PuppetMan

Martin Rojas

Abstract

Currently machinima tool development has been focused on camera and scene controls. These tools are very important, but have created a void in tools for the control of the performing characters in machinima pieces.

Introduction

The inspiration of the project was a mixture of wanting to create more expressive machinima and being able to share actions. After trying to build a machinima performance, it was clear that there was a lack of control in the quality and type of movement that the characters were able to perform.

The lack of expressiveness in the characters hampered the quality of the performance and the potential for complex performances. This block on complex performance has the potential to hamper the growth of machinima as a serious competitor to pre-rendered animation. It was due to this void that the idea of the creation of a tool that would solve this problem emerged.

Previous / Related Research

Good documentation on machinima is a bit sparse. Even though the concept has been forming since the first 3D game, but it has not been until recently that the quality of the games have made it possible for legitimate arguments to be made to the possibilities of replacing old style pre-rendered animation.

Some of the previous research that has address control of dame characters in a more details level.

- *Puppet Show DWIG, Michael Nitsche, Devon Hunt, Alex West*
- *TUI3D, DWIG, Michael Nitsche, Alie Mazalek*
- *enBodied Digital Creativity DWIG, Michael Nitsche, Alie Mazalek, Tandav Krishna*

Design/ Constraints

The first iteration of the tool was for creating an environment that would allow a bridge to connect game controller to the bones in the Unreal Characters so that smoother motions could be performed by the characters. When creating this tool the problem of not being able to control more than a couple of bones at a time were cause for rethinking the capabilities of the tool.

In order to solve the problem about the inability to control more than one bone the idea of recording the input for each bone appeared. The next stage was to create a recording capability which focused on saving the input from the controller and allowing for a progressive generation of the performance. Through this iteration the problem of creating an efficient storage structure capable of reading, writing and changing close to 100 values per second was very difficult.

After a comprehensive architecture was created for handling the data stream, allowed for the next challenge in the creation of the tool. The need for a clean a simple user interface was necessary since the tool would run on the background or next to the puppet and not have the full attention of the user. While easy enough to provide the data on the current status of the tool at a simple glance from the user, it was this potential lack of disability that prompted the last feature of the tool.

A feature which focuses on mapping all of the inputs needed to handle the tool from the Xbox controller.

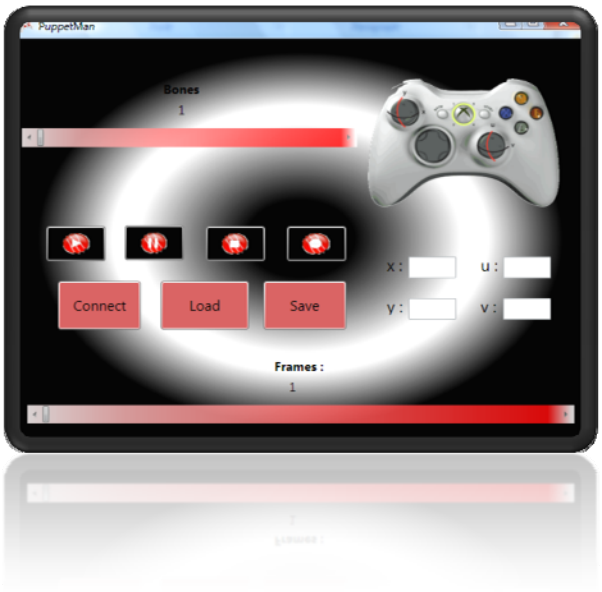
Technologies

The tool was created on the .NET framework and written in C#. The reasoning for the use of this language rather than Java was that the .NET framework uses a lot of libraries that are standard on almost all windows machines. These libraries include the xbox controller and the correct udp server to send the information to Unreal tournament. The last reason for choosing that language is that the language is part of an environment that is constantly improving and keeping with times and not likely that it will disappear in the recent future rendering the tool useless.

Second the tool connects to the Unreal Tournament 2004 mod Movie Sandbox. In this mod a character had to be created with the right bones in mind in order to set all of the movements for the character.

Final Prototype

The final prototype of the system adhered to the parameters set in the design process of the tool. While there may still be some bugs in the system and not as versatile as previously expected. Although the system was designed from the beginning to be adept to change and very flexible in its implementation. This was a priority and is true in its final inception since all of the parameters need to be changed in a few places and cause for the entire program to be able to adapt to new parameters.



Conclusion

The puppetMan tool has achieved most of the goals that were set in its design process. Yet the result was not as elegant as expected since there were some restrictions placed on program by the Unreal Engine. These restrictions that caused for a reconstruction of the tool changed some of the original ideas of interoperability in the simple form of the tool. The possibility of interoperability is still present in the tool since the entire save and load functions still work.

It is clear that puppetMan is a first step into a tool that will open up the possibilities to a new field in film and expression.