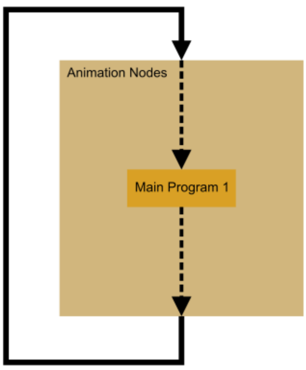
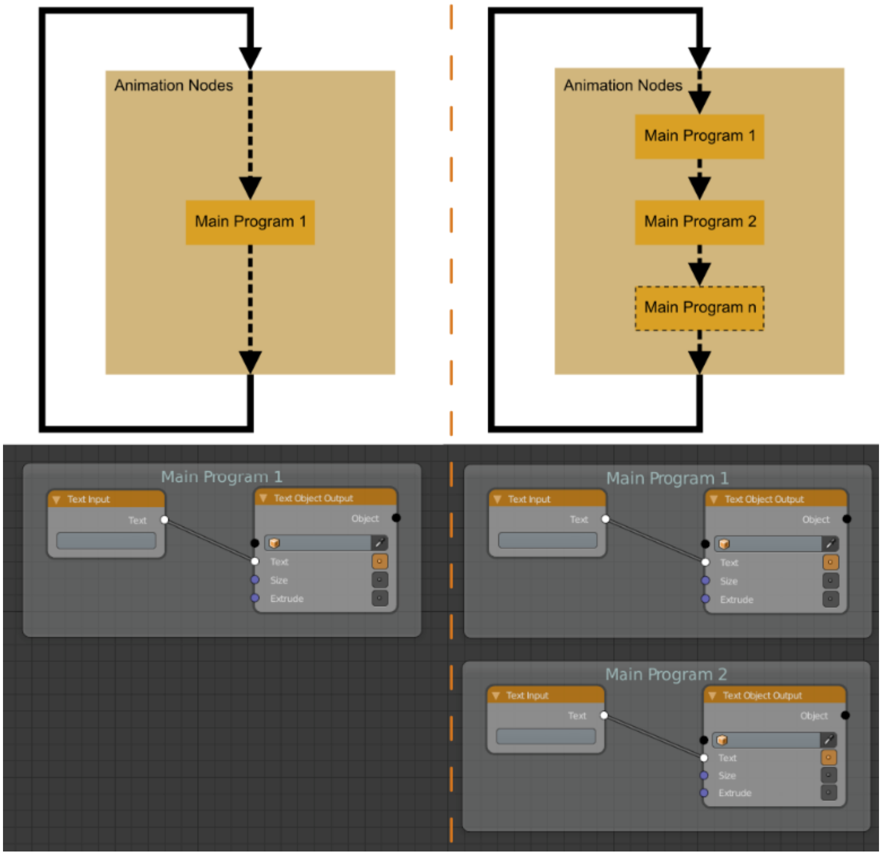


FIGURE 3. Ideological Animation Nodes execution cycle (Tiss, 2017)

4.5.1 Main programs

Each working node tree in Animation Nodes must have at least one Main program. It is executed only once in each cycle. It is possible to have more than one Main program in one node graph. In that case these programs are stacked and executed consecutively. Their execution order is undefined but in most cases, it makes no difference. Picture 13 illustrates the execution of a single Main program's execution on the left and multiple Main program execution schemes on the right with practical examples accordingly below.



PICTURE 13. Main Program's execution scheme with practical example (Tiss, 2017)

4.5.2 Subprograms

According to Animation Nodes manual (2017), every node tree made with Animation Nodes can be considered as a program. In programming, to repeat specific command lines with different parameters, it is good practice to use functions or loops. The same rules apply to Animation Nodes. (Animation Nodes: Subprograms 2017)

Subprograms allow user to create specific block of functions that can be called from anywhere in the node tree with corresponding Invoke Subprogram node. Each subprogram has its own unifying color. All nodes in the same subprogram share the same node color, it eases the recognition of each nodes' affiliation. In Animation Nodes v2.0.4 there are three subprogram types available: loops, groups and scripts.

4.5.2.1 Loop

Loop is a special subprogram type in Animation Nodes that can be repeated more than once in each Blender's main cycle, depending on the count of iterations, it is similar to loops in programming (Animation Nodes: Loop 2017). Its trait is iteration count feature which defines, how many times the loop will be repeated. Model of Loop's execution in Animation Nodes is shown in figure 4.

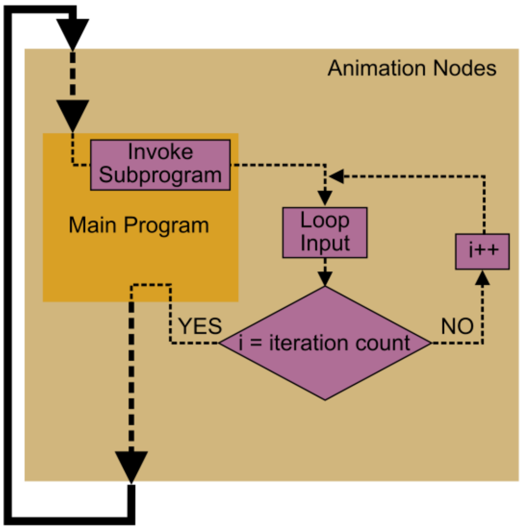
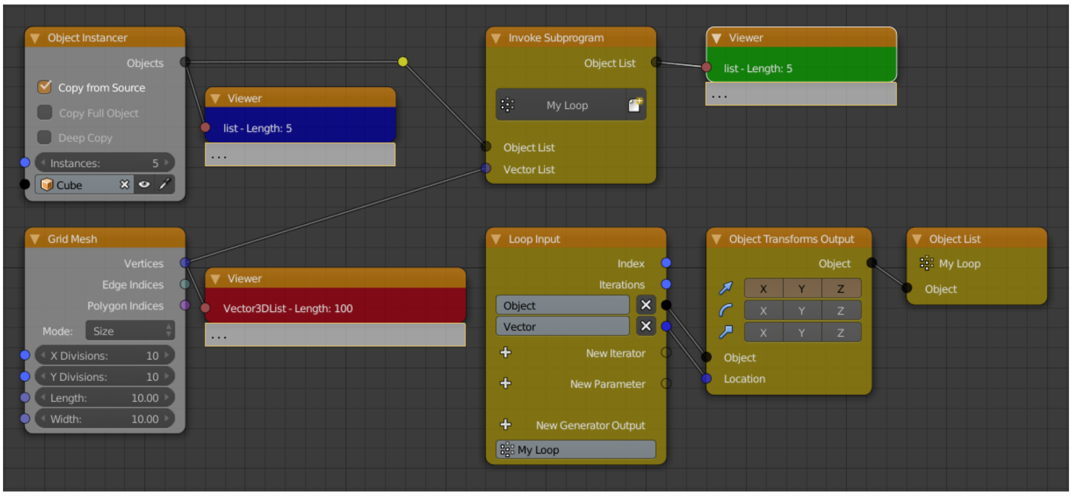


FIGURE 4. Model of Loop's execution in Animation Nodes (Tiss, 2017)

Iteration amount can be defined by a user by manually entering the value in the Iterations input socket of the Invoke Subprogram node, highlighted in picture 14, or by any type of List. Note that if there are more than one list passed as a parameter to the Invoke Subprogram node, then the shortest of them will define the iteration amount. In most cases the length of different lists, prepared for loops, are even. In picture 15 is shown a practical example, where the iteration count is defined by the shortest list. The Loop Input node iterates through lists and returns single values gathered from input lists at corresponding index. The Index is a sequence number of the current iteration.

PICTURE 14. Manual iteration value set for Loop Input node (Tiss, 2017)



PICTURE 15. Defining loop's iteration amount with shortest list (Tiss, 2017)

4.5.2.2 Group

Group in Animation Nodes can be reconsidered as function in programming (Animation Nodes: Group 2017). It can be reused with different parameters without copying all consisting nodes. This approach allows efficiently the reuse collection of specific node trees. Group can be called from anywhere in the node tree by using the corresponding Invoke Subprogram node. Group is executed once per each call. Group's Invoke Subprogram node can also be implemented into loop's cycle and vice versa. In figure 5 is illustrated group's execution in Animation Nodes.

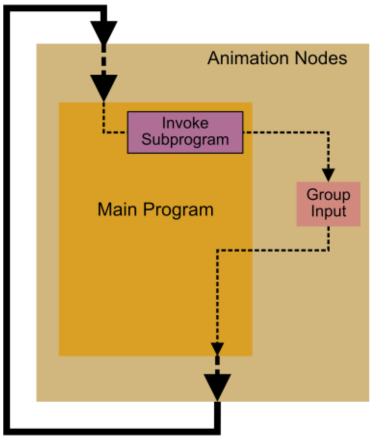
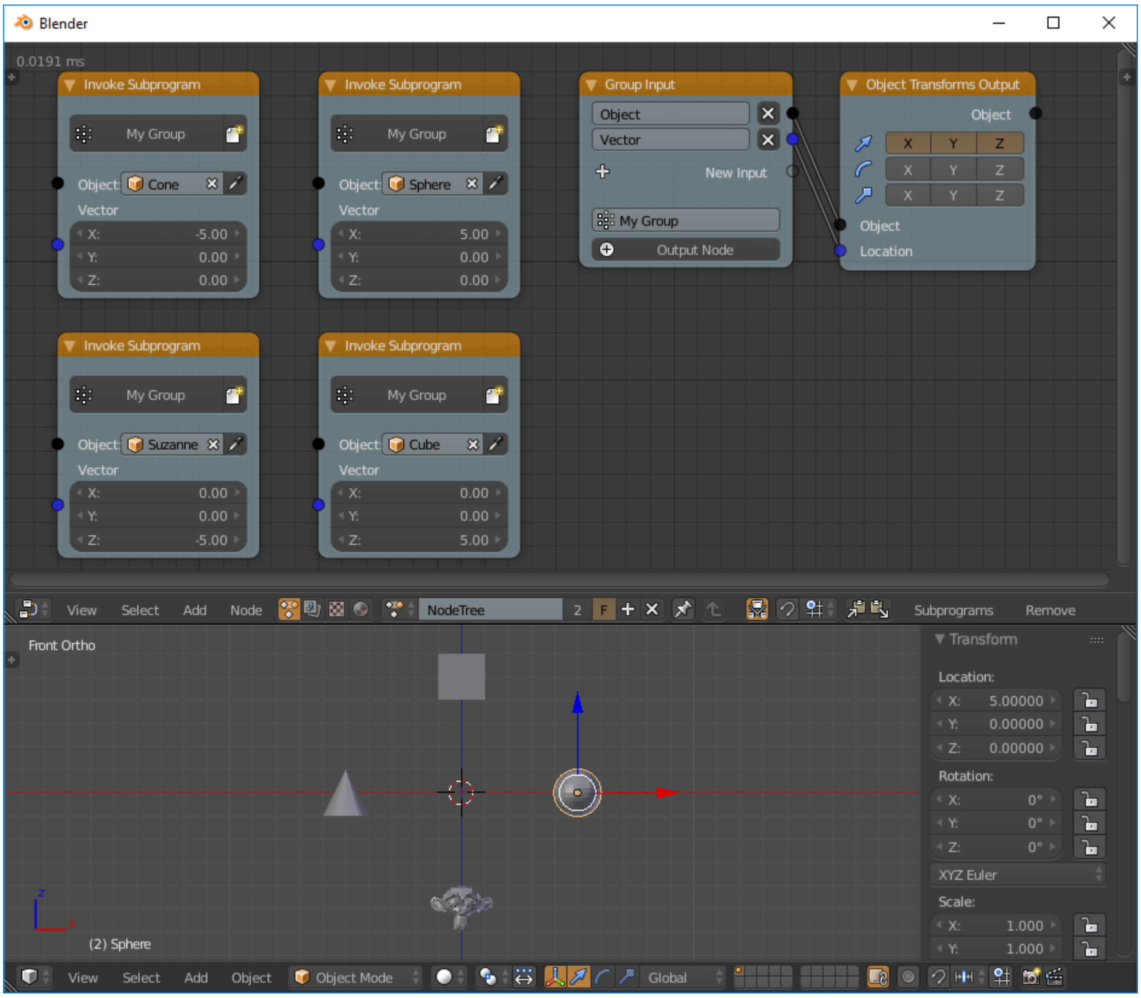


FIGURE 5. Model of Group's execution in Animation Nodes (Tiss, 2017)

In picture 16 is shown a practical example, how Group can be reused with different parameters. Each Invoke Subprogram node has different parameters: different objects and vector values. The Group Input node executes its functions with submitted data individually. Therefore, each object is placed at defined coordinates.



PICTURE 16. Practical example of Group's reusability in Animation Nodes (Tiss, 2017)

4.5.2.3 Scripts

Even though Animation Nodes is considered as visual scripting language and offers many different nodes, sometimes users can be limited by their functionality, therefore script node allows user to write Python code and execute it with the related Invoke Subprogram node (Animation Nodes: Script 2017). The Script node is considered as advanced feature and requires knowledge of Python programming language.

5 CASE STUDY: ANIMATION DEVELOPMENT COMPARISON USING ANIMATION NODES AND KEYFRAME ANIMATION TECHNIQUES

To show an example of the previously described Animation Nodes' main program and subprogram features and compare them to the keyframe animation method, a simple scene, that can be animated using both techniques almost equally well, was made. To evaluate features between AN and the keyframe animation method, the test scene, shown in picture 17, is animated in two different techniques:

- Using Animation Nodes only
- Using the keyframe animation method only

PICTURE 17: Test scene output graphics, made with Blender (Tiss, 2017)

Beside modelling, shading and lighting some preparation tasks, which ease animation workflow, were done beforehand. To make the Group Input nodes work correctly and to ease object selection when using the keyframe animation method, object groups were created beforehand. To ensure correct object scaling and rotation, their pivot points were changed by necessity. "VentBlade.n" objects were parented to corresponding "VentBox.n" object to exclude manual animation of "VentBlade.n" translation.

The task for this animation is to build up a part of a city. The most frequently animated object parameters are:

- Object translation
- Object rotation around local axis
- Object scale
- Object hide/unhide

Objects are grouped by their nature to ease their selection during the animation process and to separate them from other objects, because grouped objects can share the same animation settings with different timing. This is achievable by creating a reusable node group with Animation Nodes or by sharing Action Strips when using keyframe animation method.

Even though there are many objects that must be animated individually, thanks to animation reusability, it is possible to split the whole animation task in eight subtasks:

- Location, Rotation and Scale animation
- Object hiding and unhiding from the scene and render
- “CurbStone” animation
- “Mug” Location and Rotation animation
- “VentBlade.n” Rotation
- Spline bevel for “Sign_AnimationNodes_Circle” animation
- “Neon_Blue” and “Neon_Pink” material wiggle animation
- Text animation

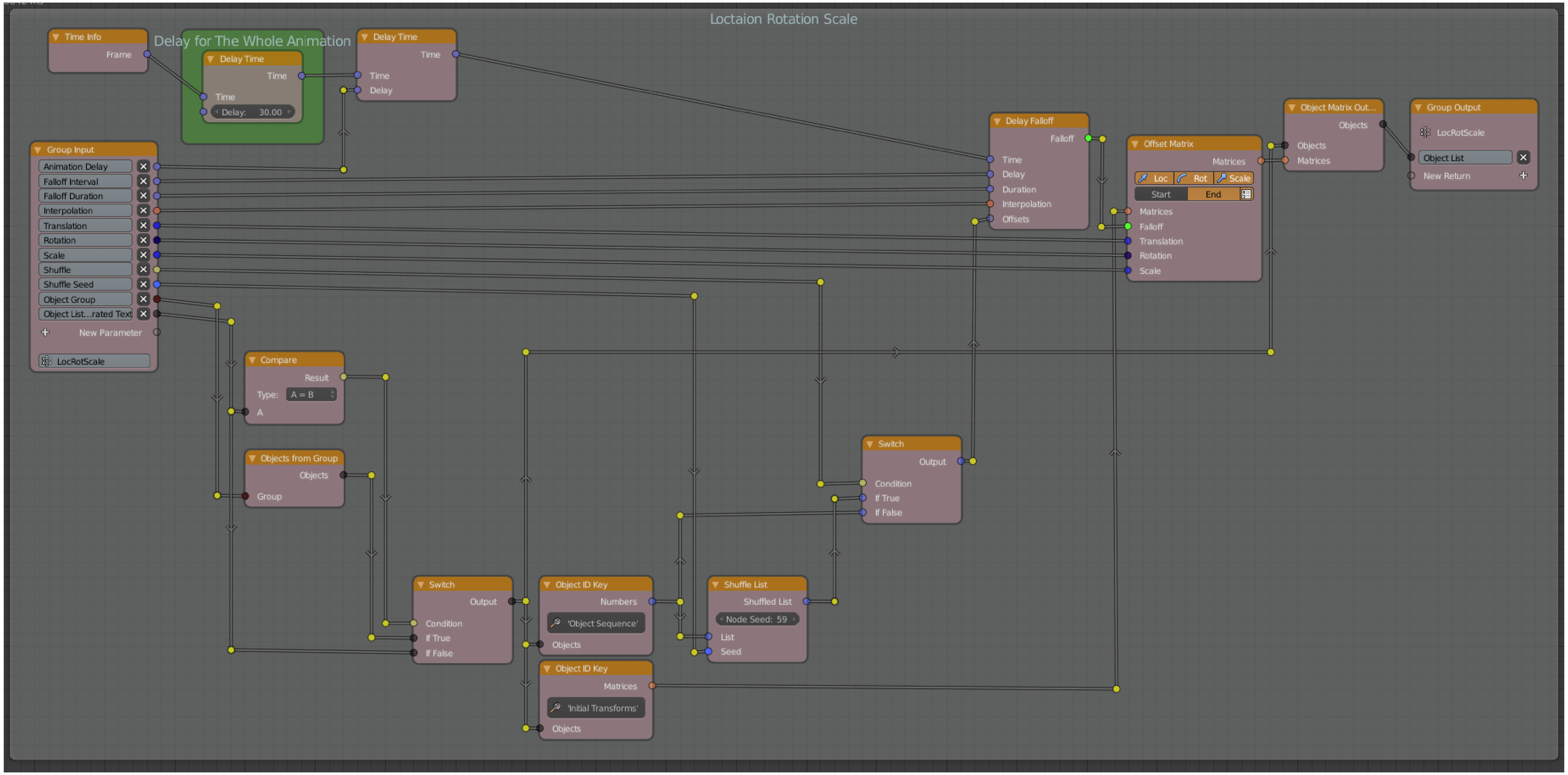
5.1 Location, Rotation and Scale animation

5.1.1 Location, Rotation and Scale animation using Animation Nodes only

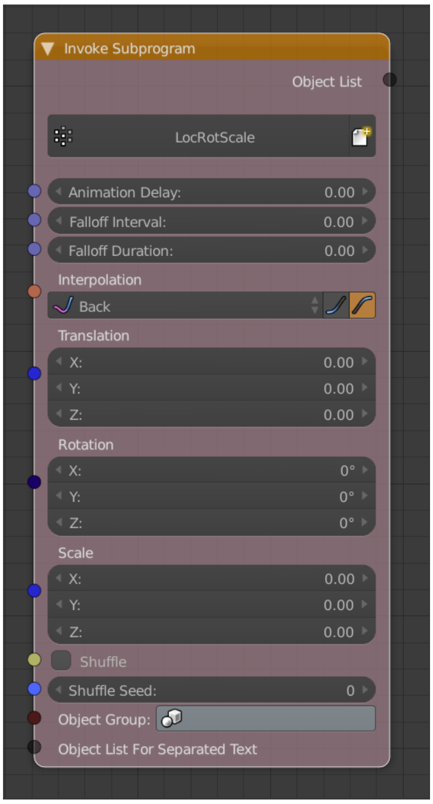
For the animation task, most object’s location, rotation and scale values must be animated, therefore when using AN, big advantage is reusability of a node group with certain configurable input sockets. Before creating a reusable node group, it is important to reconsider all the necessary attributes and options which are going to be animated and also would ease tweaking animation afterwards. Therefore, it is good practice to define all animation attributes beforehand and only after that start making the node tree. To create a reusable Location, Rotation and Scale animation node group with few extra settings, following features must be included in the group:

- When does object, animation start – Animation Delay
- Interval between same group object's animations – Falloff Interval
- For how long each animation will last – Falloff Duration
- Interpolation mode
- Object Delta Transform – Translation
- Object Delta Rotation – Rotation
- Object Delta Scale – Scale
- Use initialized or random object animation sequence – Shuffle
- Change Shuffle seed
- Object group input to get object list and theit "Initial Transforms" matrices
- Object List input for text animation

The Structure of a "LocRotScale" node group is represented in picture 18. The "LocRotScale" node group is created in order to animate all the attributes mentioned above. In picture 19 is shown the "LocRotScale" group's Invoke Subprogram node which calls and executes the whole "LocRotScale" node tree with settings set in the Invoke Subprogam's node. Thus, there is no need to copy the whole node tree. This node group lets the user animate all objects in the same object group automatically according to set settings. By reusing this node group, it is possible to accomplish most of the whole animation task.

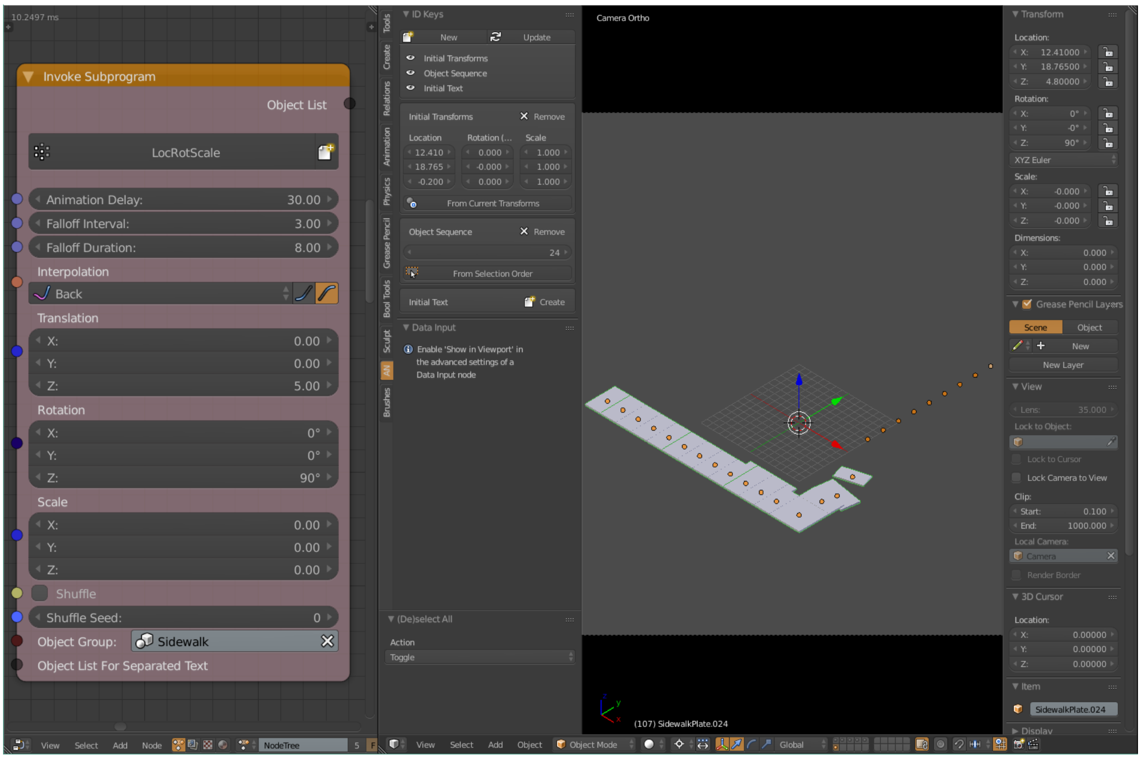


PICTURE 18: The “LocRotScale” animation node group made with AN in Blender (Tiss, 2017)



PICTURE 19: The "LocRotScale" group's Invoke Subprogram node made with AN in Blender (Tiss, 2017)

In picture 20 is shown a "Sidewalk" object group's animation settings on the left and the scene on the right side. The advantage of using the "LocRotScale" node group is that this single group animates all the objects which are in the same object group using the same animation settings between these objects. In the "Sidewalk" object group there are 25 separate objects. This approach eases and fastens the animation process by excluding manual action strip adjustment which is described in section 5.1.2.



PICTURE 20: “Sidewalk” object group’s animation with set setting in the Node Tree (Tiss, 2017)

In figure 6 is represented Animation Delay, Falloff Interval and Falloff Duration socket’s approximate look on timeline in Blender. An Animation Delay socket defines after how many frames animation of the “Sidewalk” object group will start. It must be kept in mind that this value is added to the “Delay for The Whole Animation” delay node’s value in the “LocRotScale” group’s structure, highlighted with a green frame in picture 18. “Falloff Interval” defines how many frames there are between beginning of two object animations. Meanwhile “Falloff Duration” defines the length of each object’s animation in frames.

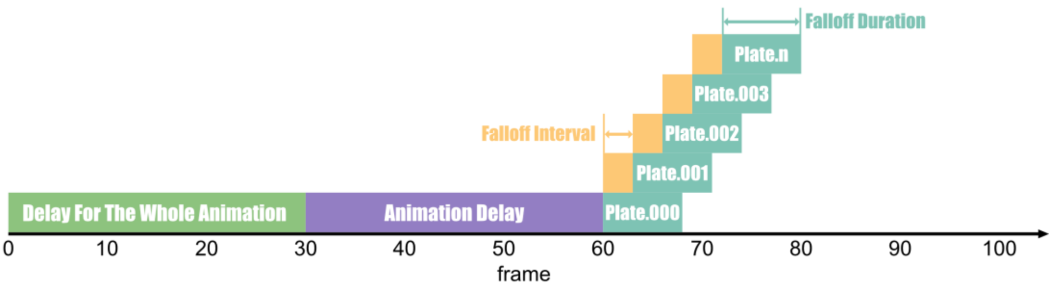
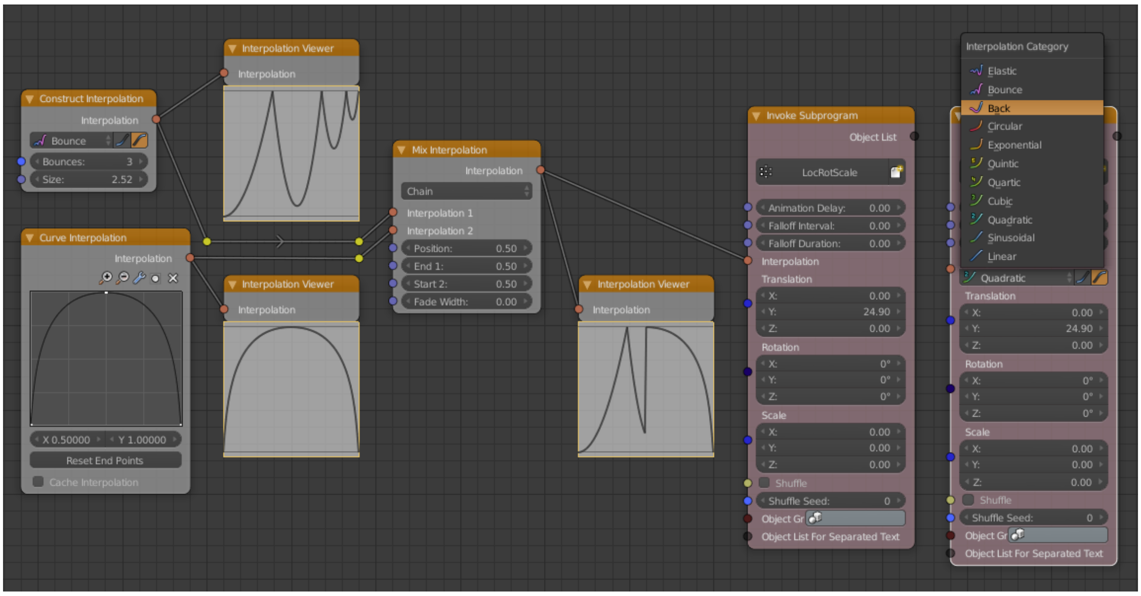


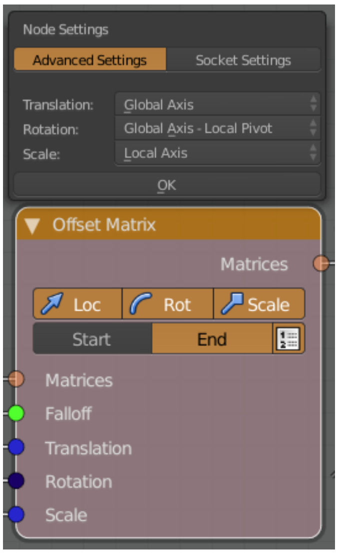
FIGURE 6: The “LocRotScale” Invoke Subprogram node’s graphic (Tiss, 2017)

The Interpolation socket defines the animation curve in time for animated values – Translation, Rotation and Scale. In Animation Nodes interpolation curves can be constructed in many ways which introduce flexibility in animation. Along custom curve creation using the "Curve Interpolation" node there are also pre-defined values found in the "Construct Interpolation" node which can be tweaked easily. Even though mostly interpolation method is picked from the Invoke Subprogram node, these features are very handy in other scenarios. In picture 21 are shown various ways how interpolation curves can be constructed in Animation Nodes.



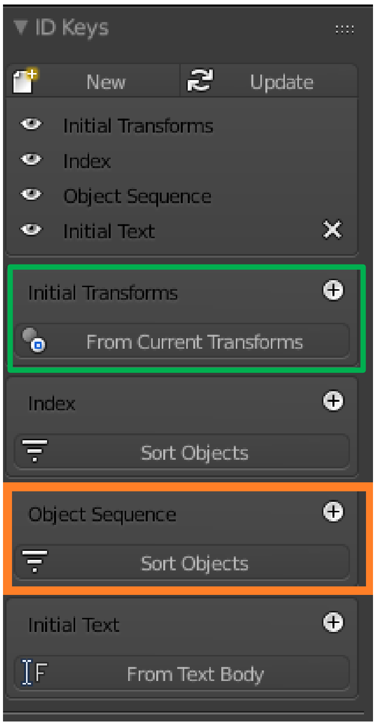
PICTURE 21: Interpolation curve creation in Animation Nodes (Tiss, 2017)

To animate Translation, Rotation and Scale values, Offset Matrices node in the "LocRotScale" node group is used, shown in picture 22. This node uses a list of Matrices to define each object's Start or End state in the scene. Each object has corresponding matrix which contains the object's Initial Transforms – Location, Rotation and Scale. These values must be initialized before passing them to offset matrix node. It can be done in 3D View AN tool shelf, highlighted with green in picture 23. These matrices are initialized for the whole object group at the same time. This operation can be compared to LocRotScale keyframe insertion for each object except values are not tied to a specific frame. The Offset Matrices node also has an advanced settings menu which allows to use various Transform methods. These extra settings ease the control of Transforms and fastens animation process.



PICTURE 22: Offset Matrices node with Advanced Nodes Settings above (Tiss, 2017)

The next feature of this node group is the Shuffle setting. By checking this checkbox, it is possible to determine whether an object group is animated in a random or defined sequence order. Defining object order is a fairly simple process – user must select objects in order in which they should be animated and create a new Object Sequence, highlighted with orange square in picture 23, in which ID Keys tool shelf is shown.



PICTURE 23: ID Keys tool shelf (Tiss, 2017)

Along the Shuffle feature the next implemented Shuffle Seed feature is related. By changing the seed value, the object animation sequence changes randomly and most importantly – it happens immediately to the whole chosen object group. This feature excludes manual animation strip dragging to get randomness effect. figure 7 shows ideological "Shuffle Seed" feature while using "Sidewalk" object group.

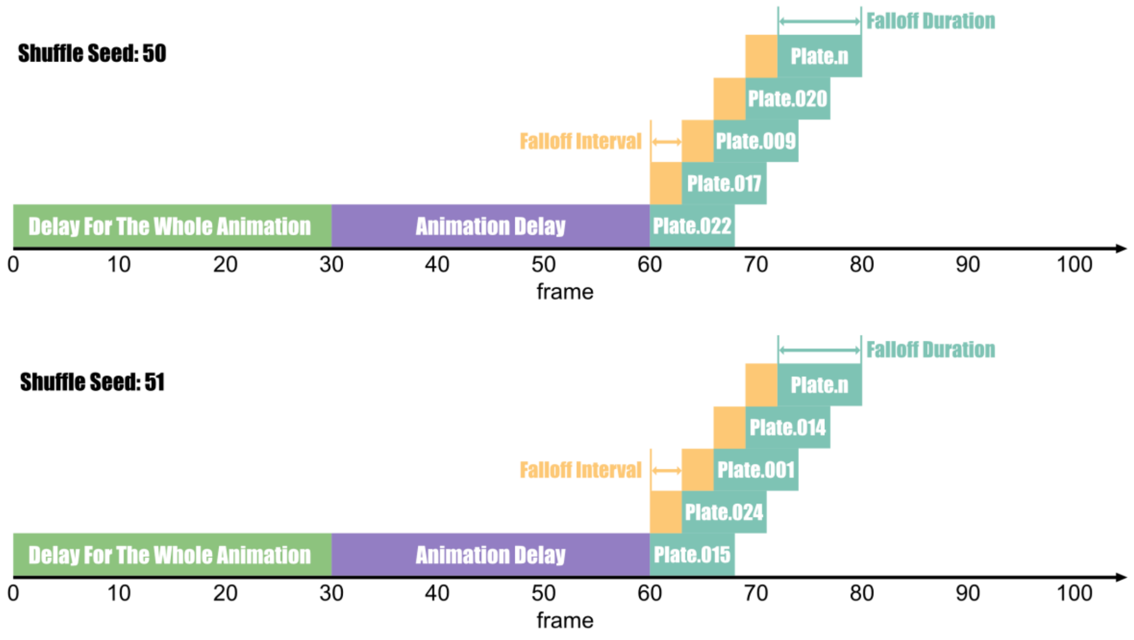
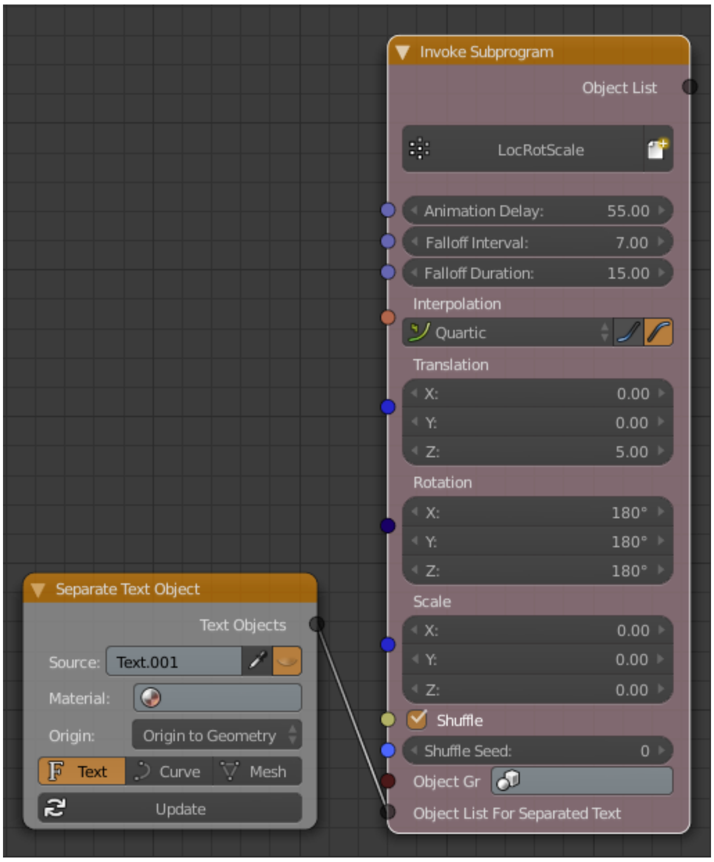


FIGURE 7: Ideological Shuffle Seed feature graph (Tiss, 2017)

The Object group socket determines which object group is going to be animated with applied settings above. Group can be chosen from the drop-down menu which contains all object groups in current scene.

"Object List for Separated Text" is meant to work in combination with another node – The Separate Text Object node. In picture 24 is shown the combination of two. Separate Text Object separates each letter from text object and makes them separate objects. The benefit of this node is that the text can be modified and after pressing Update button the object list updates immediately. This feature excludes manual work for letter separation and pivot point repositioning. In this case study, this feature is used for text animations, described in section 5.8.1.

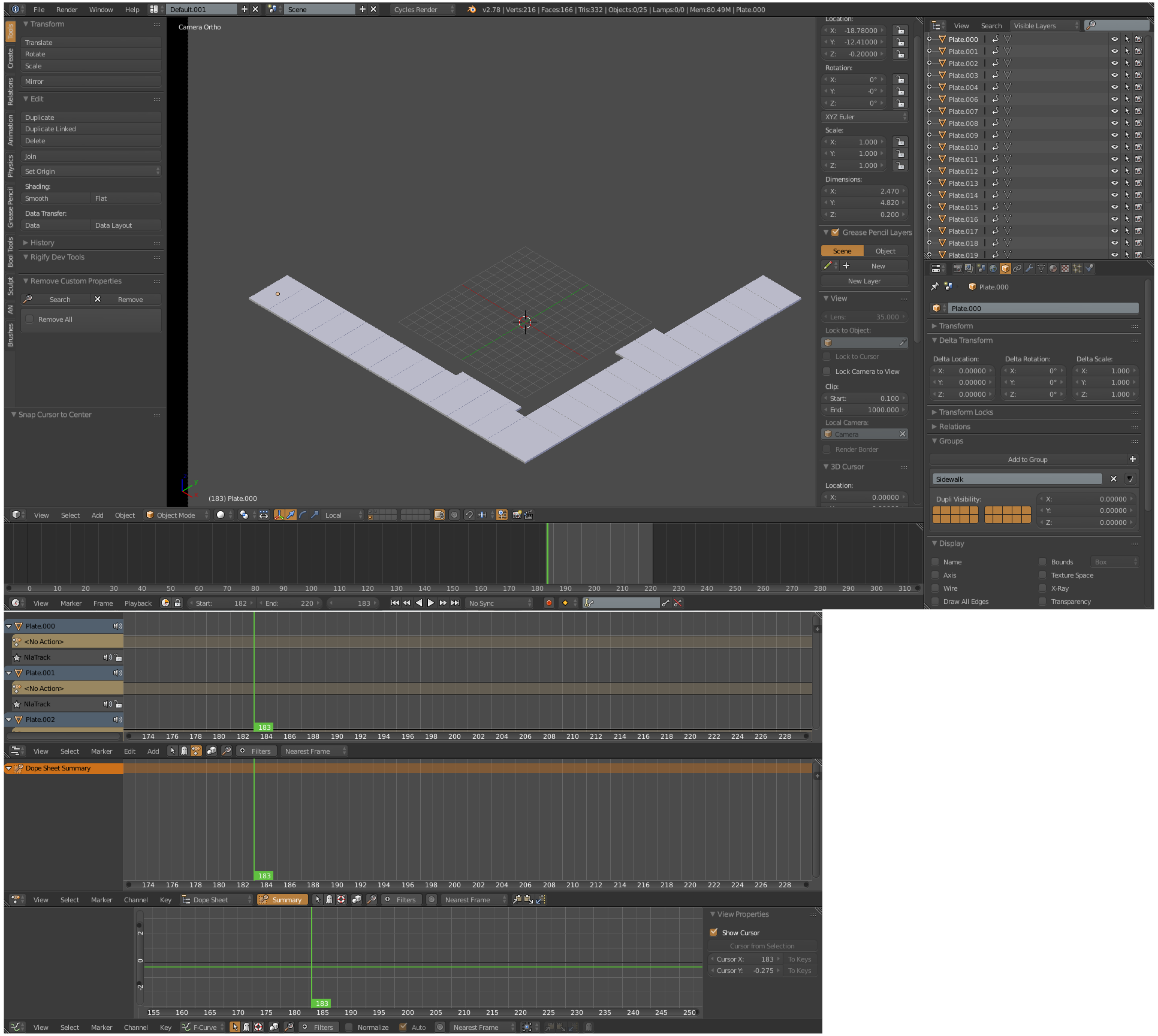


PICTURE 24: Separate Text Object combination with "LocRotScale" Invoke Subprogram node (Tiss, 2017)

5.1.2 Location, Rotation and Scale animation using keyframe animation method only

To compare a node tree setup made with Animation Nodes and the keyframe animation method “Sidewalk”, the same object group, is animated, this time using keyframe animation method only. These two methods have different workflows but share the same values that must be animated. In this chapter, it will be explained how previously manipulated values can be animated without using Animation Nodes. To compare these two methods as equally as possible the same object group’s animation is repeated with the keyframe animation method.

To work with the keyframe animation method fluently it is more comfortable to have at least two displays. It is because many different working windows must be accessed frequently. An example of workspace configuration is shown in picture 25.



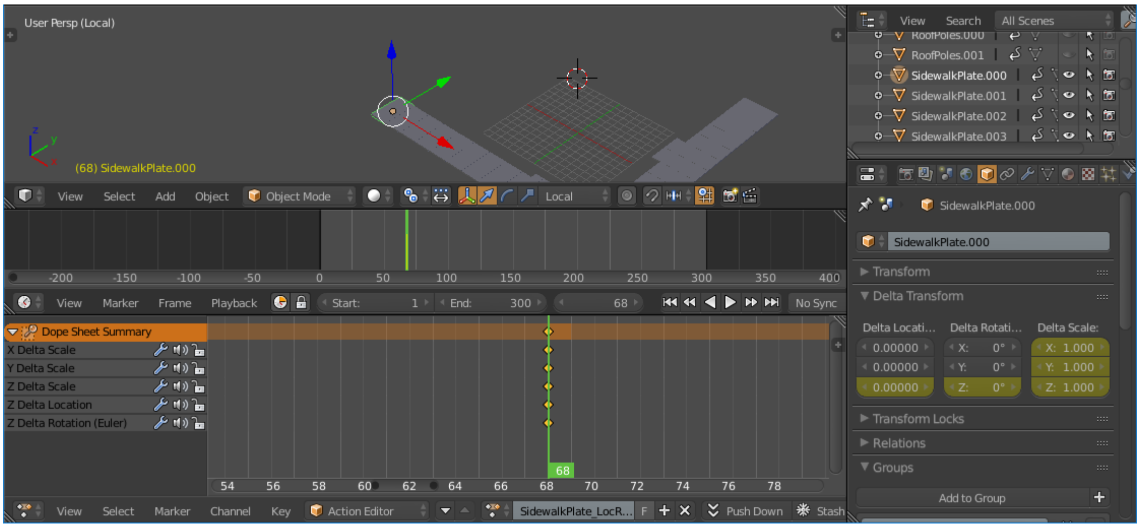
PICTURE 25: Comfortable workspace setup to work with keyframe animation method in Blender (Tiss, 2017)

The keyframe animation method is straightforward, therefore user does not have to re-consider all necessary settings beforehand and values can be animated on the go. Not like when using Animation Nodes, user does not have to predefine necessary values. Excluding this stage, it is possible to save time and animate single objects faster.

Instead of initializing the end Transforms of the object, using the keyframe animation method, the animator must set keyframes in the end state of each animated value. To replicate previously made animation, following settings must be taken account:

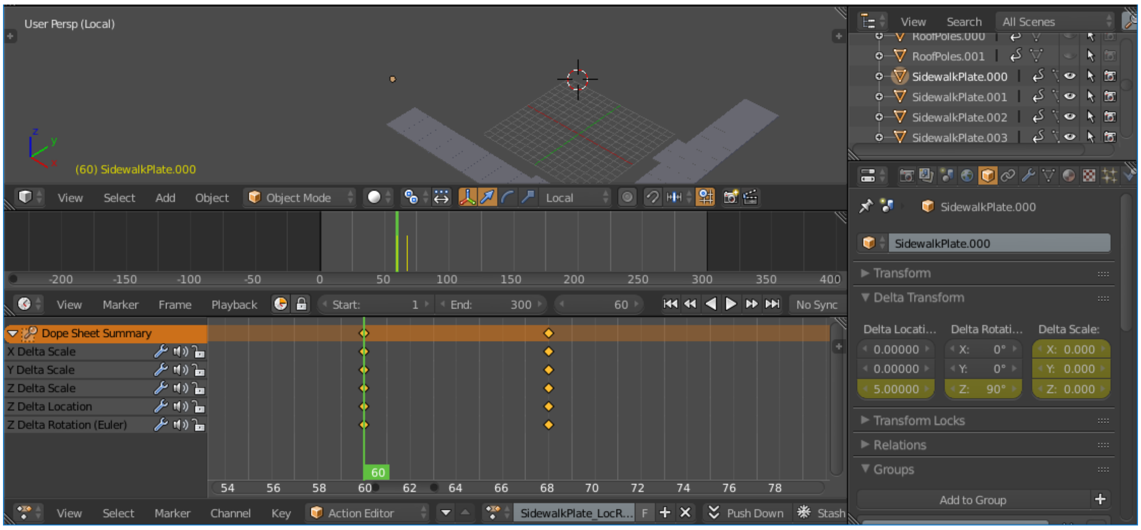
- The start of animation of the first object is frame 60 (Animation Delay (30 frames) + Delay for The Whole Animation (30 frames))
- Delay of the next object's animation is 3 frames from previous
- Duration of animation is 8 frames
- Interpolation method: Back, Ease Out
- Translation along Z axis by 5 Blender units
- Rotation along Z axis by 90°
- Start scale of the object is 0
- "SidewalkPlate.n" objects are animated in following sequence

According, to predefined values the end of the first object's animation is frame 68. That is the frame at which end Transforms must be set. To keep the animation project organized it is recommended to create a new action in Dope Sheet. In picture 26 is represented a set keyframes in a new action strip of animated values.



PICTURE 26: Set keyframes of Location, Rotation and Scale end values for SidewalkPlate.000 (Tiss, 2017)

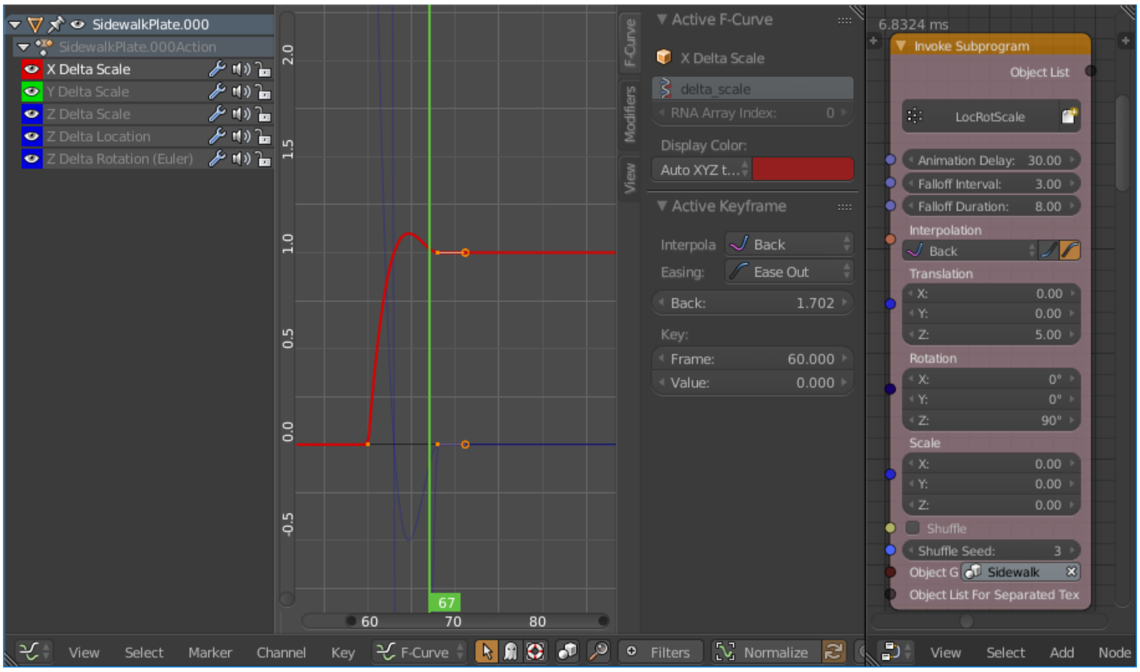
The Animating Delta Transform values is more equal to AN node tree execution, in which delta values are submitted only. Using Delta Transform allows user to set rounded values which fastens the animation development process. To finish the SidewalkPlate.000 animation, the beginning keyframe of the animation must be set on frame 60. In picture 27 are shown the Delta Transform values at frame 60.



PICTURE 27: Set keyframes of Location, Rotation and Scale start values for SidewalkPlate.000 (Tiss, 2017)

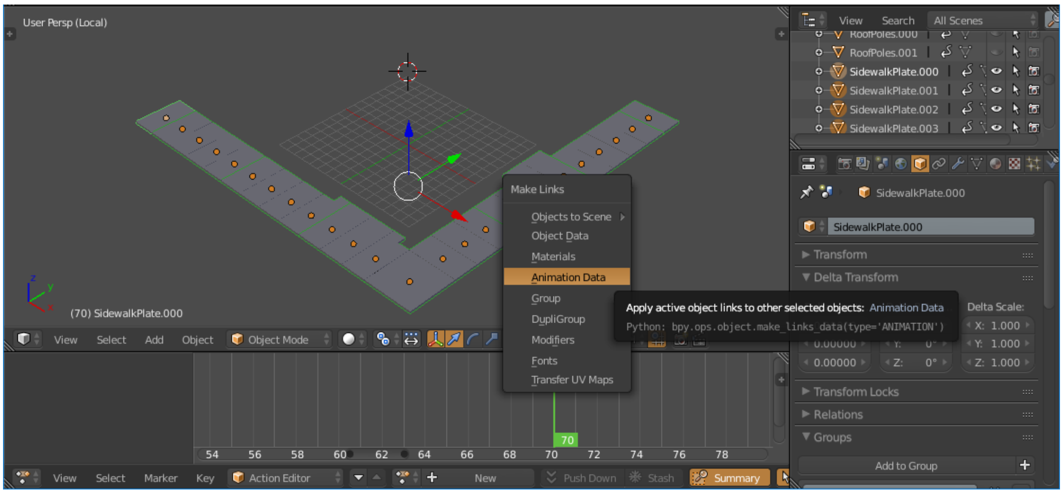
These steps are equivalent to Animation Delay, Falloff Duration, Translation, Rotation and Scale functions in the “LocRotScale” node group.

The next step is to set interpolation. There are four interpolation curves that must be changed to “Back” with Ease Out. When selecting the Interpolation mode in the node group, it affects all values at the same time. When using the keyframe animation method each interpolation curve must be adjusted individually. This preference gives extra flexibility to animation but in such short animation there is no need for such adjustments. Therefore, in this case it just ads extra work when using the keyframe animation method. In picture 28 the visual presentation of setting the interpolation method in Graph Editor on the left and Node Editor on the right are being compared.



PICTURE 28: Interpolation settings in Graph Editor on the left and “LocRotScale” Invoke subprogram node on the right.

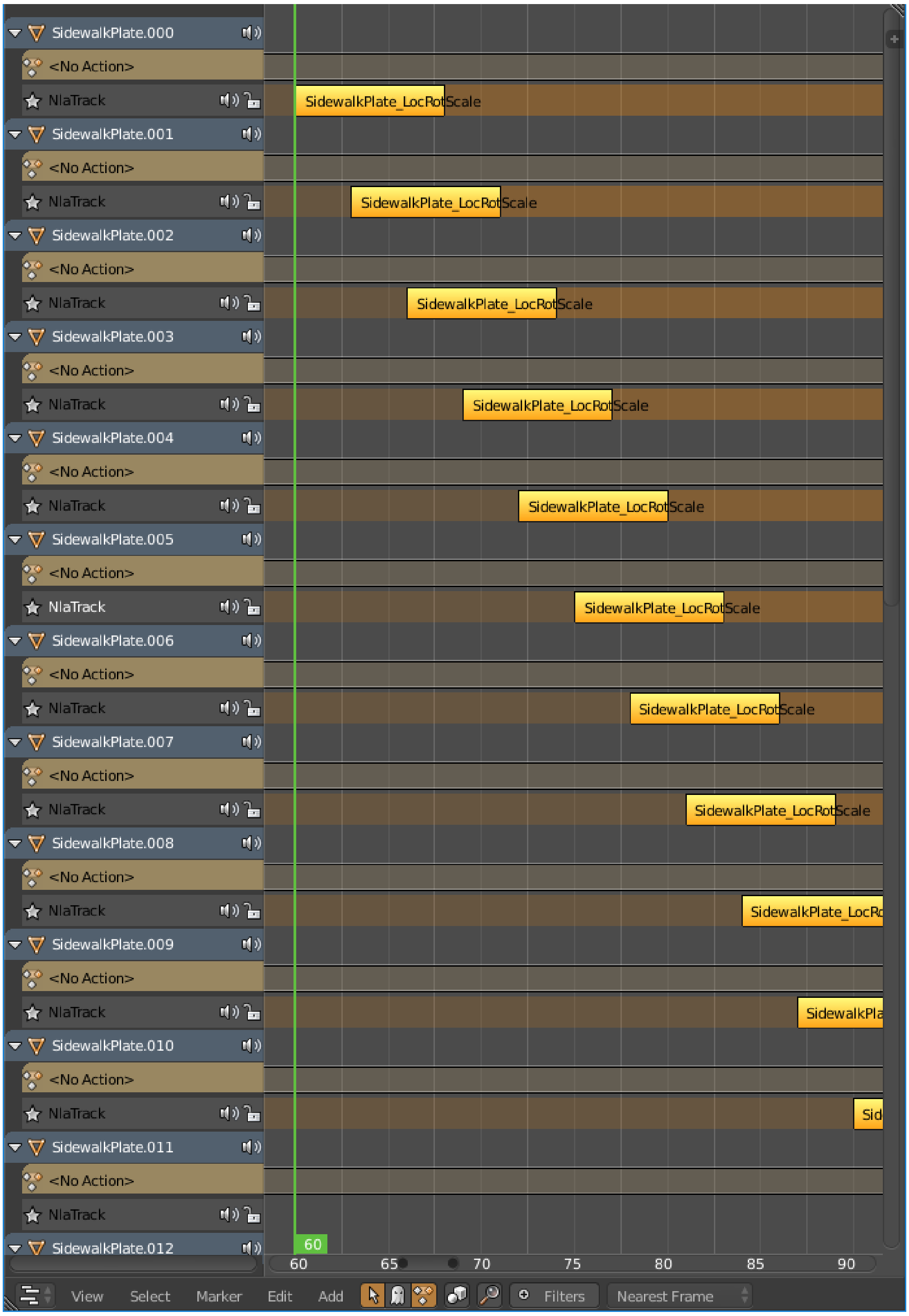
To reuse the same animation for the rest of the objects in “Sidewalk” object group and implement Falloff Interval and the Use Sequence settings, user must Push Down the SidewalkPlate_LocRotScale Action to create an action strip in NLA Editor. After that all objects in the object group must be selected. Note that the already animated object must be the active object. Then using Make Links (shortcut key in 3D View: Ctrl+L), Animation Data is selected, shown in picture 29. These steps copy the already made Animation Strip from the active object to other objects. Now all objects in the Sidewalk object group share the same animation strip, shown in picture 30. Naming objects incrementally helps to keep animation organized and eases to create proper object animation sequence.



PICTURE 29: Copying Animation Data from active object to selected objects (Tiss, 2017)

PICTURE 30: “SidewalkPlate.n” objects’ animation strips in NLA Editor (Tiss, 2017)

To achieve the Falloff Interval setting with the keyframe animation method, user manually must drag animation strips in the NLA Editor. In picture 31 is shown finished action strip sequence for the object animations of the “Sidewalk” object group.

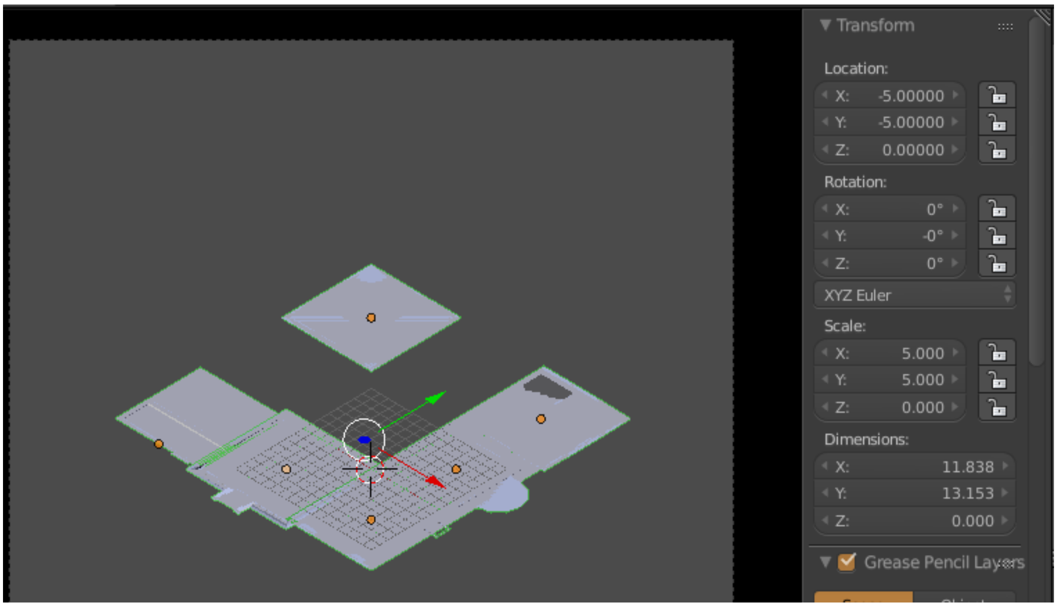


PICTURE 31: Finished action strip sequence for object animations of “Sidewalk” object group (Tiss, 2017)

5.2 Object hiding and unhiding

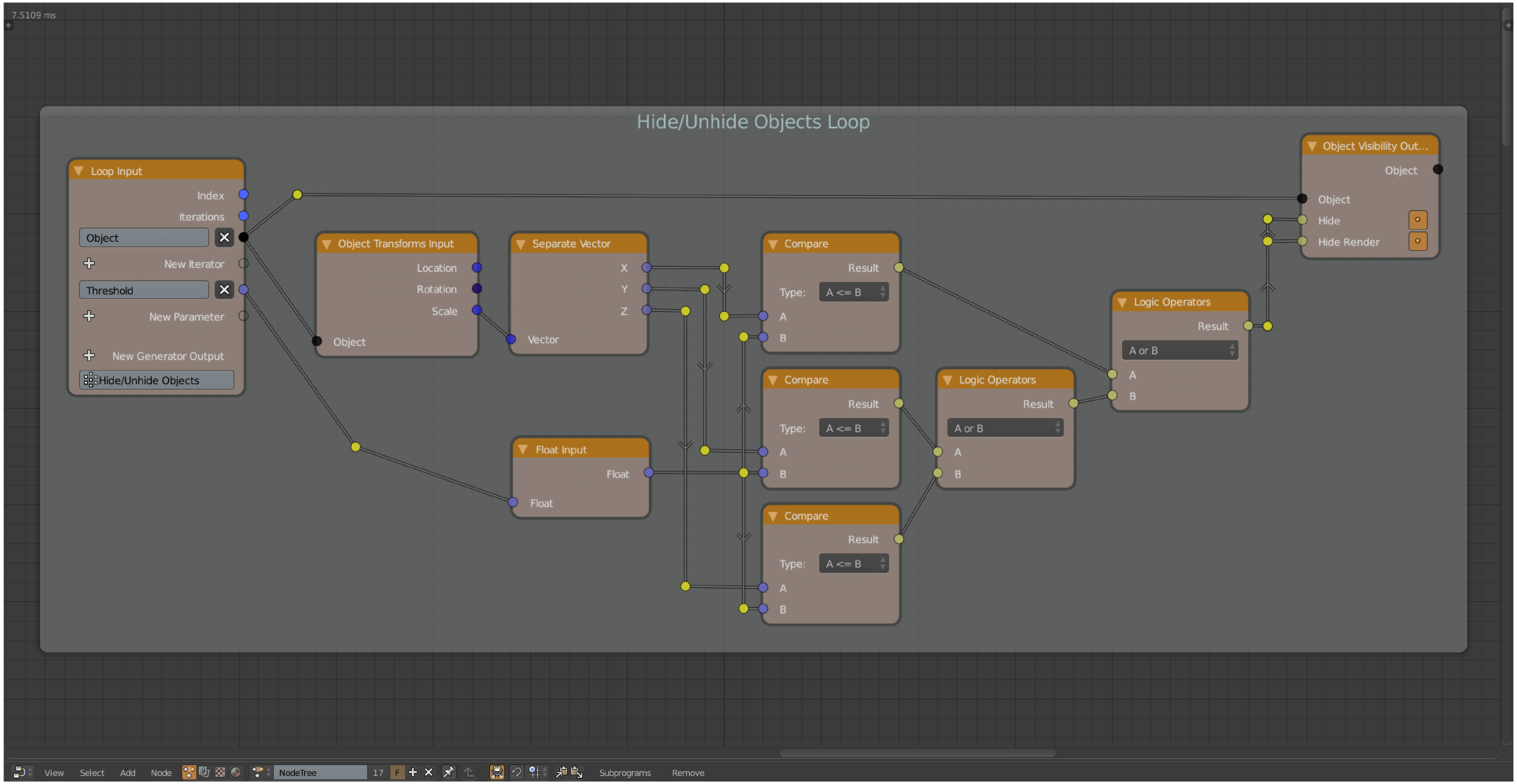
5.2.1 Object hiding and unhiding using Animation Nodes

In this animation, there are objects that are not scaled along all axes, therefore one or more scale values are kept constant. Because at least two scale values are not changed, there appears unwanted scale effect of object in the animation scene, shown in picture 32. To avoid this artefact, the object must be hidden from 3D Viewport and Render until the beginning of certain object's animation.



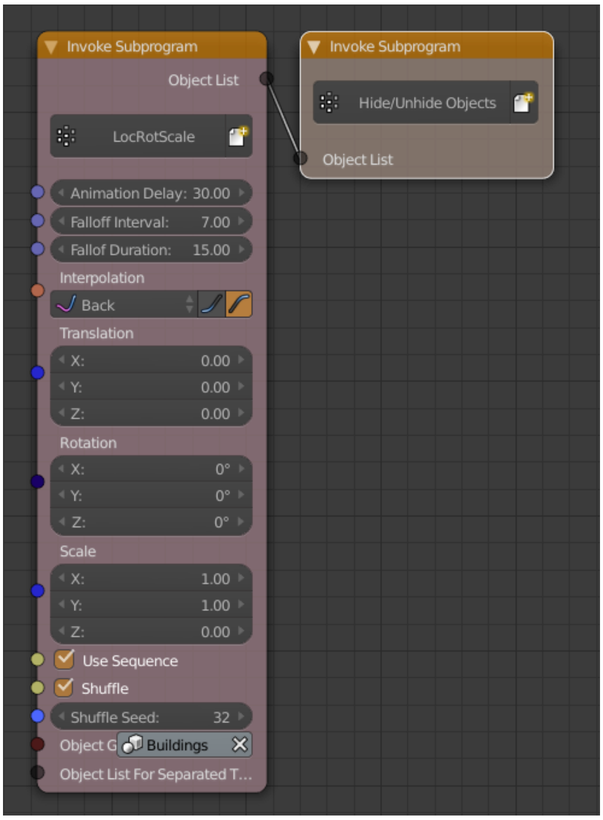
PICTURE 32: "Buildings" object group in 3D Viewport with Z scale 0.0 (Tiss, 2017)

To hide and unhide objects from the scene by using Animation Nodes, it is effective to set threshold value, which controls either object is hidden or not. In this case study, a reusable loop is created, which hides objects, which at least have one scale value equal or less than 0.01, which is nearly invisible. "Hide/Unhide Objects" loop, is shown in picture 33.



PICTURE 33: “Hide/Unhide Objects” Loop’s node tree (Tiss, 2017)

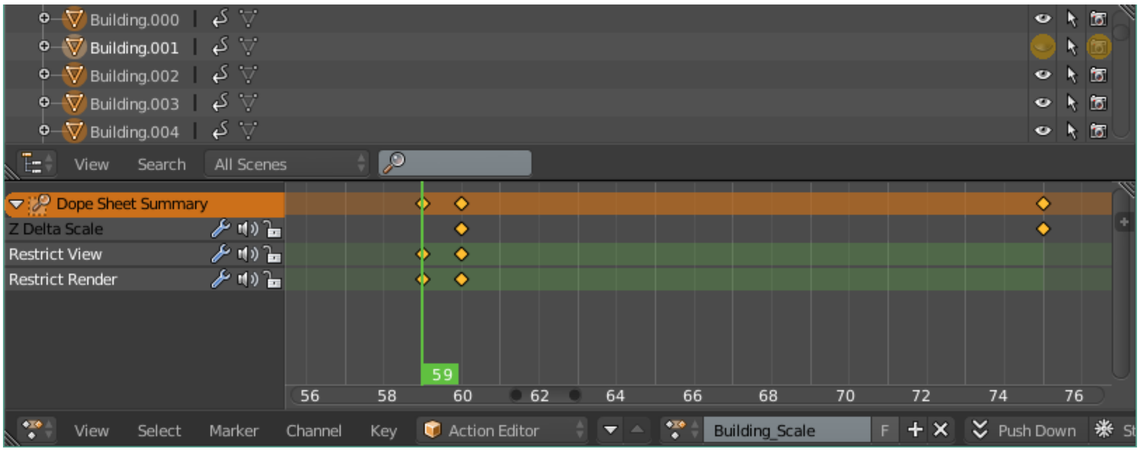
The “LocRotScale” node group outputs object list, which contains each object’s Location, Rotation and Scale values. In this case, only Scale values are necessary. “Hide/Unhide Objects” loop iterates trough returned Object List from “LocRotScale” Invoke Subprogram node and compares each object’s scale to threshold value 0.01 which is defined by the user. If any of the object’s Scale value is equal or less than threshold value, then it is hidden from Viewport and Render. picture 34 represents the combination of “LocRotScale” and “Hide/Unhide Objects” Invoke Subprogram nodes.



PICTURE 34: Combination of the “LocRotScale” and "Hide/Unhide Objects" Invoke Subprogram nodes (Tiss, 2017)

5.2.2 Object hiding and unhiding using keyframe animation method

When using the keyframe animation method different approach is used. Instead of hiding or unhiding object regarding to its scale, each object is hidden straightforward until the beginning of its animation. User must animate object’s hide from the viewport and hide from the render settings. This is done by inserting hide/unhide keyframes before object’s animation when creating a new action. In picture 35 is shown hide/unhide keyframes in the “Building_Scale” action.

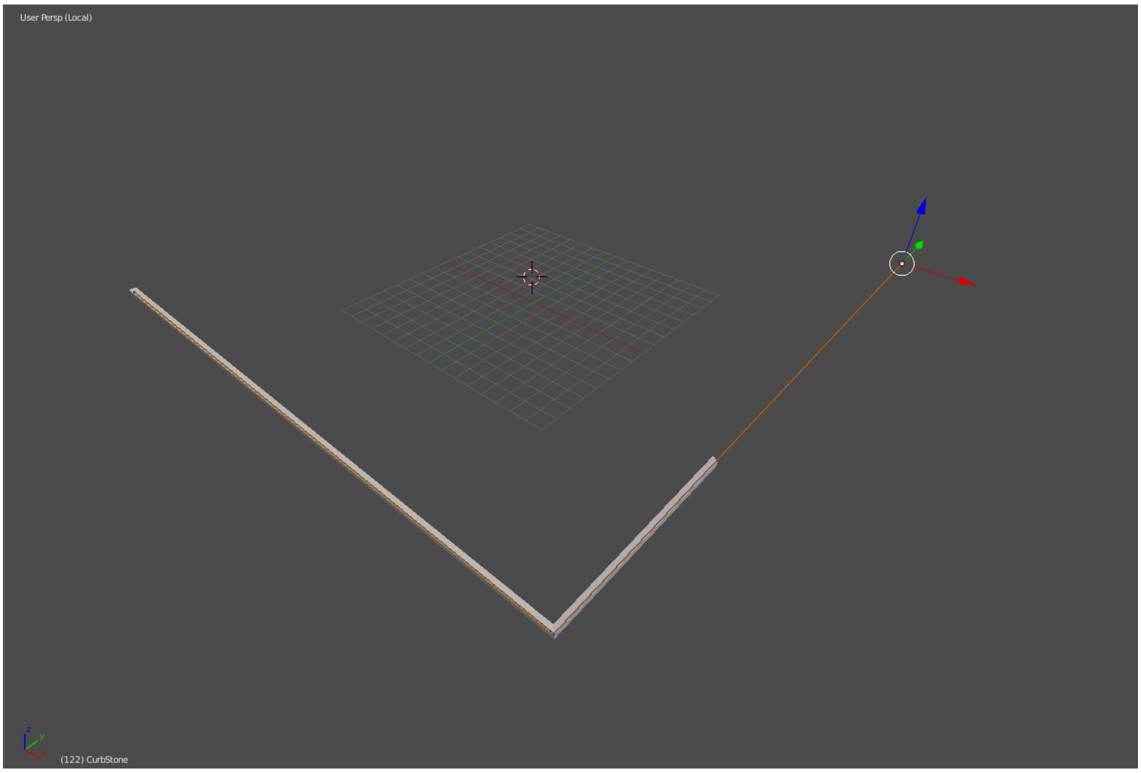


PICTURE 35: Object hide/unhide from 3D Viewport and Render using keyframe animation method (Tiss, 2017)

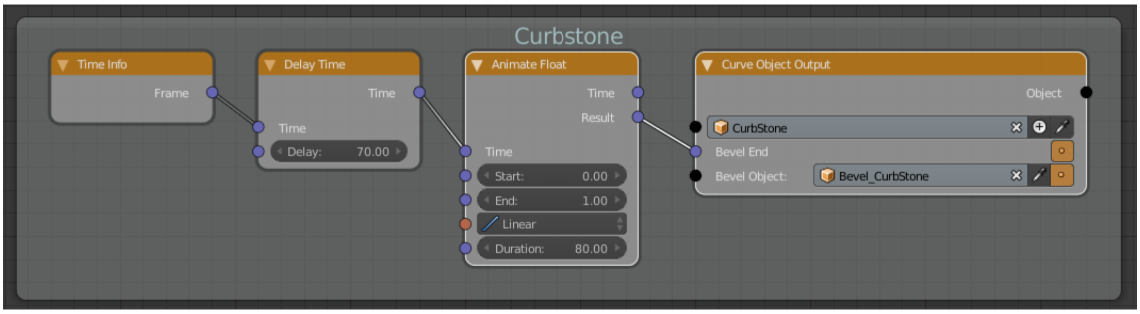
5.3 "CurbStone" animation

5.3.1 "CurbStone" animation using Animation Nodes

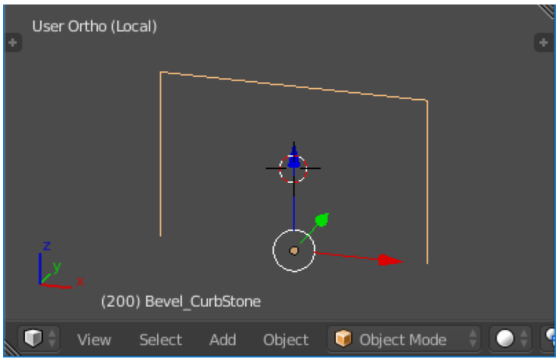
"CurbStone", shown in picture 36, is a curve object, therefore it desires different approach of animation than mesh objects. To animate "CurbStone" with Animation Nodes a separate program is needed. Since there is only one "CurbStone" object it is enough to execute the program only once during the "data-visualize-edit" cycle. "CurbStone" animation's node tree is shown in picture 37. Note that Delay for The Whole Animation is not applied to this node tree, therefore Delay Time value is set separately. Bevel End value is a float number therefore the Animate Float node is used from which float value from 0.0 to 1.0 in time of 80 frames is passed linearly to Curve Object Output node. In Curve Object Output node "CurbStone" object is selected as object to be affected. In this case study Bevel_Curbstone is selected as Bevel Object which defines the shape of Bevel. Bevel_Curbstone is a spline object, shown in picture 38.



PICTURE 36. “CurbStone” object (Tiss, 2017)



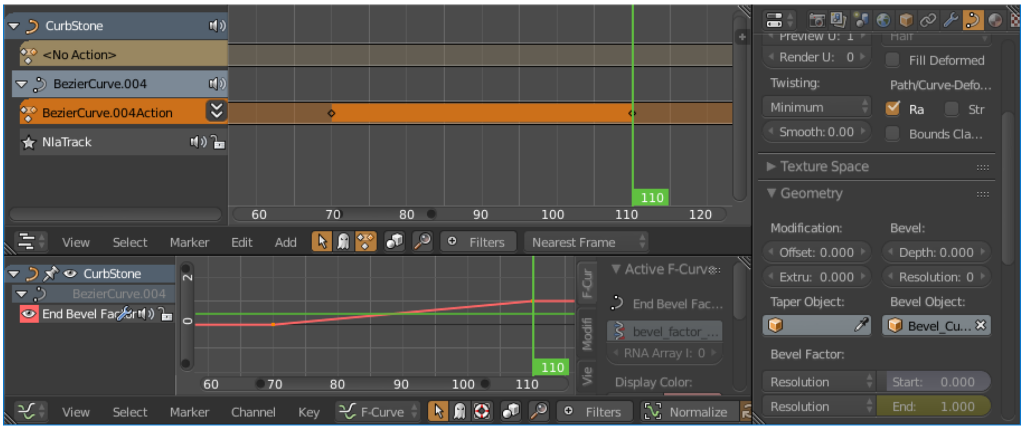
PICTURE 37: “CurbStone” animation node tree (Tiss, 2017)



PICTURE 38: Bevel_Curbstone spline object (Tiss, 2017)

5.3.2 “CurbStone” animation using keyframe animation method

To replicate the same effect with keyframe animation method, Bevel value must be animated in the Object Data panel. In picture 39 is shown “CurbStone” animation settings using the keyframe animation method. Interpolation must be changed to linear in the Graph Editor. The Bevel_Curbstone must be selected as the Bevel Object.

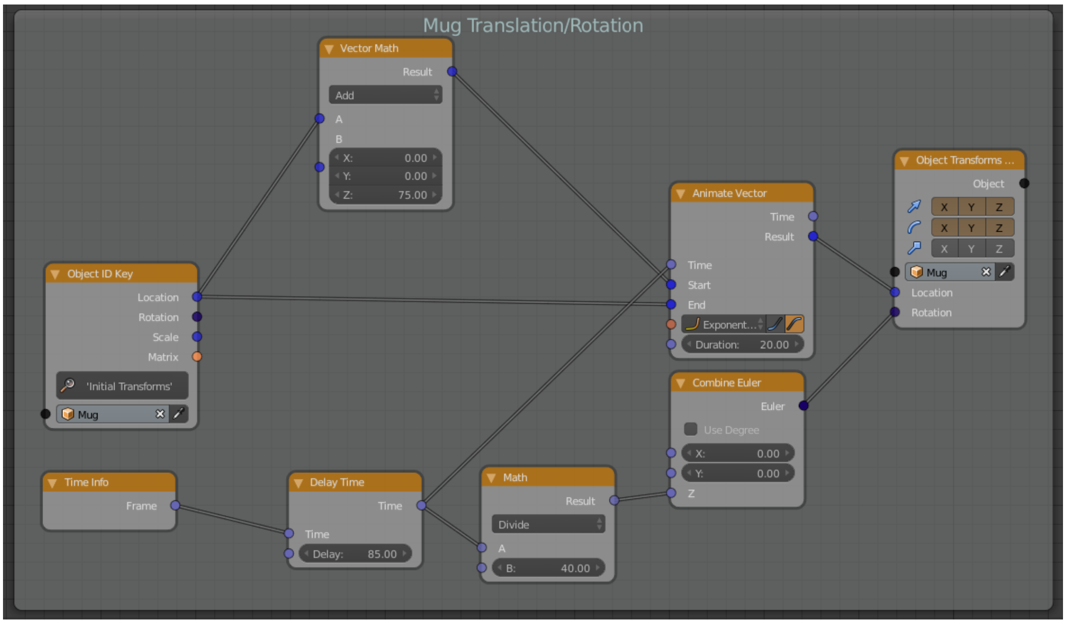


PICTURE 39: “CurbStone” animation using keyframe animation method (Tiss, 2017)

5.4 “Mug’s” Location and Rotation animation

5.4.1 “Mug’s” Location and Rotation animation using Animation Nodes

“Mug’s” animation differs from other animations with its infinite rotation during the animation. Since “LocRotScale” node group is finite, endless rotation is not achievable by using this node group. Even though there are workarounds to make this animation happen by reusing the same node group, more controllable setup is achievable by creating a separate node tree. In picture 40 is shown “Mug’s” animation using Animation Nodes.

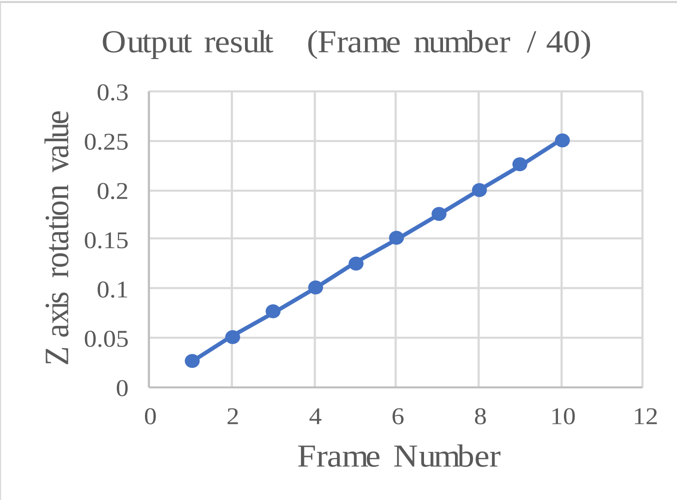


PICTURE 40: Node tree for "Mug's" Location and Rotation animation (Tiss, 2017)

To animate "Mug's" location, Animate Vector node is used. It receives two vector values to define the start and the end location and passes the result to the Object Transforms Output node. To make infinite rotation the Combine Euler node is used so rotation around X and Y axis can stay constant. Only rotation around Z axis is necessary. To define Z rotation value each frame number is divided by predefined step size – 40 in the Float Math node. This approach ensures linear rotation increment. Math node's output results from frame 1 to 10 are shown in table 2.

TABLE 2. The Float Math node's output results according to frame number in range from 1 to 10 (Tiss, 2017)

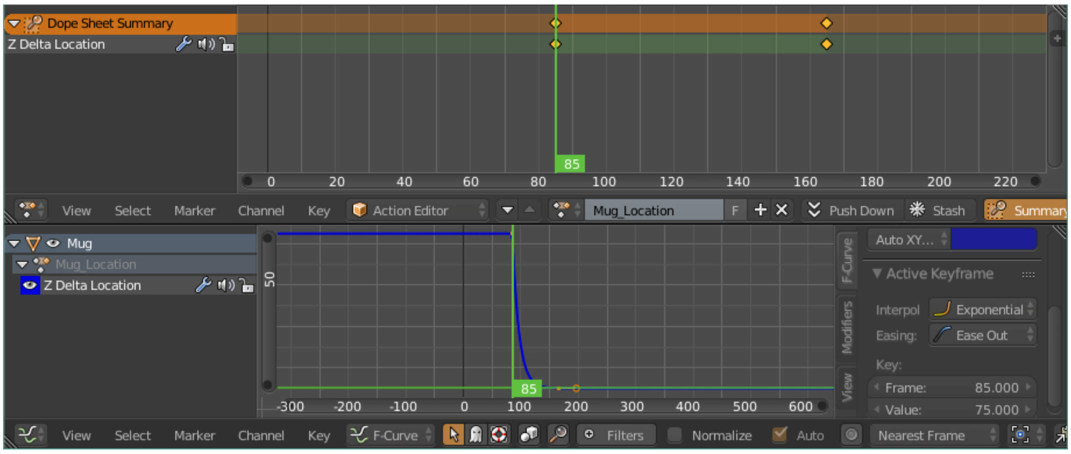
Frame number	Output result (Frame number / 40)
1	0.025
2	0.05
3	0.075
4	0.1
5	0.125
6	0.15
7	0.175
8	0.2
9	0.225
10	0.25
n	n/40



5.4.2 “Mug’s” Location and Rotation animation using keyframe animation method

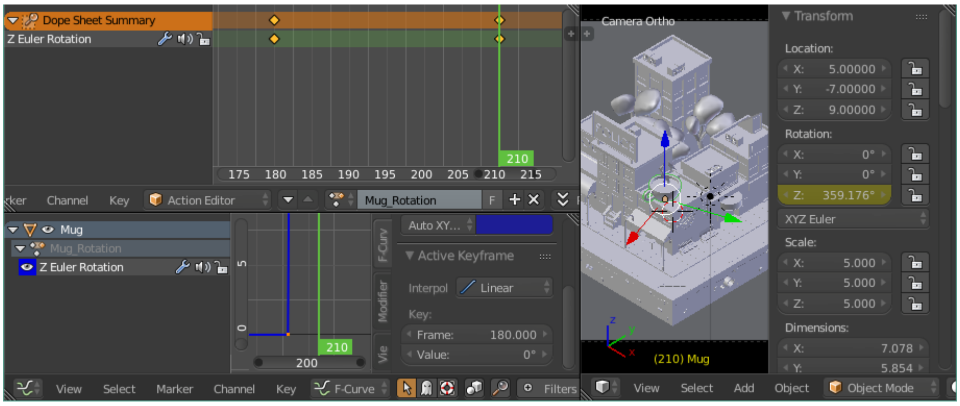
To achieve a similar result by using the keyframe animation method there must be two separate actions: one which is responsible for translation and another which is responsible for rotation only.

In picture 41 is shown “Mug’s” translation animation setup using the keyframe animation method. Z Delta Location value is 75.0 at frame 85 and 0 at frame 165.

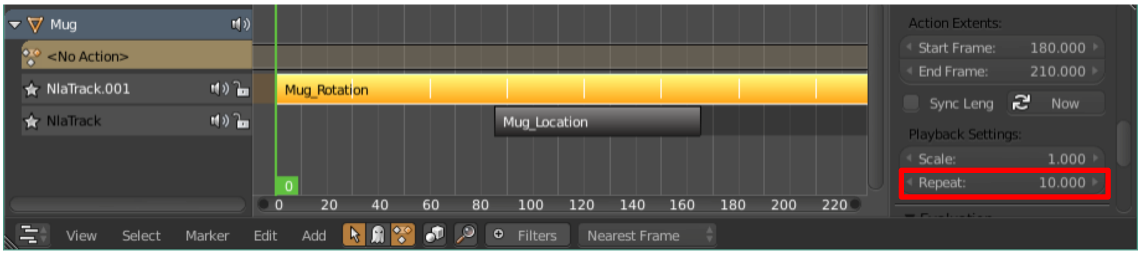


PICTURE 41: "Mug's" translation animation setup using the keyframe animation method (Tiss, 2017)

To create an infinite rotation loop it is mandatory to create a seamless repeatable action strip with “Mug’s” rotation. In picture 42 is shown “Mug’s” rotation, in 30 frames, it turns by 359.176° along Z axis. It is recommended to make interpolation to linear, to keep rotation even unless other effect is desired. The rotation in the next frame must be 360° or 0° along Z axis, therefore it is possible seamlessly repeat “Mug Rotation” action throughout the whole animation by using Repeat setting, highlighted in picture 43. This could also be done with Cycles modifier in the Graph Editor.



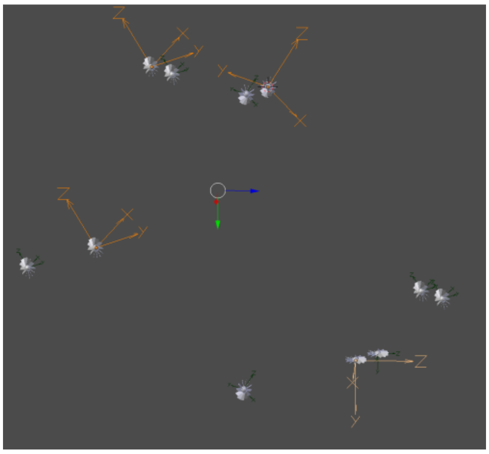
PICTURE 42: “Mug’s” rotation value along local Z axis in 30 frames (Tiss, 2017)



PICTURE 43: Repeat setting in NLA Editor (Tiss, 2017)

5.5 “VentBlade.n” rotation

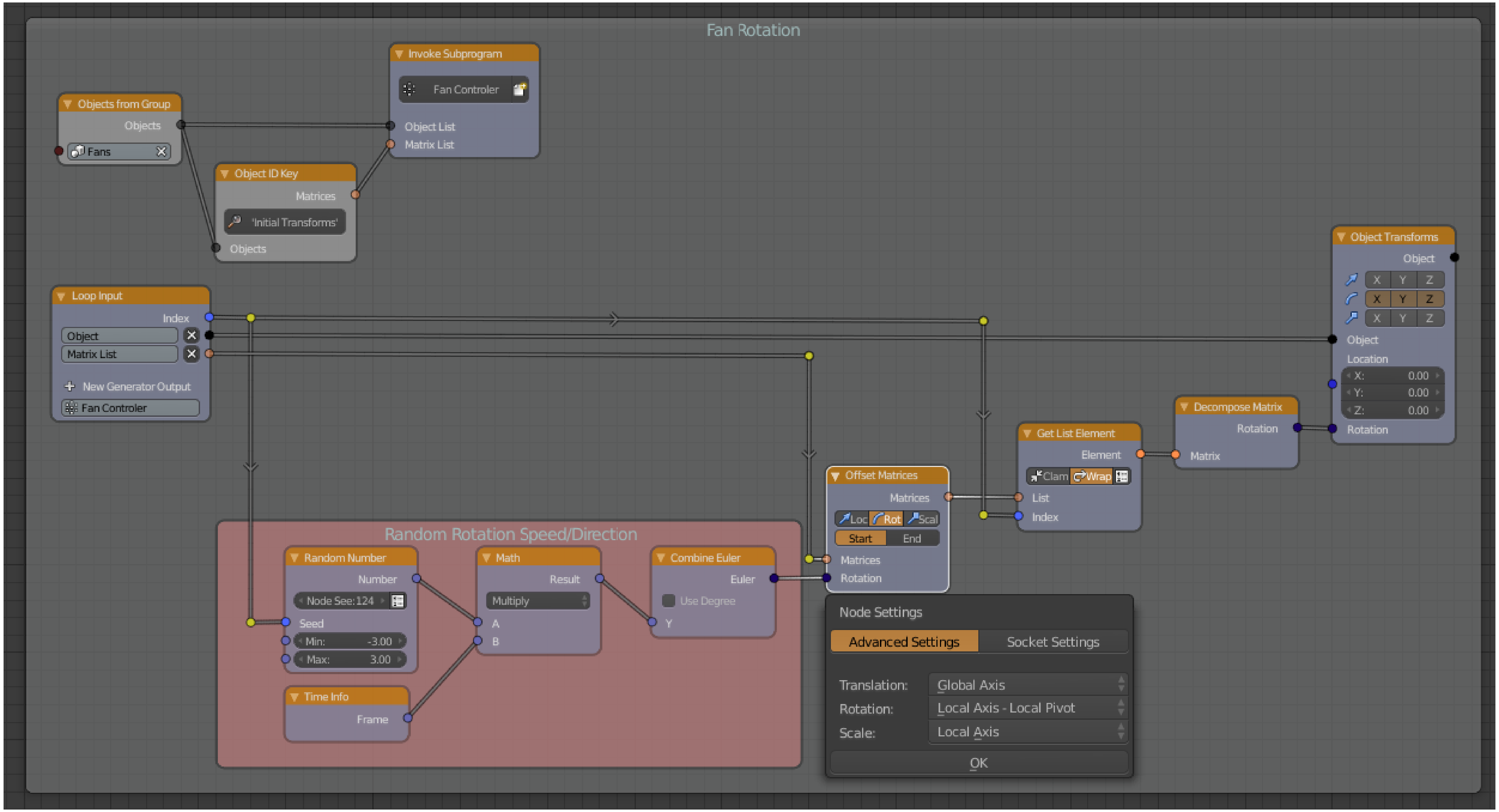
Approach to rotate “VentBlades” is very like “Mug’s” rotation animation – the animation must be infinite. Since each “VentBlade.n” is parented to a corresponding “VentBox.n” object, there is no need to take care of each object’s Translation. The main difference between “Mug” and “VentBlades” is that there are more than one object and they are rotated differently in the 3D space. In picture 44 are shown “VentBlades” local axis.



PICTURE 44: "VentBlades" local axes in 3D space (Tiss, 2017)

5.5.1 “VentBlade.n” rotation using Animation Nodes

Although the “VentBlades” are rotated in different directions they all share a common interest – each of them must be rotated around the local Y axis. When working with Animation Nodes, rotating objects around local axes is possible by using an Offset Matrices node. Therefore, Transforms must be initialized before passing the object group into the Object ID Key node. In picture 45 is shown the node tree of the “FanController” loop’s node setup for infinite “VentBlade” rotation.



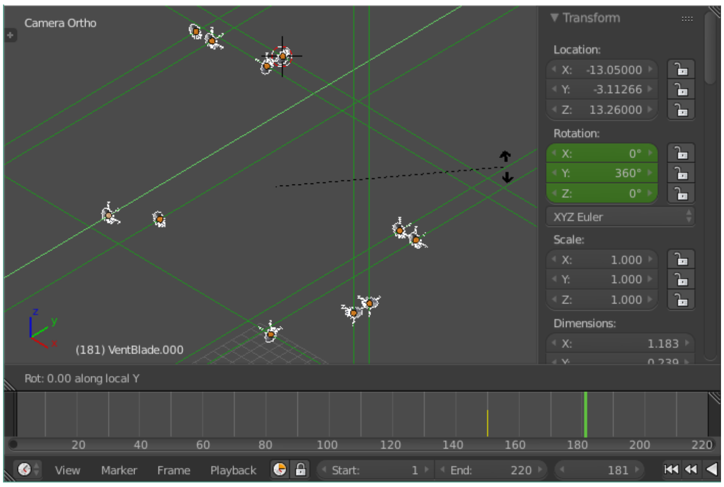
PICTURE 45: Node tree for “VentBlade.n” infinite rotation (Tiss, 2017)

The created node tree collects data from the “VentBlade” object group and passes them to a Fan Controller loop, in which each “VentBlade’s” rotation is treated individually. The loop has two inputs: an Object List and a Matrix List. Selected object from the Object List determines which object is rotated while corresponding matrix provides the information about the object’s transforms including rotation data which are changed over time. Since in the Offset Matrices’ Advanced Node Settings Rotation is set to Local axis, all “VentBlades” turn around their pivot points in local space.

Random Rotation Speed/Direction block allows to set each “VentBlade’s” speed randomly in specified range. Since Random Number value is set from -3.0 to 3.0 it also determines whether “VentBlade.n” will rotate clockwise or counter clockwise. By setting minimum and maximum Random Number values it is possible to determine the range of minimum and maximum speed. The random number becomes a step size of rotation and stays constant over time. This preference assures linearity in VentBlade’s animation. Multiplying step size with the frame number user gets the final rotation value at current frame which is passed to Combine Euler node and sets new start rotation value of each object. Since Start value is changed rotation will last until animation is interrupted.

5.5.2 “VentBlade.n” rotation using keyframe animation method

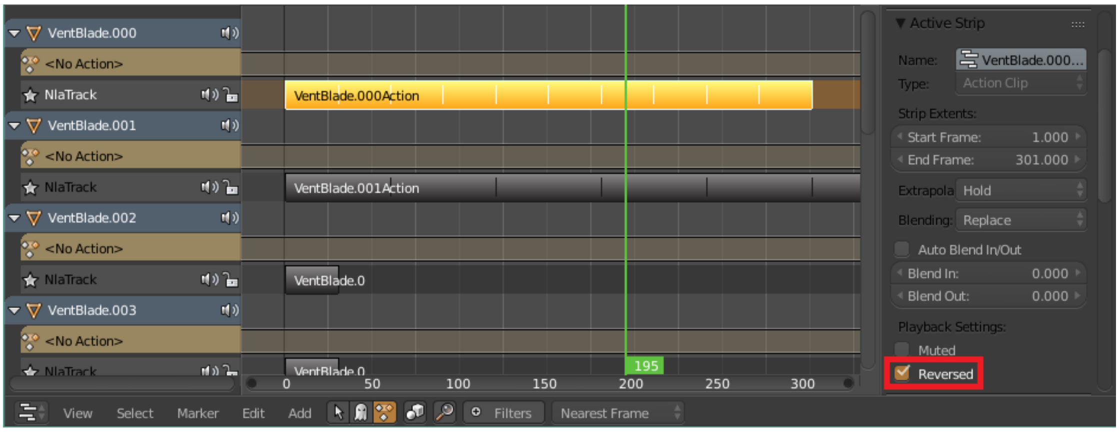
To achieve the effect described in section 5.5.1 with keyframe animation method all “VentBlades” must be animated at the same time, thus it is possible to animate each object’s rotation along its local axis at the same time. Copy animation data does not work in this case because it copies rotation settings along global axis. In picture 46 is shown set keyframes for “VentBlade” 360° rotation along their local Y axis.



PICTURE 46. “VentBlades” being rotated by 360° in duration of 31 frames (Tiss, 2017)

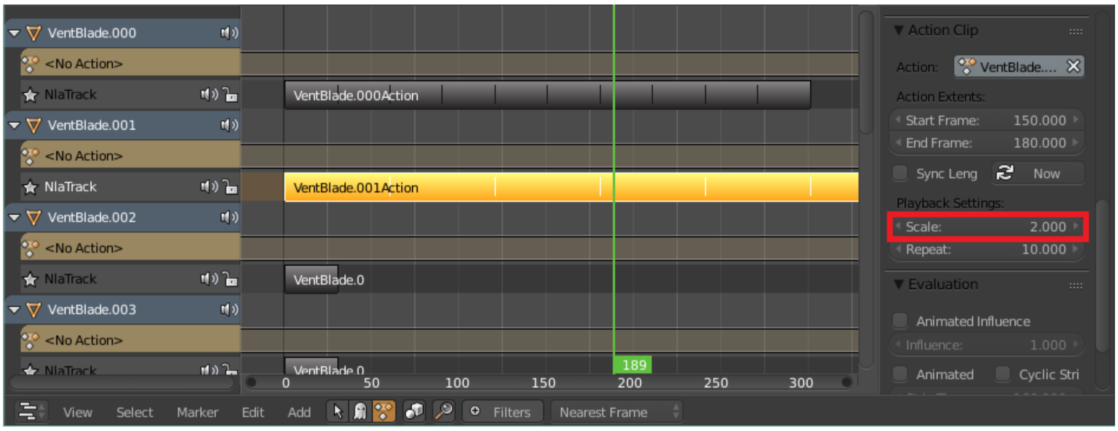
After creating 360° rotation along their local Y axis, a seamless animation loop must be created. To create a seamlessly repeatable action strip, it is possible to use the approach described in section 5.4.2. After setting the last keyframes and changing the interpolation method to linear, each action must be pushed down in the NLA Editor individually and then these strips must be dragged to the beginning of the animation. Repetition could also be achievable by the Cycles modifier in the Graph Editor.

To change the rotation direction of a certain “VentBlade” in the NLA Editor at the Active Strip Settings Reversed option must be checked, highlighted in picture 47.



PICTURE 47: Reversed setting in NLA Editor (Tiss, 2017)

To change rotation speed of certain “VentBlade” in NLA Editor at Action Strip settings Scale must be adjusted, shown in picture 48.

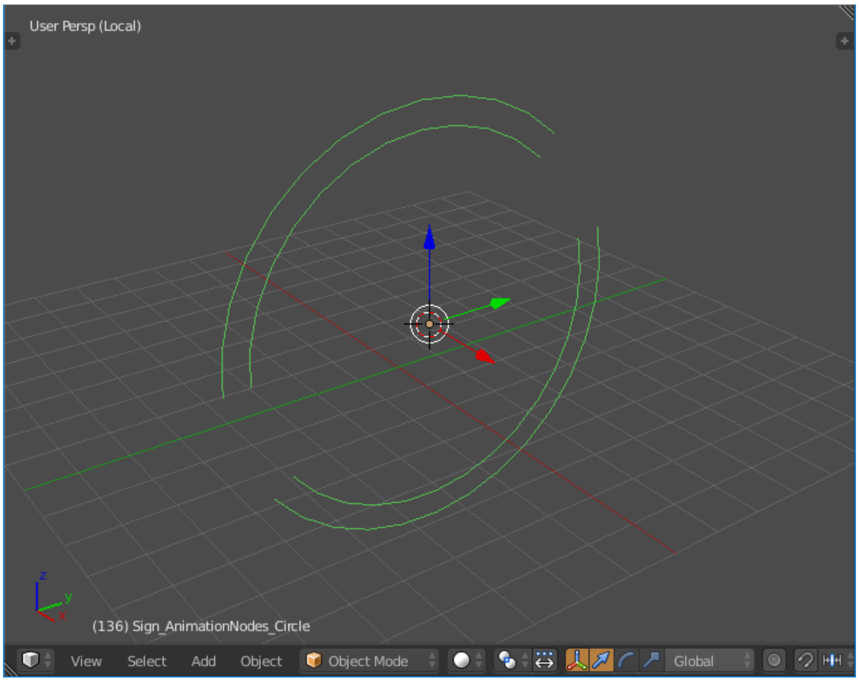


PICTURE 48: Scale setting in the NLA Editor allows the adjusment of the duration of the action strip (Tiss, 2017)

5.6 Spline bevel for the “Sign_AnimationNodes_Circle” animation

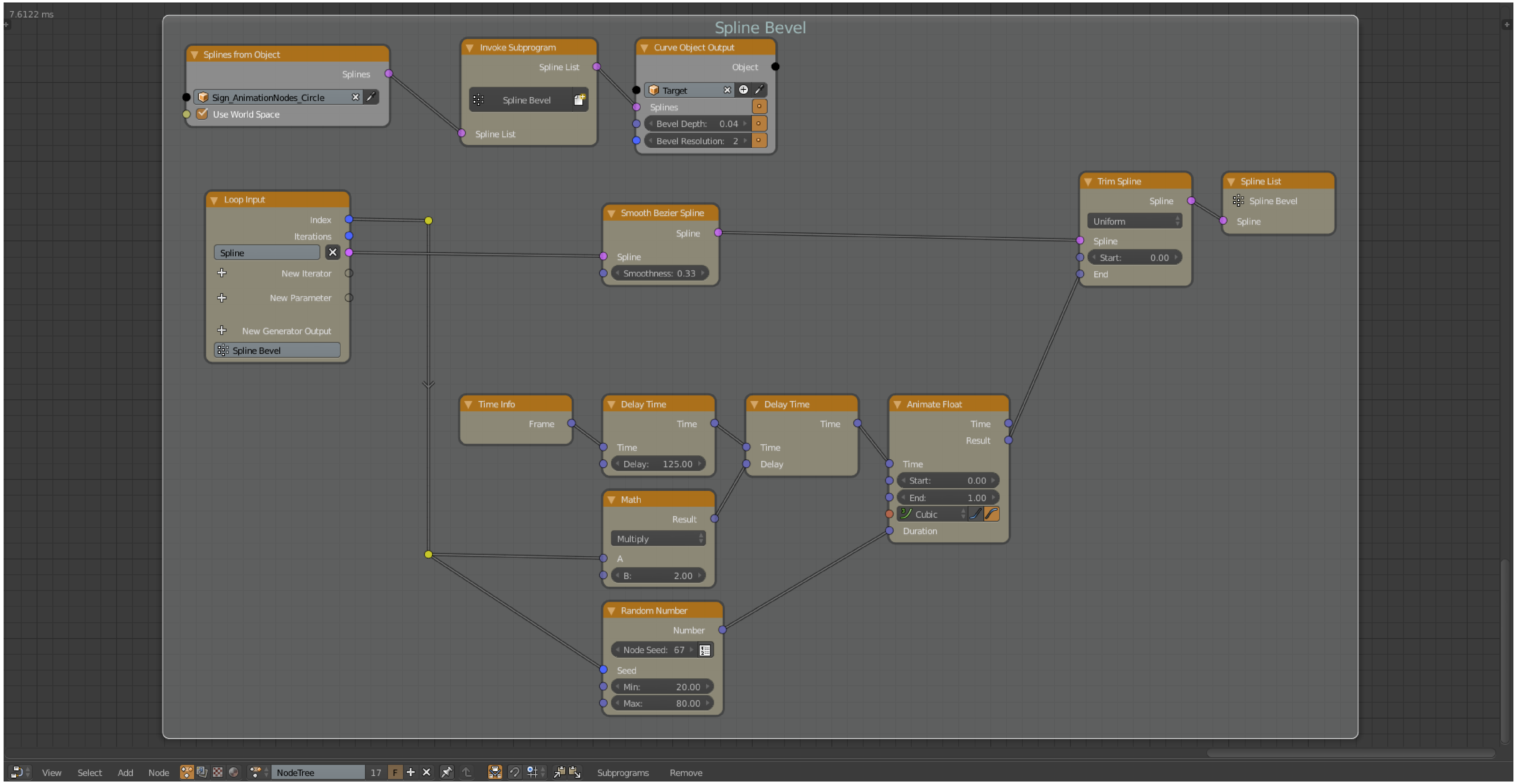
5.6.1 Spline bevel for the “Sign_AnimationNodes_Circle” animation using Animation Nodes

“Sign_AnimationNodes_Circle” is a curve object but different from “CurbStone”. The difference between these two objects is that “Sign_AnimationNodes_Circle” consist of four splines, shown in picture 49. Therefore, approach of this curve’s bevel animation differs from the one presented in section 5.3.1.



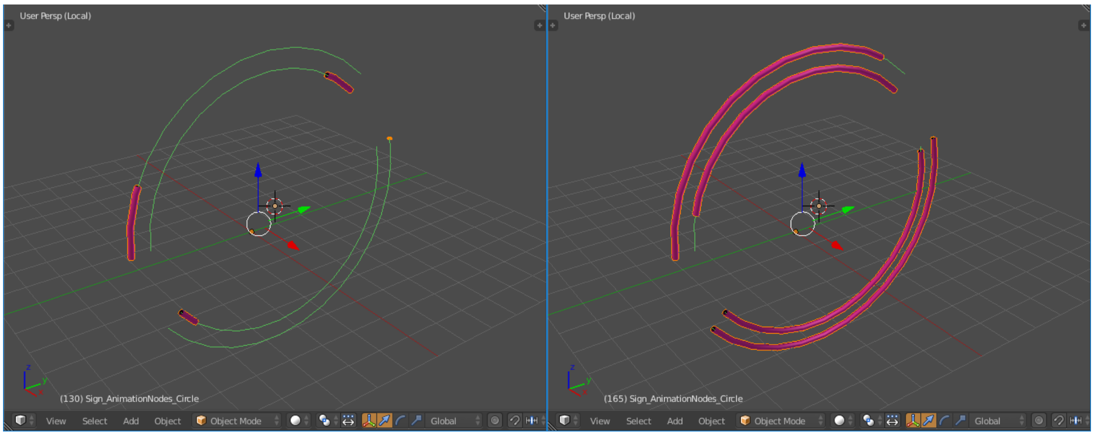
PICTURE 49: “Sign_AnimationNodes_Circle” splines (Tiss, 2017)

When using Animation Nodes, it is possible to separate splines in one object. This approach allows the animation of each spline individually, which means that it is possible to set animation delay, duration independently and randomly if such option is implemented in the node tree. To achieve independent spline bevel animation, a loop which iterates through the spline list, a reading from curve object is needed. The task of this loop is to give each spline different animation duration times and to set interval between the beginning of each spline's animation and after that to recreate a Spline List which can be passed to the Curve Object Output node. The described node tree's structure is represented in picture 50.



PICTURE 50: Node tree for “Sign_AnimationNodes_Circle” bevel animation (Tiss, 2017)

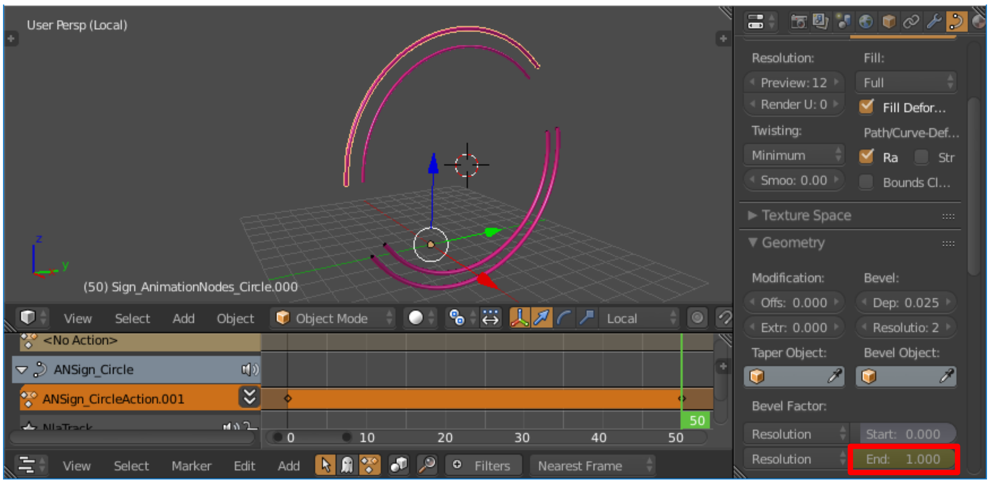
In picture 51 on the left side is shown how bevelling of each spline looks at frame 130 and on the right side how do they look at frame 165. Note that the bottom spline starts the last and finishes the animation first – this means that its animation delay is the longest but duration is the shortest.



PICTURE 51: "ANSign_Circle" bevel animation at frame 130 on the left and at frame 165 on the right (Tiss, 2017)

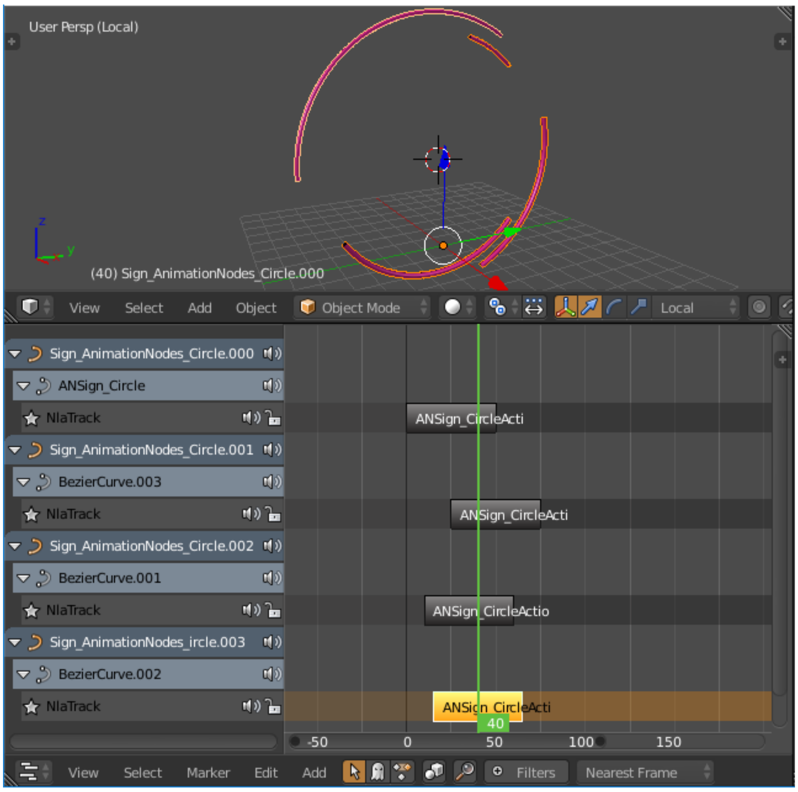
5.6.2 Spline bevel for "Sign_AnimationNodes_Circle" animation using keyframe animation method

To replicate the same animation by using a keyframe animation method the spline must be separated into individual curve objects. When using this specific animation method, it is impossible to access each spline of an object individually. Therefore, to imitate the animation made with Animation Nodes, user must make a new action with the Curve bevel animation where the Bevel Factor, End value must be animated manually from 0.0 to 1.0. The settings of this procedure are highlighted in picture 52.



PICTURE 52: “Sign_AnimationNodes_Circle” bevel animation settings with keyframe animation method (Tiss, 2017)

After making the action strip for at least one curve object, it is possible to link animation data as described in chapter 5.1.2. To achieve animation delay and different duration, action strips can be dragged and scaled along the time axis in the NLA Editor, shown in picture 53.



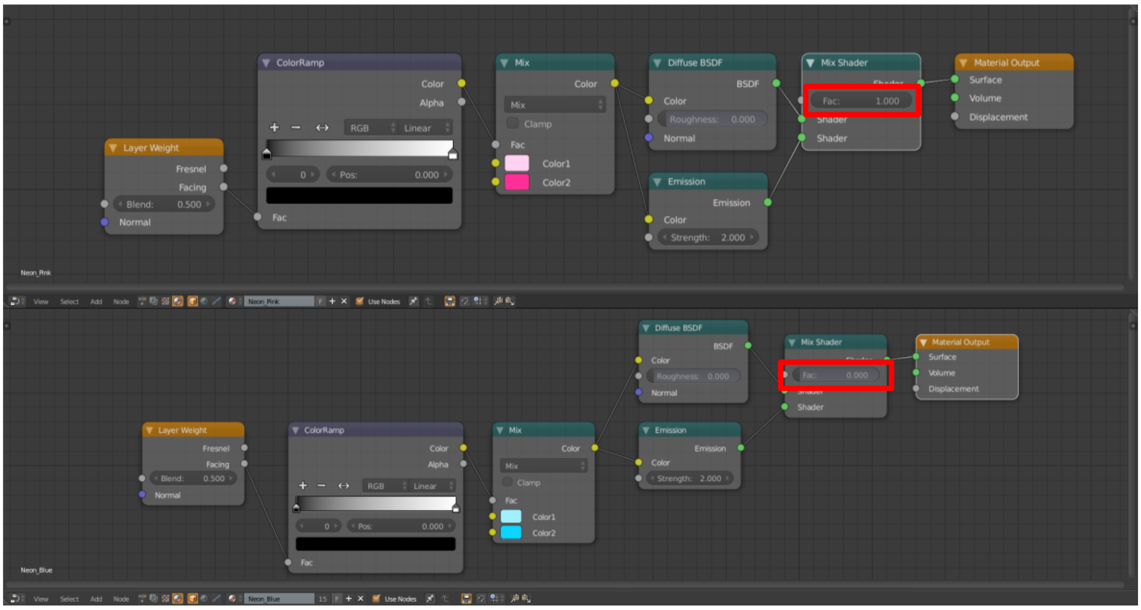
PICTURE 53: “Sign_AnimationNodes_Circle” splines’ action strip placement in NLA Editor (Tiss, 2017)

Afterwards animation strips can be dragged to desired location on the timeline.

5.7 “Neon_Blue” and “Neon_Pink” material wiggle animation

5.7.1 “Neon_Blue” and “Neon_Pink” material wiggle animation using Animation Nodes

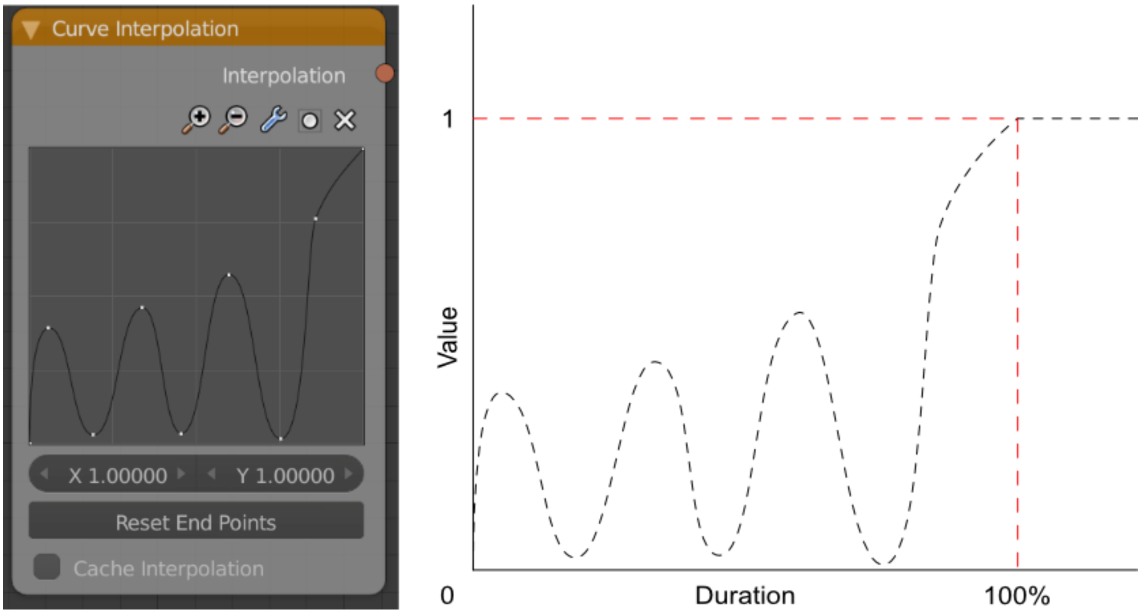
Animation Nodes can not only manipulate the object Tranforms, but also access Cycles material settings. It is possible to change shader settings by using the Cycles Material Output node. Two materials were created to achieve the “Sign_AnimationNodes_Circle” and the “Sign_AnimationNodes” wiggling shader effect: “Neon_Blue” and “Neon_Pink”. These two material node setups are shown in picture 54. The only difference between these materials are color values in the Mix RGB node.



PICTURE 54: “Neon_Pink” and “Neon_Blue” material setups in Material Node Editor (Tiss, 2017)

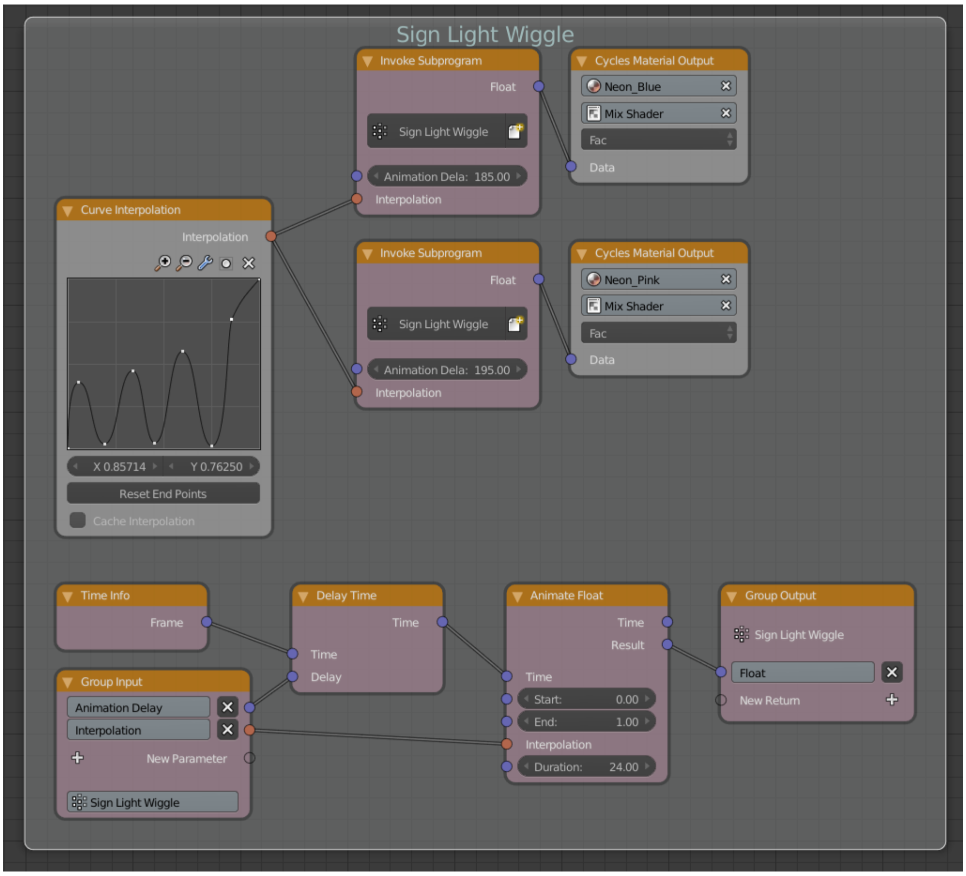
The Cycles Material Output node can access different material settings of each node in a certain material node tree. In this Case Study to achieve wiggling “on/off” light effect, the Mix Shader Factor value between the Diffuse and Emission shaders, highlighted in picture 54, is the setting of interest. Factor value is a float number between 0 and 1 which defines predominance between two input values. In this case predominance of the Diffuse BSDF over the Emission shader. When the factor value of the Mix Shader node is 0 then 100% of Diffuse BSDF and 0% Emission shader is used but when factor value is 1 – 0% Diffuse BSDF and 100% Emission shader is applied.

In Animation Nodes, there is a Number Wiggle node which wiggles number value in defined range. Even though this node does exactly what is needed to do, it cannot guarantee the end value of the animation to be 1. Therefore, a different approach is needed. Curve Interpolation node allows user to create a custom interpolation curve, giving the user more control over the animation and to set its behaviour as needed. In picture 55 is shown the Curve Interpolation node created for the case study on the left and its explanation graph on the right.



PICTURE 55: Interpolation Curve node for Mix Shader Factor value animation on the left and its graph on the right (Tiss, 2017)

Once the Interpolation is defined it is possible to animate the float number value in the desired way. To do that for two materials a reusable node group can be applied. In picture 56 is shown the "Sign Light Wiggle" node tree.



PICTURE 56: “Sign Light Wiggle” node tree (Tiss, 2017)

“Sign Light Wiggle” node group takes two input values: Animation Delay and Interpolation. The group animates the float number value from 0.00 to 1.00 with a Duration of 24 frames. Animate Float node animates output value in time based on the given interpolation curve. Later the Animate Float output value is set as a Factor for the Mix Shader nodes to the associated materials. In figure 8 are shown output values of the Animate Float node over time at frames 0, 8, 16 and 24 of the Duration.

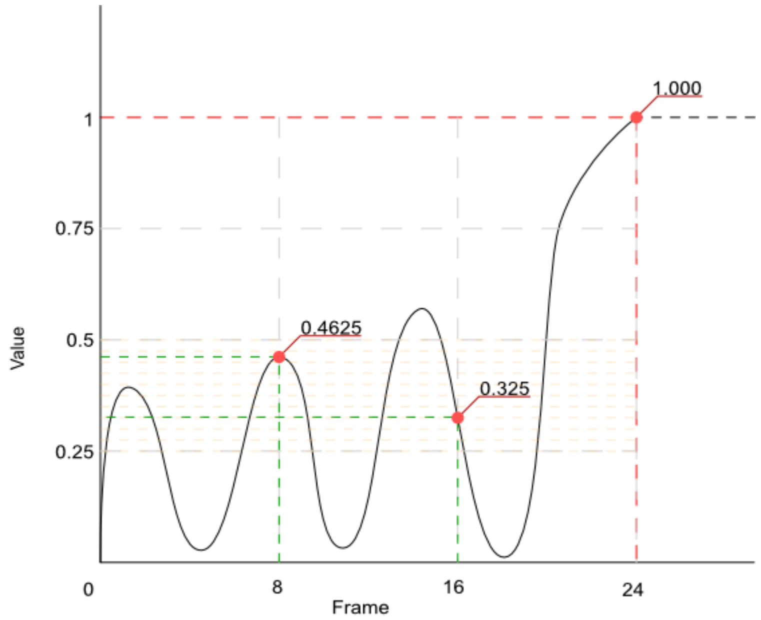
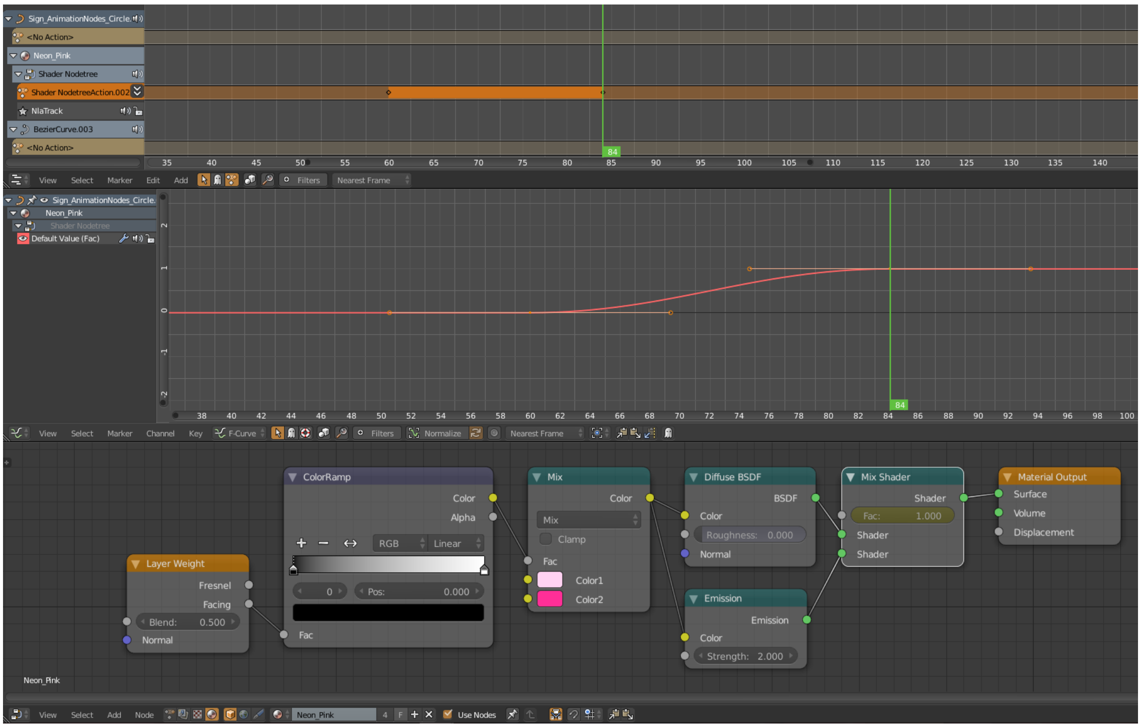


FIGURE 8: Output values of the Animate Float node over time at frames 0, 8, 16 and 24 of Duration (Tiss, 2017)

5.7.2 "Sign_AnimationNodes_Sign" and "Sign_AnimationNodes_Circle" material wiggle animation using the keyframe animation method

To replicate a created animation with the keyframe animation method, the factor value of the Mix Shader must be animated manually. Again, there are more than one approach to achieve the desired result: set many "on/off" keyframes or use Noise modifier in Graph Editor. In this case study, a Noise modifier is used. To apply the modifier between keyframes start and end values must be set. In picture 57 is shown the default interpolation between the start and the end keyframes of the Facture value animation.



PICTURE 57: Default interpolation between the start and end keyframes of the Factor value animation (Tiss, 2017)

Adding a Noise modifier eases and fastens the animation process. Even though a Noise modifier is randomly generated it offers several controls to adjust amplitude, phase, off-set, scale, depth and set a frame range. In picture 58 is highlighted set a Noise modifier to the Factor value interpolation curve with applied settings. After creating an animation strip, Animation Data can be linked to other materials as well and edited in the NLA Editor later.

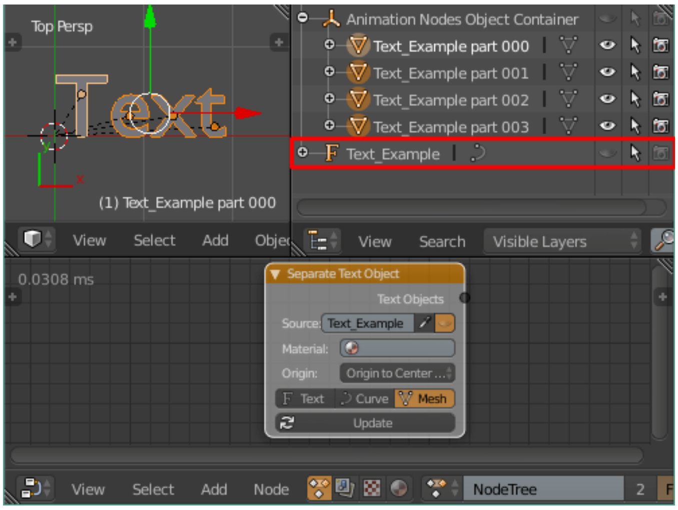
PICTURE 58: Set Noise modifier to the Factor value interpolation curve with applied values on the right (Tiss, 2017)

5.8 Text animation

5.8.1 Text animation using Animation Nodes

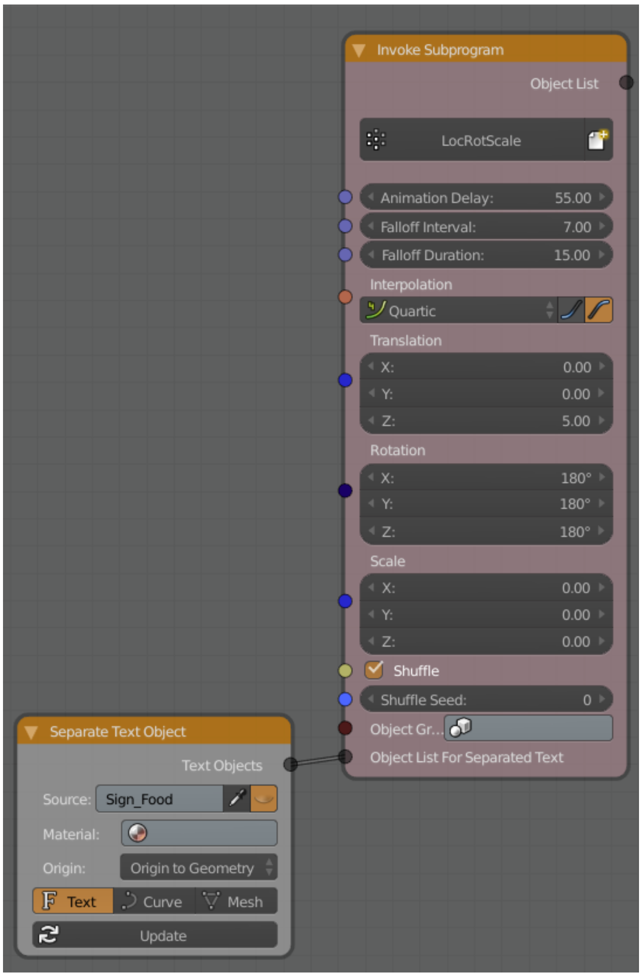
In this Case Study text animation is very like a mesh object animation. The text object must be divided in the way that each letter becomes a single object. To get the desired result their pivot points must be adjusted accordingly. Since each letter is a single object, it is possible to create an object group or an Object List of selected objects and reuse the LocRotScale node group.

When using Animation Nodes, a Separate Text Object node becomes handy. The node takes text object as input value, copies it and hides the original from the Viewport and Render. Copied text is converted into a specified object type, each letter separated and each pivot point adjusted accordingly. In picture 59 is shown the object hierarchy in the Outliner after the Separate Text Object node is executed, the text in 3D Viewport and the node tree in the Node Editor. Notice that the original "Text_Example" is hidden from the Viewport and the Render and there are four new objects created instead.



PICTURE 59: Object hierarchy in the Outliner after a Separate Text Object node is executed (Tiss, 2017)

This approach allows one to keep the original text object intended. The benefit of it is that the original text object can be changed after and by toggling "Update" in the Separate Text Object node, the output object list gets updated automatically. Since output the Object List is forwarded to a LocRotScale Invoke Subprogram node, no animation data is changed nor lost. In picture 60 is shown the combination of Separate Text Object and LocRotScale Invoke Subprogram node, that is used in the case study.



PICTURE 60. Separate Text Object node in combination with LocRotScale's Invoke Subprogram node (Tiss, 2017)

5.8.2 Text animation using keyframe animation method

To replicate text animation with keyframe animation method, few different approaches exist:

- Create certain amount of single letter text objects and adjust their pivot points manually,
- Create the text object, convert it to a Curve or a Mesh object, separate each letter individually and adjust their pivot points manually.

After creating separate objects and readjusting their pivot points, their animations are held the same way as other objects, described in section 5.1.2.

5.9 Main differences between Animation Nodes and keyframe animation method

Both methods deliver very similar outcomes but in two different approaches. Each method has its advantages and disadvantages. While working with Animation Nodes certain preparation is desirable to ease reusable node tree creation which takes time and some effort to do while keyframe animation method is straightforward. Despite the diversity in animation development, the main differences between these two methods reveal when user must tweak an animation.

When using Animation Nodes all settings can be tweaked in the Node Editor window. The advantage is that when tweaking certain settings, these settings are applied to the whole object group at the same time. This approach fastens animation adjustments.

Keyframe animation method is very flexible and allows to animate each object separately, giving them different settings. Settings created with this method can also be tweaked but for each object individually which makes the task very repeatable and time consuming.

5.9.1 Production value

Not always there is a clear winner when it comes down to animation tools or methods. Sometimes several methods combined can deliver most interesting, surprising and effective results. Even though many things are not covered in this Case Study, using the same principles it is possible to achieve many different effects. Also, few of most interesting effects are achievable when keyframe animation method is combined with modifiers and Animation Nodes. During the learning process, it was found that Animation Nodes is capable of a lot more than described in this thesis. Several author examples can be found in appendix 3.

DISCUSSION

Motion graphics has evolved dramatically. Nowadays there are plenty of 3D software suites available for motion graphic development. Which one is the best? Each of them has its cons and pros but in bigger scope they all share the same features and principles.

Many artists and studios prefer to use tools which are often called industry standards as Cinema4D, Maya, 3dsMax and others. Without a doubt those are great software suites that deliver high quality outcomes. Meanwhile Blender is somehow left outside the industry standard group. According to author's experience, it can be noticed in labour market where most animation studios prefer applicants with knowledge of industry standard software more than Blender users.

Blender is very powerful tool, when it comes to 3D artwork creation. It is possible to create flat design animations, complex photorealistic architecture renders, game assets and a lot more. Even though it is possible to deliver top quality content with Blender, it lacks something very important and valuable in daily work – full time technical support.

While big brands as Autodesk Inc. or Side Effects Software Inc. employ hundreds of people all around the globe to make their products great, Blender's development and support more relies on volunteer community and sponsors of Blender Foundation. Animation Nodes is going the same path.

Jacques Lucke develops Animation Nodes mostly by himself with some support from other programmers. He also gathers feedback from users and improves Animation Nodes with each release. He delivers OSS product by free will in his spare time along studies at the university without getting paid for it, except voluntarily donations from people. It is great effort and amazing human feature nowadays to give something away for free.

Animation Nodes is very a powerful Blender add-on already, regardless it is not, developed by professional software development team. Animation Nodes offers way more possibilities for motion graphics than described in this thesis. Even though it might seem complex and esoteric system at the beginning, it is becoming intuitive within a time.

As shown in the case study it is not always mandatory to use one or other technique to create certain type of animation. Many similar effects can be achieved in different ways. Main differences reveal when user must tweak the animation – it is way faster to adjust couple values in a node tree than drag or scale tens of keyframes or animation strips, which do not even fit the computer's screen. But on the other hand, setting keyframes manually can lead to better tuned animation.

Animation Nodes calculates everything in the background, which means that more data to process there are, the slower node tree execution and viewport update will be. This is something in what keyframe animation method is slightly ahead – it plays animation closer to real time in 3D viewport when shading is set to solid. On the other hand, it is possible to bake almost everything that is made with Animation Nodes to keyframes allowing to speed up viewport update.

From user perspective, it is almost impossible to compare which animation method is better or worse, faster or slower. It highly depends on personal knowledge and preference. For example, Midge Sinnaeve (www.themantissa.net), a Blender artist from Antwerp, Belgium, produces abstract procedural motion graphics and VJ loops mostly using modifiers and other features that are already implemented in official Blender releases.

"A lot of people tell me that AN absolutely awesome, and everything makes sense, and love it. But at least the same amount of people tells me that it is stupid to give artists a tool that works like a programming language." (Lucke 2017).

Animation Nodes truly is based on programming rules and for AN user it is only beneficial to understand programming principles. Even though it might be complicated for beginners, for more advanced users this approach allows to create new and unseen effects.

Within time of using Animation Nodes, the add-on becomes quite intuitive and usage of it becomes logical. Its abilities of object animation are tremendous and once combined with built in features like modifiers or keyframe animation method – almost infinite. For author, it has become an everyday tool.

Everything has a workaround. There is no bad or great software to create art – tool does not make an artist. Blender and Animation Nodes are OSS which allow people to make art for free and with some effort involved it is possible to deliver at least the same if not even better results than paid software suites can.

REFERENCES

Adobe Systems Inc. Adobe After Effects Licensing Cost. Read 24.10.2017. https://www.adobe.com/fi/creativecloud/plans.html?single_app=aftereffects&pro-moid=RPZBNCW8&mv=other

Animation Nodes. 2017. Interface. Animation Nodes Manual. Released 27.08.2015. Updated 20.06.2017. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/getting_started/interface.html#node-settings

Animation Nodes. 2017. Get List Length. Animation Nodes Manual. Released 27.09.2016. Updated 03.05.2017. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/nodes/list/get_list_length.html

Animation Nodes. 2017. Group. Animation Nodes Manual. Released 29.08.2015. Updated 29.05.2017. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/subprograms/group.html

Animation Nodes. 2017. Loop. Animation Nodes Manual. Released 29.08.2015. Updated 29.05.2017. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/subprograms/loop.html

Animation Nodes. 2017. Script. Animation Nodes Manual. Released 29.08.2015. Updated 19.06.2017. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/subprograms/script.html

Animation Nodes. 2015. Socket Types. Animation Nodes Manual. Released 25.08.2015. Updated 04.09.2015. Read 24.10.2017. http://animation-nodes-manual.readthedocs.io/en/latest/dev_guide/socket_types.html

Animation Nodes. 2017. Subprograms. Animation Nodes Manual. Released 29.08.2015. Updated 13.05.2017. Read 24.10.2017 http://animation-nodes-manual.readthedocs.io/en/latest/user_guide/subprograms/index.html

Autodesk Inc. 3ds Max Licensing Cost. Read 24.10.2017. https://www.autodesk.eu/products/3ds-max/subscribe?plc=3DSMAX&term=1-YEAR&support=ADVANCED&quantity=1

Autodesk Inc. Maya Licensing Cost. Read 24.10.2017. https://www.autodesk.eu/products/maya/subscribe?plc=MAYA&term=1-YEAR&support=ADVANCED&quantity=1

Blender Foundation. About. Read 21.10.2017. https://www.blender.org/about/

Blender Foundation. Introduction to Nodes. Blender 2.79 Manual. Read 20.10.2017. https://docs.blender.org/manual/en/dev/render/blender_render/materials/nodes/introduction.html

Blender Foundation. Node Editor. Blender 2.79 Manual. Read 29.09.2017. https://docs.blender.org/manual/en/dev/editors/node_editor/introduction.html

Crook, I. & Beare, P. 2015. Motion Graphics Principles and Practices from the Ground Up. Bloomsbury Publishing Plc.

HistoryofInformation. Lee Harrison's Scanimate: The First Widely Applied Analog Computer Animation System. Read 02.11.2017. http://www.historyofinformation.com/expanded.php?id=4469

Lampel, J. 2015. The Beginners Guide to Blender. eBook. Read 23.10.2017. http://www.blenderhd.com/wp-content/uploads/2015/08/BeginnersGuideToBlender.pdf

Lucke, J. Developer of Animation Nodes. 2017. Skype interview 04.04.2017. Interviewer Tiss, R.

MarketsandMarkets™ Research Private Ltd. 2017. 3D Animation Market by Technology (3D Modeling, Motion Graphics, 3D Rendering, and Visual Effects), Vertical (Media and Entertainment, Healthcare and Lifesciences, Manufacturing), Services, Deployment, and Region - Global Forecast to 2022. Released April 2017. Read 21.10.2017.

http://www.marketsandmarkets.com/Market-Reports/3d-animation-market-92736816.html

Maxon Computers GmbH. Cinema 4D Broadcast R19 Licensing Cost. Read 24.10.2017. http://www.maxonshop.de/epages/es10565932.sf/de_DE/?ObjectPath=/Shops/es10565932/Products/%2219%204%2000%22/SubProducts/%2219%204%2000-0002%22

Roosendaal, T. 2011. Blender Wiki. Read 29.09.2017. https://wiki.blender.org/index.php/Dev:Source/Architecture/Overview

Scanimate: The Origins of Computer Motion Graphics – Film. 2016. Lynda.com. Released 29.06.2016. https://www.lynda.com/Motion-Graphics-tutorials/Documentary-Feature/474683/510578-4.html

Side Effects Software Inc. Houdini Indie Licensing Cost. Read 24.10. 2017. https://www.sidefx.com/buy/#houdini1

Slick, J. 2016. List of 3D Software – Full 3D Suites. Read 24.10.2017. https://www.lifewire.com/full-3d-suites-2006

Slick, J. 2017. The Best Free 3D Software To Download. Read 21.10.2017. https://www.lifewire.com/free-3d-software-list-2005

Steinmetz, F. & Hofmann, G. 2014. The Cycles Encyclopedia. Your Guide to Everything Cycles. Version 1.4. eBook. Blender Diplom.

TechTerms. Node. Read 24.10.2017. https://techterms.com/definition/node

APPENDICES

Appendix 1. Skype Interview with Jacques Lucke 06.04.2017.

1. Please tell briefly about yourself.

I am Jacques Lucke, currently I am 19 years old. I live in Hennigsdorf (very close to Berlin). Now, I am 3rd semester student of computer sciences. Got into programming, video/image editing, vfx and other related stuff about 9 years ago.

2. What is Animation Nodes?

Animation Nodes is a node based visual scripting system designed for motion graphics in Blender. If you consider all these things (nodes, visual, scripting, designed for motion graphics, Blender, ...) together, you'll get Animation Nodes. I call it a "system", because it is not just a language but a whole framework and "designed for", because it's mainly used for motion graphics, but it was always a goal to be able to do a lot more with it.

One interesting fact is that AN is completely open source and available since the first day of development. On the second day, I uploaded the first video about AN on my Youtube channel (it's still online) and started the AN thread on www.blenderartists.org. Now the thread is the largest in that category of the forum.

3. How and why did you start to develop Animation Nodes?

I have been using Blender for 5 years now, started to develop add-ons 2.5 years ago.

My first add-on (to get started with the Blender Python API) was the Sniper add-on. My second add-on for Blender is already Animation Nodes (note that I had quite a bit of programming knowledge before).

I had many ideas for different Motion Graphics add-ons for Blender but I found that many of them share the same base. So, I tried to figure out a way to combine all that stuff into one add-on (also because I was lazy and did not want to write and maintain that many add-ons).

Making it node based was just a logical conclusion. I was fascinated by node based systems for a few years already (since I worked with Nuke during a two-week internship) and thought about developing one myself for a long time.

4. What is Animation Nodes capable of?

It's hard to say what is Animation Nodes capable of because it is a scripting system. It's like asking what Python or any other programming language is capable of. It has hundreds of nodes built in already for motion graphics, math, ...

One of the great strengths of Animation Nodes is its high flexibility and extensibility. I heard of people who developed their own set of nodes for whatever they are trying to achieve. That does not have to do something with Motion Graphics.

5. What are cons and pros of Animation Nodes?

Talking about pros and cons in general is also not easy. It always depends on what you are trying to solve. Also note that Animation Nodes is never really finished, so talking about individual features as cons might not be a good idea. Also, many things are pros for some people and cons for others. For example, the fact that it works very similar like a programming language. A lot of people tell me that AN absolutely awesome, and everything makes sense, and love it. But at least the same amount of people tells me that it is stupid to give artists a tool that works like a programming language.

This makes it sometimes difficult to make decisions of course. It's hard, if not impossible, to design software for so many different people.

6. What is the future of Animation Nodes?

I'm currently working on the AN 2.0 release. It will either come out in roughly 2-3 months or in summer, depending on how far I get when I have holidays. Big goals for the future (unlikely that they make it all into the same release):

- Cython: Cython is awesome, it allows me to optimize AN where performance is important. I plan to "cythonize" more aspects of AN to make it faster for many common things.
- Vectorization: example - if you put a single float value into a math node, it should operate on this element. However, when you plug in a list of floats you want it to operate on all the floats at once. I don't want separate nodes for that. (It's like function overloading if you know the terminology). This will have many benefits as you will see in the next release.
- Declarative: So far most of the time you want to do something in AN you can't just say what you want, but you have to know how to get there. It would be more artist friendly the other way around. The first step into this direction is the new

Falloff system I'm working on. Checkout my latest videos to see roughly how it works.

- Plugins: Allow others to develop their own nodes for AN and quickly share them with others via a little plugin manager within AN.

This is what I can think of right now. I'm sure there is more but I don't have and never had a master plan for AN so I'll adjust priorities on the go.
Also, the users always have high influence on the areas I work on.

Appendix 2. Case Study's production files.

All files in this appendix were created by the author of this thesis and for undefined time frame are available online:
https://drive.google.com/drive/folders/1w3919GIb0QZ6mYVA0v6uiL4SsGOaPrO-?usp=sharing

- CaseStudy_AN.blend - Blender file with case study's animation developed with Animation Nodes
- CaseStudy_Keyframe.blend - Blender file with case study's animation developed with keyframe animation method
- CaseStudy_AN_30fps.mp4 - rendered animation of the case study developed with Animation Nodes at framerate 30fps
- CaseStudy_AN_60fps.mp4 - rendered animation of the case study developed with Animation Nodes at framerate 60fps
- CaseStudy_Keyframe_30fps.mp4 - rendered animation of the case study developed with keyframe animation method at framerate 30fps
- CaseStudy_Keyframe_60fps.mp4 - rendered animation of the case study developed with keyframe animation method at framerate 60fps
- CaseStudy_Keyframe_vs_AN_30fps.mp4 – side by side comparison of CaseStudy_Keyframe_30fps.mp4 and CaseStudy_AN_30fps.mp4
- CaseStudy_Keyframe_vs_AN_60fps.mp4 – side by side comparison of CaseStudy_Keyframe_60fps.mp4 and CaseStudy_AN_60fps.mp4

Appendix 3. Examples created during the learning process.

All files in this appendix were created by the author of this thesis and for undefined time frame are available online:
https://drive.google.com/drive/folders/1-TxMmJRQG2lv7EYRqS_ffYlbl7EmJ77t?usp=sharing

- Amber_Rings.mp4 – animation created combining keyframe animation method and Animation Nodes
- Pipes.mp4 - animation created combining keyframe animation method, modifiers and Animation Nodes
- Spikes.mp4 - animation created combining keyframe animation method, modifiers and Animation Nodes
- Spiral.mp4 - animation created combining keyframe animation method and Animation Nodes
- Tapes.mp4 - animation created combining keyframe animation method and Animation Nodes
- Towers.mp4 - animation created combining keyframe animation method and Animation Nodes
- Tubes.mp4 - animation created combining keyframe animation method, modifiers and Animation Nodes