International Journal of Internet, Broadcasting and Communication Vol.14 No.3 155-160 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.3.155

IJIBC 22-3-19

Study of Script Conversion for Data Extraction of Constrained Objects

Choi, Chul Young

Major of Animation and Visual Effects, Dongseo University, Professor, Korea
freechoi@dongseo.ac.kr

Abstract

In recent years, Unreal Engine has been increasingly included in the animation process produced in the studio. In this case, there will be more than one of main software, and it is very important to accurately transfer data between the software and Unreal Engine. In animation data, not only the animation data of the character but also the animation data of objects interacting with the character must be individually produced and transferred. Most of the objects that interact with the character have a condition of constraints with the part of character. In this paper, I tried to stipulate the production process for extracting animation data of constrained objects, and to analyze why users experience difficulties due to the complexity of the regulations in the process of executing them. And based on the flowchart prescribed for user convenience, I created a program using a Python script to prove the user's convenience. Finally, by comparing the results generated according to the manual flowchart with the results generated through the script command, it was found that the data were consistent.

Keywords: *Animation Production, Python script, Parent Constrain, Unreal Engine, Maya software*

1. Introduction

In the past, it was easy to distinguish the method of producing video content. The software and production processes used for animation, game, and visual effects of film were easily distinguishable. However, in recent years, the game market has grown rapidly and the spread of game engines capable of realizing realistic images is further expanding the scope of contents using engines. The lumen function, which has been implemented since Unreal Engine 5, enables realistic lighting with very simple operation. All of the animations in Figure 1 were created using Unreal Engine, and the proportion of Unreal Engine used in the production process is very high. Conversely, the fact that it occupies a high proportion means that the advantages of existing CG software such as Maya still exist. Therefore, there will be more than one of main software, and studios will organize the production process using these software and Unreal Engine together. In this way, large and small problems occur in transferring the object or animation data produced by software to other software. For example, a digital actor will always wear objects such as clothes or necklaces, which move with the actor as he moves. And digital actors are always interacting with objects around them. These objects are in a constraint relationship with the character. To import animation data of objects in a constraint

Manuscript Received: June. 22, 2022 / Revised: June. 24, 2022 / Accepted: June. 27, 2022
Corresponding Author: freechoi@dongseo.ac.kr
Tel:*** - **** - ****
Major of Animation and Visual Effects, Dongseo University, Korea

relationship into Unreal Engine, I need to output the animation data individually. In this paper, we will deal with certain rules necessary to extract animation data from these objects.

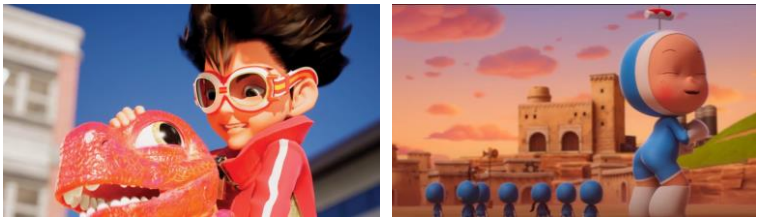


Figure 1. Animation using Unreal Engine Dino Power (left) Yumi's cells (right)[1][2]

When animators create and modify character animations using Maya software, they often use group or parent commands because it is easy to use when constraints between objects are required. However, if Constraint on/off is frequently required, various constraint commands such as parent, point, orient, scale, and aim are used to establish the relationship between objects. In addition, the character rig provided by Maya such as Human IK provides a function called 'Effector' that can control the constraint of up to 4 different objects per joint.

Table 1. Constraint relationship between objects

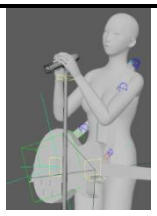
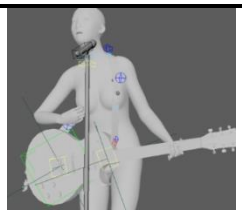
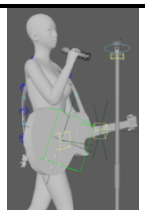
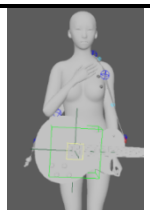
Hand Constrain	Case 1	Case 2	Case 3	Case 4	Case 5
					
Other Hand	O				
grab the mike	O	O		O	
Mike fixed to the stand	O	O			
Guitar on Character	O	O	O	O	O
Head of Guitar			O		

Table 1 shows the conditions for the constraint of objects as an animation I worked on.[4] In case 1, a character rigged with Human IK has both hands on a microphone placed on a stand. In this case, when the right hand is moved, the constraint is set so that the microphone, the microphone stand, and the left hand also move. In this case, it is necessary to constrain the hand to the other hand, but caution is required. When I use the constraint function directly from one part of the body to another, a problem arises in character rig. In this case, the constraint function must be used indirectly, and the method that can be used in this case is to use a locator as an intermediate medium.

2. Data extraction of constrained objects

2.1 Production Pipeline

Table 1 shows the action of holding and releasing the microphone and guitar while the digital actor is singing. During the animation, I can see how many constraint relationships between hands and objects are turned on/off. The constraint function is also set by default between the microphone and the microphone

stand, and between the guitar and the digital actor. It can be said that a very complex constraint relationship is established between digital actors and objects in one animation scene. In this situation, a problem occurred in the process of outputting the animation data of characters and objects as an independent file according to the request of the client. The constraint function can be produced by using the 'group, parent, constrain' functions. When using functions other than constrain, the output animation data comes out as '0' in the graph of the graph editor as shown in Figure 2. I was able to submit the final result by working again to extract the correct animation data of the constrained object. The problem with this process was that, because the work process was chaotic, the operator who was originally working needed to proceed with the work to the end. This can be said to be a hindrance factor in making an efficient work process. In the process of creating animations using 'MetaHuman[3]', it is essential to use Unreal Engine. And it is convenient to use Maya software to easily create character animations. Therefore, it can be said that the process of transferring data from Maya to Unreal Engine is very important in the animation production process using 'MetaHuman'. As mentioned earlier, the process of transferring data of constrained objects seemed complicated, so I tried to define the production process as shown in Figure 3 for this process and identify the problems.

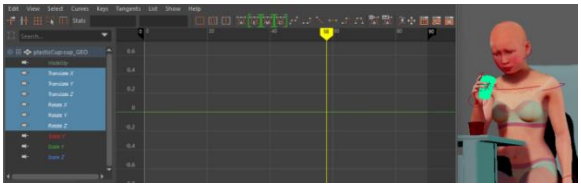


Figure 2. Animated curves of cups grouped by parenting in 'MetaHuman' right hand

In order to extract the animation data of the cup without any problems, a flowchart for object constraint as shown in Figure 3 is required. The most important element in the flowchart is the order of applying the constraint function between an object such as a cup in hand and an object constrained by it. However, if I use the constraint function on two objects at this time, a problem arises in character rigging. In this case, instead of connecting the objects directly, I can solve the problem by connecting them using a locator in the middle. For example, as in case 1 of Table 1, a problem occurs when one hand of Human IK has to use the constraint function on the other hand. So, as shown in the flowchart, a locator is created, and the constraint relationship between two objects and a locator is defined after that. After creating the locator, the process of adjusting the position of the locator to the position of the cup proceeds as shown in step 2, Figure 3. When I use the parent constrain function with maintain offset checked off, I can move one object to the position of another object. After using this function to move the locator to the position of the cup, I must delete the constraint history from the channel window before proceeding to the next step.

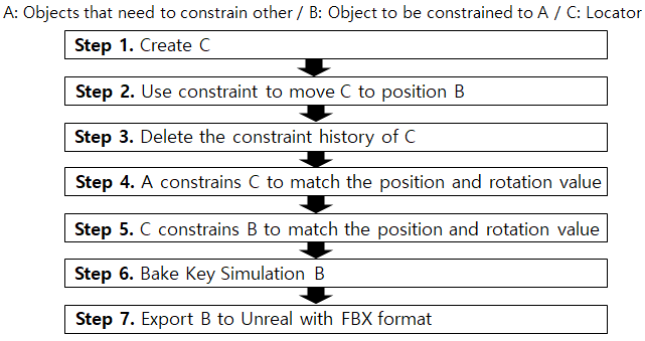


Figure 3. flowchart for data extraction of constrained objects

In step 3 of Figure 3, the constraint relationship between the hand of the 'MetaHuman' and the locator is established so that the locator moves according to the movement of the hand. In step 4 of Figure 3, the constraint relationship between the cup and the locator is established so that the cup moves according to the movement of the locator. In the last step, the key bake simulation process is performed so that the movement of the cup can be displayed as a curve graph in the graph editor window. Once that's done, the cup will have individual animation data and can import animation data into Unreal Engine.

2.2 Python script development

Following the flowchart shown in Figure 3 can be a difficult process for the operator if it is not the operator's area of expertise. It is necessary to establish a constraint relationship for a total of three objects by including two objects and a locator. It is a difficult process because the result varies depending on the order of the objects selected by the operator. This is more likely to cause problems in spaces with multiple workers rather than one. If the process of transferring animation data between different software such as Maya and Unreal Engine is absolutely necessary in the company's animation production process, the data extraction process of these constrained objects is a very important process. Therefore, it was decided that it is necessary to find a method that the operator can use easily for the manufacturing process shown in Figure 3. For this reason, I tried to create a program using a Python script for the animation data extraction process of the constrained object. At the beginning of production, I tried to develop the script step by step based on the flowchart in Figure 3. However, in the process of development, the process of using the 'parent constrain' function to move the locator to the position of the cup seemed complicated. So, when creating a locator, the same result could be produced by finding the position value of the selected cup and moving the locator to this place. So, as shown in Figure 4, when creating a locator, I made a function to find the position of the first selected object by setting the position and rotation values. I checked the location using the print function to see if the result is the same.

```
# Make locate and move to position of Prop
def make_locator(m):
    select_obj = cmds.ls(sl = True)
    lo = cmds.spaceLocator( )
    position = cmds.getAttr(select_obj[0]+'.translate')
    rot = cmds.getAttr(select_obj[0]+'.rotate')
    cmds.setAttr( lo[0]+'.translate',position[0][0],position[0][1],position[0][2])
    cmds.setAttr( lo[0]+'.rotate',rot[0][0],rot[0][1],rot[0][2])
```

Figure 4. Function script for locator creation and movement

And when designing the program window, in the beginning, I tried to make a command button step by step in the flowchart. In this process, several elements that needed correction were discovered. First, the existence of the locator is very important in the manufacturing process, but the user does not need to know the existence of the locator. Since the user is only interested in the constraint relationship between the cup and the handle of the hand that needs restraint, it was judged that the button to create a locator makes the user confused in understanding the relationship. Second, it was judged that there is no need to create a button at each step of applying the inter-object constraint function. In a flowchart, I defined the steps because the order of the steps is important. However, many buttons can confuse and complicate users, and it would be better than complicated if one button could handle all the processes.

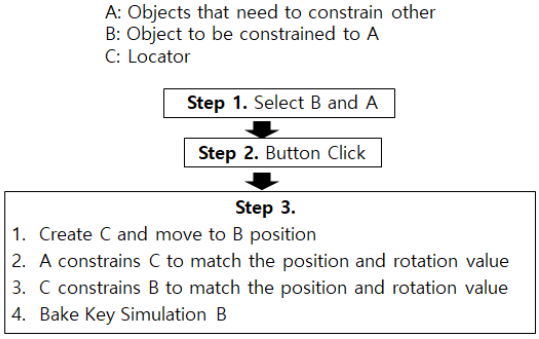


Figure5. Script flowchart for data extraction of constrained objects

This is the easiest way for users to understand what the script is doing. So, when the user selects only the cup and the handle of the hand and clicks the command button, the animation data of the cup is generated as a curve in the graph editor window. The flowchart of the Python script developed in this way is shown in Figure 5. According to the flowchart in Figure 5, the execution process was developed as a Python script. Script development proceeds in three steps. First, a pop-up window that appears when the program is run and a button that can give commands were made. Second, I made a locator and designated a function to find the position value of the object to be constrained and move it. Third, the parent constrain process was created using the two selected objects and a locator. After that, the command was designated to show the movement of the first selected object in the graph editor window by performing a bake simulation. Figure 6 shows the window that pops up when the developed final script and program are executed.

Figure 6. Completed Python script(left), Popup Window (right)

Using the created Python script program, I tried to extract the animation data of the cup using the scene as shown in Figure 7. When I click the command button after selecting the cup and the handle of the hand, I can see that the cup movement is bake simulated as a graph with key-frames applied in the graph editor. The animation curve data of the cup created using the script was shown to be consistent with the result produced by following the flowchart in Figure 3, thus proving the performance of the script.

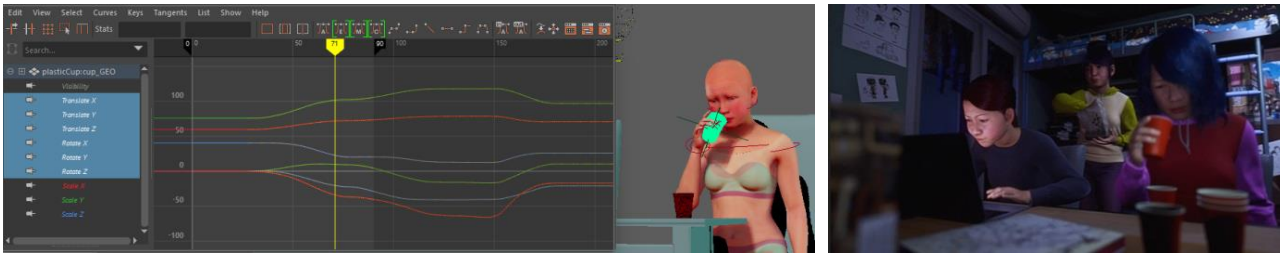


Figure 7. Script applied animation curve (left) Applied curve to Unreal Engine (right)

3. Conclusion

Tools that can easily create realistic, customizable characters such as 'MetaHuman' are continuously reducing the work processes required for animation production. And in order to increase the production efficiency of animation or video content using these tools, it is necessary to establish a production process by linking the strengths of each tool. This situation showed that as mentioned above, problems occur in the process of connecting software and work, and if the method of solving this problem does not match the professional field of the operator, it can be difficult for the operator. This situation showed that, as mentioned above, a problem occurs in the process of exchanging data between software, and it can be very difficult for the operator if the method to solve this problem does not match the operator's professional field. In fact, when the process of Figure 3 was explained to animation students in a university class, most of the students complained of difficulties in understanding and practicing. In the case of a problem that is somewhat difficult to understand, such as setting the relationship between objects, I decided that it would be very helpful to the user to support it through the development of a program with a simple function using a Python script, so I made a program and let the students try it. As a result, the students showed how to easily use the program to output animation data.

References

[1] Dofala Studio: https://www.dofala.co.kr
[2] Yumi's Cells : https://www.youtube.com/watch?v=X7-rBDXEpRc/
[3] MetaHuman : https://MetaHuman.unrealengine.com/
[4] Gu-jie, Chulyoung Choi, "Case Study of Motion Capture Data Processing for Actual Digital Actor Motion Implementation," The Korean Journal of Animation, Vol.14, No.2, pp. 23~41, June 2018. DOI : 10.51467/ ASKO.2018.06.14.2.23
[5] Balgum Song, Dong-hyuk Choi, "A Study of Retargeting Motion Capture Data to Unlimited Characters", Cartoon and Animation Studies, vol. 57, pp.159~185, Dec 2019, DOI : 10.7230/KOSCAS.2019.57.159
[6] Ju-Hyun Nam, "A Study on Efficient Motion Data Editing of Motion Capture System : Focused 'Fighters Club' Character" , Journal Article published 30 Jun 2014 in , JKGS, Vol.14, No.3, pp.25~33, June 2014. DOI : 10.7583/JKGS.2014.14.3.25
[7] Yangyang He, Chulyoung Choi, "A Study of Facial Expression of Digital Character with Muscle Simulation System," IJASC, Vol.8, No.2, pp. 162-169, June 2019. DOI: http://dx.doi.org/10.7236/IJASC.2019.8.2.162
[8] Chulyoung Choi, "Study of Character Animation to improve Production Efficiency," IJASC, Vol.9, No.2, pp.179~184, June 2020. DOI : http://dx.doi.org/10.7236/IJASC.2020.9.2.179
[9] Chulyoung Choi, "Case study of Creating CG Handheld Steadicam using maya nParticle", IJASC, Vol.10 No.3 157-162, Sep 2021, Doi: http://dx.doi.org/10.7236/IJASC.2021.10.3.157
[10] Ryu seuc-Ho, Park Yong-Hyun, Kyung Byung-Pyo, Lee Dong-Lyeor, " 3D Game Character Animation Pipe -line to Improve Utilization of Motion Capture", JKCA, Vol.8,No7, pp.120~127, July 2008, DOI: https://doi.org/10.5392/JKCA.2008.8.7.120