Stadium Crowd Tool in Houdini with Key-Frameable Motion Controls

Mei-Chieh Chen

Submitted in Partial Fulfillment of the Requirements
For the Degree of Master of Fine Arts in Visual Effects
at
Savannah College of Art & Design

© August 2013, Mei-Chieh Chen

The author hereby grants SCAD permission to reproduce and to distribute publicly paper and electronic thesis copies of document in whole or in part in any medium now known or hereafter created

__________________________________________________________/______/____________
Mei-Chieh Chen
Author

__________________________________________________________/______/____________
Deborah R. Fowler
Committee Chair

__________________________________________________________/______/____________
Stuart Robertson
Committee Member

__________________________________________________________/______/____________
Ashwin Inamdar
Committee Member
Stadium Crowd Tool in Houdini with Key-Frameable Motion Controls

A Thesis Submitted to the Faculty of the Visual Effects Department
in Partial Fulfillment of the Requirements for the
Degree of Master of Fine Arts in Visual Effects
Savannah College of Art and Design

By

Mei-Chieh Chen

Savannah, Georgia
August 2013
# Table of Contents

List of Figures .......................................................................................................................... 1

Abstract ......................................................................................................................................... 2

1. Introduction ............................................................................................................................ 3

2. Background ............................................................................................................................ 5
   2.1 Crowd Scenes Created with Massive and Houdini ............................................................. 5
   2.2 Crowd Scenes Created with Houdini Alone ........................................................................ 6

3. Key Discussion ....................................................................................................................... 9
   3.1 Approaches for Importing Key-Frame Animation to Houdini ............................................. 9
      3.1.1 OBJ Sequence ............................................................................................................ 10
      3.1.2 FBX with Geometry Cache ....................................................................................... 11
      3.1.3 BVH Approach ....................................................................................................... 12
   3.2 Workflow for Importing Motion Capture Data to Houdini ................................................. 12
      3.2.1 File Formats from Vicon Blade ................................................................................ 13
      3.2.2 Advantage of Binding Skin in Houdini ..................................................................... 14
   3.3 Placement .......................................................................................................................... 14
   3.4 Render Time ...................................................................................................................... 15
      3.4.1 Factors Affecting Render Time ................................................................................... 15
      3.4.2 Internal File Formats for Houdini ............................................................................ 15
4. Methodology

4.1 Motion Capture Format Conversion

4.2 Binding Skin in Houdini

4.2.1 Geometry Capture Tool

4.2.2 Edit Capture Region Tool

4.3 Motion Manipulation

4.3.1 Assigning Motion to Different Body Parts with Delete CHOP

4.3.2 Passing Key-Framed Parameters with Fetch CHOP

4.3.3 Blending Motion with Math CHOP

4.3.4 Triggering Motion with Lookup CHOP

4.3.5 Scaling Motion with Trim CHOP

4.4 Randomization

4.4.1 Speed Randomization and Motion Offset with Shift and Stretch CHOP

4.4.2 Material Override

4.4.3 Creating Empty Seats and Offsetting Agent Point Positions

5. Conclusion

5.1 Final Render Stills

5.2 Conclusion

6. Bibliography

7. Appendix

7.1 Channel Names and Channel Numbers for BVH Skeleton

7.2 User Guide for the Tool
List of Figures

1.1 Applications for creating crowd simulation................................................................. 4
2.1 Crowd scene in *Invictus* (2009).................................................................................. 5
2.2 Crowd scene in *J. Edgar* (2011)................................................................................ 6
2.3 Crowd scene in *Superman Returns* (2006) ................................................................. 7
2.4 Crowd scene in *Angels and Demons* (2009) .............................................................. 7
2.5 Crowd scene in *Cloudy with a Chance of Meatballs* (2009) ....................................... 8
2.6 * Crowd Surfing commercial for Pepsi.......................................................................... 9
3.1 Approaches for importing key-frame animation to Houdini......................................... 10
3.2 Workflow for importing motion capture data to Houdini ............................................. 13
4.1 Motion capture file conversion in Command Line Tools ............................................. 17
4.2 Binding skin to a skeleton with the Geometry Capture Tool....................................... 18
4.3 Adjusting skin weight with Edit Capture Region Tool............................................... 19
4.4 Specifying a range for the channel numbers in a Delete CHOP ................................. 21
4.5 Assigning different motions to different body parts.................................................... 22
4.6 User interface for custom parameters......................................................................... 23
4.7 Setting proportion of a motion with a Math CHOP..................................................... 23
4.8 Scaling a motion with a multiplier ............................................................................. 25
4.9 Assigning random texture maps to the agents............................................................. 28
4.10 Creating empty seats by randomly deleting points..................................................... 29
4.11 Offseting point positions for the agents..................................................................... 29
5.1 CG agents populated in a procedural stadium............................................................ 31
5.2 Stadium crowd generated by the tool ......................................................................... 31
7.1 User interface of the stadium crowd tool for placement............................................. 35
7.2 Structure of the stadium stands generated by the tool................................................. 36
7.3 User interface of the stadium crowd tool for motion controls .................................... 37
Stadium Crowd Tool in Houdini with Key-Frameable Motion Controls

Mei-Chieh Chen

August 2013

Due to Houdini’s procedural power, Houdini has participated in the production pipeline for generating crowd simulation in films and commercials by visual effects studios. A crowd simulation tool developed within Houdini would enable artists to create stadium crowds using Houdini as the primary software. Custom parameters are built in the tool allowing artists to set keys along the timeline for controlling the actions of the stadium crowds. The placement function of the tool is designed for a quick generation of the stadium stands with agents positioned on top of the structure accordingly. With this tool, artists would be able to create stadium crowds with minimum amount of preparation time.
1. Introduction

Crowd scenes in visual effects films or computer animations are often completed with computer-generated crowds for better control of the behaviors. Some popular applications used to generate CG crowds are Massive, Golaem Crowd, Side Effects Houdini, and Softimage CrowdFX. Among these, Massive is the most widely used application for generating large-scaled crowd simulation in feature films by visual effects studios. However, for individual artists who wish to create crowd simulation, not only the functionality but also the pricing of the software is an important factor to be taken into account. At the time when this thesis was written, the license for Massive and Golaem Crowd costs $17,999 USD and $6,599 USD respectively (Figure 1.1), which potentially makes Houdini and Softimage CrowdFX more affordable options for individual artists. CrowdFX is a new feature introduced in Softimage 2013 and has not been used in crowd creation in films or commercials as frequently as Houdini. On the other hand, Houdini has been adopted to create crowd scenes in the industry by large visual effects companies such as Double Negative, Method Studios, Sony Pictures Imageworks, and Framestore. Thus, Houdini is considered an ideal candidate for individual artists to make crowd simulations.
However, the pipelines and Houdini toolsets for creating crowd simulation from production companies are mostly proprietary and cannot be accessed by individual artists. Creating crowd simulation from scratch in Houdini is also quite time-consuming for artists who wish to create crowds for immediate use. Therefore, the purpose of this thesis is to build an original tool that would enable a CG artist to rapidly set up and simulate stadium crowds in Houdini.

---

2. **Background**

2.1 **Crowd Scenes Created with Massive and Houdini**

Houdini, in conjunction with Massive, was utilized by Method Studios in Vancouver to create crowd simulation for two movies, *Invictus (2009)* and *J. Edgar (2011)*. In *Invictus*, Method Studios visual effects supervisor Geoffrey Hancock first directed the team to edit the motion capture data and to create a library of animation in Massive. Then this motion data was carried to Houdini for the next stage of the simulation. With this workflow, Method Studios successfully filled the entire virtual stadium with motion-captured digital crowds (Figure 2.1). Hancock stated that the procedural nature of Houdini’s workflow was really beneficial since the team was able to interchange agent’s appearances or animation on an individual basis.6

![Crowd scene created by Method Studios. Invictus. Directed by Roger Donaldson.](image)

*Figure 2.1  Crowd scene created by Method Studios. Invictus. Directed by Roger Donaldson.*


---

Method Studios followed the same workflow to simulate street crowds for the movie *J. Edgar* (Figure 2.2). Due to Houdini’s point-based system, Method Studios was able to make quick iterations and changes on an individual level by manipulating the attributes on the point clouds.\(^7\)


**Figure 2.2** Crowd scene created by Method Studios. *J. Edgar*. Directed by Clint Eastwood. Imagine Entertainment, 2011. DVD. Warner Home Video, 2012.

### 2.2 Crowd Scenes Created with Houdini Alone

More visual effects studios chose to generate crowd simulation solely within Houdini. The pivotal Metropolis Hospital shot in the movie *Superman Returns (2006)* was composited with more than 1000 digital people in the scene. The Orphanage, the company in charge of the shot, used Houdini to create a full CG crowd and integrated them with the live-action plate (Figure 2.3).\(^8\)

---


\(^8\) Bill Desowitz, *Superman Returns: The Passion of the VFX — Part 2*, Animation World Network,
For the crowd shots in the movie *Angels and Demons (2009)* (Figure 2.4), Double Negative developed a motion capture-based crowd solution within Houdini. The team ran the particles through a simulation that drove the application of the required motion-capture clips. Houdini CHOPS network then came into place to blend between multiple motion capture clips to apply animation to each particle.⁹
Sony Pictures Imageworks created the background characters in the animation *Cloudy with a Chance of Meatballs* (2009) by building their own crowd system inside Houdini as well. The system was made up of channel operators in Houdini managing lots of animation clips and a custom particle operator to steer characters through a scene given specific goals and animation cycles to choose from.10

The 'Crowd Surfing' commercial for Pepsi produced by Framestore involved thousands of digital people waving in a party (Figure 2.6). Framestore also built their own crowd system with Houdini which allowed shots to be laid out quickly using an instancing approach. The crowd system also made it possible to split off parts of the crowd that required specific behaviors.

---

adding more layers of sophistication.\textsuperscript{11}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{crowd_simulation.png}
\caption{Crowd simulation created by Framestore. Pepsi, “Crowd Surfing.” Directed by Steve Cope. Framestore, 2012.}
\end{figure}

3. Key Discussion

3.1 Approaches for Importing Key-Frame Animation to Houdini

OBJ sequences, FBX with vertex cache and BVH files are three approaches to import key-frame animation into Houdini from other 3D packages (Figure 3.1). Each requires a different workflow and hence offers different degrees of flexibility in term of motion manipulation. Among these, the BVH file allows Houdini to manipulate animation with the most efficiency while the ability of the other two approaches to randomize any attribute is more limited.


<table>
<thead>
<tr>
<th><strong>Approach</strong></th>
<th><strong>OBJ Sequence</strong></th>
<th><strong>FBX with Geometry Cache</strong></th>
<th><strong>BVH File</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Imported as</em></td>
<td>mesh</td>
<td>mesh</td>
<td>skeleton</td>
</tr>
<tr>
<td><strong>Bind skin in Houdini?</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Motion Data in Houdini</strong></td>
<td>No channels</td>
<td>Three channels controlling the XYZ positions of every vertex in the mesh.</td>
<td>Three channels controlling the RX, RY and RZ for every bone in the skeleton.</td>
</tr>
<tr>
<td><strong>Notes</strong></td>
<td>No motion data for further manipulation</td>
<td>-Large size of cache. -Heavy Computation for large number of agents.</td>
<td>- Flexible in controlling a particular part of the character. -Total number of channels per skeleton is small.</td>
</tr>
</tbody>
</table>

*Figure 3.1 Approaches for importing key-frame animation to Houdini*

**3.1.1 OBJ Sequences**

The OBJ file format is a universal geometry format and can be easily transported from one 3D application to another. To export an OBJ sequence from Maya, MEL scripting is used to automate the generation process. In Houdini, an OBJ sequence can be read and displayed by a File SOP. If the imported animation from the sequence is a motion cycle, a modulus operator is incorporated in an expression that loops the OBJ files and associates them with the current frame.

For this approach, the character is already skinned in Maya; thus skinning can be omitted in Houdini. Although it is a straight-forward way to import key-frame animation to Houdini, the
OBJ sequence does not contain any motion curves available for Houdini to make any further modification. For that reason, it is hard to create a natural-looking crowd through this method since all agents would be performing the same animation without any randomization. As a result, an OBJ sequence is not considered to be an ideal approach for animation import.

### 3.1.2 FBX with Geometry Cache

One way to import a skinned character with animation data editable in Houdini is through an FBX file with geometry caches. Key-frame animation made in Maya is baked, cached out and saved as an FBX file. The FBX file contains the mesh of the character, and the geometry cache is a sequence of vertex transformation data stored in a separate folder. Houdini reads in the mesh from the FBX file and re-creates the animation by deforming the mesh through moving the vertices of the mesh by every frame according to the transformation data stored in the cache.

The geometry cache is interpreted as channels in Houdini, and these motion curves can indeed be further manipulated with channel operators. However, the disadvantage of this approach is that the total number of the channels from the cache might result in considerably heavy computation for crowd production. The position of each vertex in the mesh is controlled by 3 channels each in x, y, z coordinates, which means the total number of channels is three.
times the total number of vertices in an object. For simulation containing thousands of agents, the computation would be too heavy for the system to handle. As a result, this approach is feasible only for crowd simulation with a small number of agents.

3.1.3 BVH Approach

BVH is a file format usually used to store motion capture data. Key-frame animation can also be brought into MotionBuilder and saved as a BVH file. Different from the previous two approaches, BVH files will only import a skeleton and the animation data to Houdini. In Houdini, the imported animation is interpreted as channels that control the RX, RY, RZ of each bone in the BVH skeleton with a total of 30 bones. The software automatically connects the animation with the skeleton. Since the motion of each bone is controlled by three channels, the total number of channels controlling the whole skeleton is only around a hundred. With this approach, the software is able to handle the motion for thousands of agents efficiently.

3.2 Workflow for Importing Motion Capture Data to Houdini

Under most circumstances, it is necessary to capture numerous takes of a single motion in order to create a natural-looking crowd simulation. The time to process and edit all takes of all motions might be fairly lengthy. Therefore, establishing a workflow with less effort for
processing motion clips would be quite helpful in crowd simulation.

### 3.2.1 File Formats from Vicon Blade

Data captured in Vicon Blade can be exported in different formats. Many of these formats could be edited, converted and brought into Houdini with the help of MotionBuilder and the Command Line Tool (Figure 3.2). C3D, FBX and BVH are three common formats that are used to import motion capture data into Houdini with different workflows.

![Figure 3.2 Workflow for importing motion capture data to Houdini](image-url)
Since a C3D file only contains the data of the marker positions captured by the cameras in 3D space, certain data processing has to be done before a control rig is created and successfully moves with the animation. On the other hand, FBX and BVH files store the motion data as an animated skeleton and thus require less processing efforts before the final motion is imported to Houdini. If the motion clip does not need any editing in MotionBuilder, BVH files involve even less processing steps than FBX files because BVH files can be directly converted to a CMD and a BCLIP, native formats supported by Houdini. In brief, BVH is the preferred format used to import motion capture data for this thesis.

### 3.2.2 Advantage of Binding Skin in Houdini

After the animated skeleton from motion capture is imported to Houdini, binding the mesh of the character to the skeleton is taking place in Houdini itself. In fact, binding skin in Houdini has its advantages. Since the capture geometry tool in Houdini procedurally marks the affecting region for a particular bone with a handle, users can always go back and resize the handle to accommodate any changes. In addition, as long as the input meshes are enclosed within the marked region, it is possible to have multiple meshes bind to the same skeleton. For instance, with a T-shirt mesh and a sweatshirt mesh aligned in proximity, these two meshes can both bind to the skeleton. With stamping, one can randomly assign different meshes to individual agents.
3.3 Placement

Although copying and instancing can both be used to populate agents onto the stadium, manipulating motions can only be achieved by copying alone since instancing is especially designed for outputting identical objects and only works at object level. In other words, the instancepoint() function can only vary object-level parameters and cannot be passed to the CHOP network level. Therefore, it is not possible to manipulate motions on an individual basis with instancing. Copy stamping, on the other hand, is fairly flexible in accommodating the needs of randomization for crowd simulation. Stamping enables the user to modify the source geometry for each copy and is capable of communicating with the channel operators to vary the parameters controlling the motions.

3.4 Render Time

3.4.1 Factors Affecting Render Time

The main factors affecting render time are the total number of agents present in a scene, the complexity of the CHOP networks with the motion data, and the degree of randomization for the simulation. The render time for 5000 agents is around 10 minutes for a single motion and 50 minutes on average for a complex CHOP network involving scaling, triggering and blending multiple actions. In the visual project of this thesis, lighting and rendering did not seem to
contribute to a significant amount of the render time. In addition, the PBR renderer and
environment light handles the crowd simulation well without any artifacts.

3.4.2 Internal File Formats for Houdini

Houdini supports a variety of image formats and geometry formats. These formats are
considered external formats to Houdini and will not be directly read by the software. Instead,
they will be converted to Houdini’s internal formats for displaying or rendering. Internal formats
are native to Houdini and do not require any conversion. This means that directly applying
Houdini’s internal formats for texture mapping and for agent mesh will save the system from
converting the files repeatedly and thus reduce some render time. The RAT format is Houdini’s
internal image file format specifically for texture mapping. The BGEO file format is the binary
format for storing Houdini geometry.\textsuperscript{12} In the visual project of this thesis, all image files and
geometry files were converted to RAT and BGEO before being applied to the system.

4. Methodology

4.1 Motion Capture Format Conversion

Biovision (.bvh), Acclaim (.asf and .amc), and Motion Analysis (.trc) are three popular
(accessed April 30, 2013)
formats that describe animated skeletons from motion capture data. With the help of Houdini’s Command Line Tools, these file formats can be converted into .cmd and .bclip files readable by Houdini. A .bclip file is a Houdini channel file that stores the motion data which will later be interpreted by a channel operator. By sourcing the .cmd file in Houdini, the software will automatically create a skeleton and a File CHOP linked to the .bclip file.

The conversion of these three formats into .cmd and bclip files requires running a short script through the Command Line Tool (Figure 4.1). The command for Biovision, Acclaim and Motion Analysis are mcbiovision, mcacclaim and mcmotanal respectively. The most basic script without any flags first starts with the command followed by the input file name with the original extension.

---

4.2  **Bind Skin in Houdini**

4.2.1  **Geometry Capture Tool**

In order for a character to move with a skeleton in Houdini, it is necessary to bind the character’s skin to the bones with the Capture Geometry Tool. Before the tool is applied, the skeleton and the mesh should first overlap in 3D space. This can be accomplished with the skeleton enclosed in a subnetwork. With the help of a subnetwork, all the bones can be translated and scaled as a whole to match the position and the size of the mesh. Once the skeleton is properly aligned with the mesh, the Capture Geometry Tool is ready to be applied from the character tool shelf (Figure 4.2).

*Figure 4.2  Binding skin to a skeleton with the Geometry Capture Tool* 

---

4.2.2 Edit Capture Region Tool

The next step of skinning is to modify the region of affected mesh corresponding to a particular bone. Houdini’s Edit Capture Region Tool provides a quick way to set up the overall skin weight of a skeleton. The Edit Capture Region Tool is a capsule-like handle with two hemispheres at each end. Users can adjust the translation and the rotation of the handle to better position the tool in the influenced region. Increasing the radii of the hemispheres will allow greater region of the mesh being captured by the bone.

![Adjusting skin weight with Edit Capture Region Tool](image)

To use the Edit Capture Region Tool, it is easier to start with the head, the legs, and the arms before moving to the spine, the shoulders, and the clavicle. Since the spine, the shoulders,

---

and the clavicle are the parts in a skeleton that are connected to multiple bones, the editing process would be more complicated and should be done in the last step. The Edit Capture Region Tool sets up the coarse distribution of the skin weight for each bone. For finer control of the skin weight, the Paint Capture Layer tool is available to interactively change the skin weights by painting the mesh with a brush.

4.3 Motion Manipulation

4.3.1 Assigning Motion to Different Body Parts with Delete CHOP

An imported skeleton consists of a bone network and a null object. The null object is the hips, also the root of the skeleton. Only the root of the whole skeleton contains translation data while the rest of the bone objects contain only rotation data. The rotation of each bone in x, y, z direction is driven by a channel stored in the .bclip file.

A channel can be identified either by its channel name or its channel number. The channel name labels which bone this channel is associated with. The channel number is a non-repetitious integer that is assigned to each of the channels throughout the whole skeleton. Please refer to the appendix for a full listing of the channel names and channel numbers for a BVH skeleton.
By utilizing the channel number and Delete CHOP, it is possible to assign different motions to different body parts. A Delete CHOP is used to select channels from the input node keeping only the selected ones affecting the final motion (Figure 4.4). Since each channel represents the rotation of a bone, selecting a range of channel numbers means selecting the motion data for a particular part of the skeleton. For instance, one can have the upper body of a character performing one motion by selecting a range of channel numbers targeting the upper body with one Delete CHOP and have the lower body performing another motion by doing the same thing with another Delete CHOP; then combine them with a Merge CHOP (Figure 4.5).

---

4.3.2 Passing Key-Framed Parameters with Fetch CHOP

In Houdini, custom parameters can be created and displayed in a user interface (Figure 4.6).

With the user interface, it is easy for the users to set keys on the parameters. After being key-framed, the parameters will form a channel editable in the channel editor. The channels from the user input can be brought in with a Fetch CHOP and integrated with the CHOP network to control the final motions being assigned to the agents.\footnote{Side Effects, \textit{Fetch CHOP node}, Side Effects, http://www.sidefx.com/docs/houdini12.0/nodes/chop/fetch (accessed April 30, 2013)}
4.3.3 Blending Motion with Math CHOP

A Math CHOP is a channel operator that allows the users to perform arithmetic operations on channels. To blend two motion clips, the Math CHOP is first used to multiply all the channels from the two motion clips by a fraction (Figure 4.7). The fraction represents the proportion of the motion that will be participating in the resulting animation. In order to have the blended animation working properly, it is important to make sure that the two fractions add up to one in total. The fractionalized channels from the two clips are added together according to their channel names by another Math CHOP.

---

4.3.4 Triggering Motion with Lookup CHOP

Motion such as sitting to standing or the reverse does not form a cycle and requires a trigger to initiate the action. To set a trigger for a specific motion, a custom parameter can be created at the top level and be key-framed to form a channel for controlling the start and the end of the motion.

Once the attribute is created, a Lookup CHOP takes place to do the heavy lifting. A Lookup CHOP has two inputs. The first input takes the custom attribute as an index of how the animation will be mapped in time. The second input is the animation itself which will be fit into the time range. After the Lookup CHOP takes in the two inputs, it will force the animation to start when the attribute is greater than 0 and to end when the attribute gets to 1.19

4.3.5 Scaling Motion with Trim CHOP

For a motion with a distinct pattern such as clapping, it is possible to magnify or diminish the motion with the help of a Trim CHOP and several Math CHOPs (Figure 4.8). The first step is to determine the baseline channel values for a motion clip. For example, the baseline channel values for clapping are the channel values at a particular frame when the palms of the character

---

strike together. Once the frame number is found, a Trim CHOP takes the frame number,\textsuperscript{20}
extracts the baseline channel values at this frame, and keeps these values constant throughout the entire length.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Scaling_a_motion_with_a_multiplier.png}
\caption{Scaling a motion with a multiplier}
\end{figure}

The difference of the channel values between the original motion clip and the output of the Trim CHOP is calculated and multiplied with a desired number. With the multiplier greater than one, the resulting motion is magnified from the original, whereas with the multiplier less than one, the resulting motion is diminished.

4.4 Randomization

In a real stadium, each person in the crowd behaves differently. Hence, randomization plays an important role in preventing regular patterns being detected in a CG crowd. Stamping is a feature of the Copy SOP which allows each copy to be modified according to its copy number or point number. In combination with the random expression, stamping is used repeatedly to vary the size, texture, geometry, speed and motion offset of the agents on an individual basis.

4.4.1 Speed Randomization and Motion Offset with Shift and Stretch CHOP

If a motion clip is not captured long enough, regular patterns among agents will be easily discovered by the viewers. For this reason, all motion data used in this thesis were recorded more than a thousand frames in length. To further randomize the speed of a motion and to offset a motion cycle, a Stretch CHOP and a Shift CHOP are used. Since this involves modifying the length of the channel or shifting the entire channel, it is necessary to ensure all File CHOPs are repeated infinitely in a cycle.

A Shift CHOP offsets the reference position of a cycle. Since each agent is copied onto a different point, the point number can be used to serve as the random seed. The rand() function in

Houdini only returns a number from 0 to 1; therefore, it is necessary to incorporate the channel length in the expression in order to shift the motion along the entire clip. Based on the sample rate, the \texttt{icl()} function returns the length of a CHOP’s input. Multiplying the \texttt{icl()} function with the \texttt{rand()} function gives the appropriate values for the Scroll Offset.

Changing the speed of a motion can be achieved by stretching or compressing the channels with a \texttt{Stretch CHOP}.\textsuperscript{22} Stretching the channels will slow down the motion while compressing them will speed up the motion. The Length Scale is the targeted parameter being randomized. The expression of randomizing the Length Scale for the \texttt{Stretch CHOP} is very similar to that for the \texttt{Shift CHOP} with a \texttt{icl()} and a \texttt{rand()} function.

\subsection*{4.4.2 Material Override}

Houdini offers the users an option to override individual parameters on an assigned material by creating \texttt{material\_override} attributes on points. Setting up an override parameter means that the parameter values that were originally defined on the material itself will be replaced by the overridden ones. Thus, it is possible to assign random texture maps to each agent with its point number being the random seed (Figure 4.9). To assign different texture maps to the

agents, the parameter being overridden is the baseColorMap. A random function is applied to
direct the software to fetch the path of a texture map to each agent at random.

![Image](image.jpg)

*Figure 4.9 Assigning random texture maps to agents*

### 4.4.3 Creating Empty Seats and Offsetting Agent Point Positions

In reality, it is not possible to have the stadium fully occupied by people for every game. Therefore, vacancy is also taken into account by creating a custom attribute to further randomize the placement. Since agents are copied onto points, removing some points will remove some agents as well (Figure 4.10). This can be achieved with a Delete SOP through an expression. For example, the expression for 30% of the seats to be empty might be written as rand($PT) < 0.3.
In addition, not everyone in a stadium sits right on the center of a seat. To offset each agent slightly from the center, stamping is applied. With the point number being the random seed of a random function, agents will be translated away from the center by different amounts (Figure 4.11). In order to avoid the agents penetrating each other, the distance between two agents is calculated, and only a fraction of this value is used to offset the position of the agents.
5. Conclusion

5.1 Final Render Stills

Figure 5.1  CG agents populated in a procedural stadium.

Figure 5.2  Stadium crowd generated by the tool. There are 10000 agents present in this scene.
5.2 Conclusion

Standing still, sitting still, looking around, clapping and cheering, common actions for stadium crowds, are integrated into a single CHOP network which can be easily controlled with the tool user interface. Without having any prior motion capture or rigging knowledge, users can still produce stadium crowds easily with this tool in Houdini. In addition, this tool allows users to set keys to the actions at a specific time with smooth transition from one action to another. The placement feature of this tool enables the users to create stadium stands with the desired shape and to populate agents onto them with the capability of adjusting the density of the agents.

With the stadium crowd tool completely developed within Houdini, the software is indeed proven to be a friendly environment for crowd simulation. Houdini supports motion capture data to be imported into the software with an automatic connection of the channels to the skeleton. The channel operators make motion manipulation and randomization possible and flexible. Houdini digital assets and user interface also enables this tool to be used with ease. This thesis explores several approaches for creating crowd simulation in Houdini and comes to a conclusion that the BVH approach is the preferable one for both motion capture and key-frame animations. The techniques and workflow documented in this thesis might serve as a reference for artists who wish to create crowds from scratch or a tool of similar form.
6. Bibliography


Side Effects. *Character Motion Capture*. Side Effects.
   http://www.sidefx.com/docs/houdini11.1/character/motioncapture/

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/delete

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/fetch

   http://www.sidefx.com/docs/houdini12.0/io/formats/

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/lookup

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/math

   http://www.sidefx.com/docs/houdini12.0/shelf/capturegeometry/

   http://www.sidefx.com/docs/houdini12.0/shelf/editcaptureregions

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/shift

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/stretch

   http://www.sidefx.com/docs/houdini12.0/nodes/chop/lookup

Softimage. *Free Trial*. Autodesk.

7. **Appendix**

7.1 **Channel Names and Channel Numbers for BVH Skeleton**

<table>
<thead>
<tr>
<th>Channel Name</th>
<th>Channel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hips:tx</td>
<td>0</td>
</tr>
<tr>
<td>Hips:ty</td>
<td>1</td>
</tr>
<tr>
<td>Hips:tz</td>
<td>2</td>
</tr>
<tr>
<td>Hips_To_LeftUpLeg:rx</td>
<td>3</td>
</tr>
<tr>
<td>Hips_To_LeftUpLeg:ry</td>
<td>4</td>
</tr>
<tr>
<td>Hips_To_LeftUpLeg:rz</td>
<td>5</td>
</tr>
<tr>
<td>LeftUpLeg_To_LeftLeg:rx</td>
<td>6</td>
</tr>
<tr>
<td>LeftUpLeg_To_LeftLeg:ry</td>
<td>7</td>
</tr>
<tr>
<td>LeftUpLeg_To_LeftLeg:rz</td>
<td>8</td>
</tr>
<tr>
<td>LeftLeg_To_LeftFoot:rx</td>
<td>9</td>
</tr>
<tr>
<td>LeftLeg_To_LeftFoot:ry</td>
<td>10</td>
</tr>
<tr>
<td>LeftLeg_To_LeftFoot:rz</td>
<td>11</td>
</tr>
<tr>
<td>LeftFoot_To_LeftToeBase:rx</td>
<td>12</td>
</tr>
<tr>
<td>LeftFoot_To_LeftToeBase:ry</td>
<td>13</td>
</tr>
<tr>
<td>LeftFoot_To_LeftToeBase:rz</td>
<td>14</td>
</tr>
<tr>
<td>LeftToeBase_To_LeftToeBaseEnd:rx</td>
<td>15</td>
</tr>
<tr>
<td>LeftToeBase_To_LeftToeBaseEnd:ry</td>
<td>16</td>
</tr>
<tr>
<td>LeftToeBase_To_LeftToeBaseEnd:rz</td>
<td>17</td>
</tr>
<tr>
<td>Hips_To_RightUpLeg:rx</td>
<td>18</td>
</tr>
<tr>
<td>Hips_To_RightUpLeg:ry</td>
<td>19</td>
</tr>
<tr>
<td>Hips_To_RightUpLeg:rz</td>
<td>20</td>
</tr>
<tr>
<td>RightUpLeg_To_RightLeg:rx</td>
<td>21</td>
</tr>
<tr>
<td>RightUpLeg_To_RightLeg:ry</td>
<td>22</td>
</tr>
<tr>
<td>RightUpLeg_To_RightLeg:rz</td>
<td>23</td>
</tr>
<tr>
<td>RightLeg_To_RightFoot:rx</td>
<td>24</td>
</tr>
<tr>
<td>RightLeg_To_RightFoot:ry</td>
<td>25</td>
</tr>
<tr>
<td>RightLeg_To_RightFoot:rz</td>
<td>26</td>
</tr>
<tr>
<td>RightFoot_To_RightToeBase:rx</td>
<td>27</td>
</tr>
<tr>
<td>RightFoot_To_RightToeBase:ry</td>
<td>28</td>
</tr>
<tr>
<td>RightFoot_To_RightToeBase:rz</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel Name</th>
<th>Channel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>RightToeBase_To_RightToeBaseEnd:rx</td>
</tr>
<tr>
<td>31</td>
<td>RightToeBase_To_RightToeBaseEnd:ry</td>
</tr>
<tr>
<td>32</td>
<td>RightToeBase_To_RightToeBaseEnd:rz</td>
</tr>
<tr>
<td>33</td>
<td>Hips_To_Spine:rx</td>
</tr>
<tr>
<td>34</td>
<td>Hips_To_Spine:ry</td>
</tr>
<tr>
<td>35</td>
<td>Hips_To_Spine:rz</td>
</tr>
<tr>
<td>36</td>
<td>Spine_To_Spine1:rx</td>
</tr>
<tr>
<td>37</td>
<td>Spine_To_Spine1:ry</td>
</tr>
<tr>
<td>38</td>
<td>Spine_To_Spine1:rz</td>
</tr>
<tr>
<td>39</td>
<td>Spinel1_To_Neck:rx</td>
</tr>
<tr>
<td>40</td>
<td>Spinel1_To_Neck:ry</td>
</tr>
<tr>
<td>41</td>
<td>Spinel1_To_Neck:rz</td>
</tr>
<tr>
<td>42</td>
<td>Neck_To_Head:rx</td>
</tr>
<tr>
<td>43</td>
<td>Neck_To_Head:ry</td>
</tr>
<tr>
<td>44</td>
<td>Neck_To_Head:rz</td>
</tr>
<tr>
<td>45</td>
<td>Head_To_HeadEnd:rx</td>
</tr>
<tr>
<td>46</td>
<td>Head_To_HeadEnd:ry</td>
</tr>
<tr>
<td>47</td>
<td>Head_To_HeadEnd:rz</td>
</tr>
<tr>
<td>48</td>
<td>Spinel1_To_LeftShoulder:rx</td>
</tr>
<tr>
<td>49</td>
<td>Spinel1_To_LeftShoulder:ry</td>
</tr>
<tr>
<td>50</td>
<td>Spinel1_To_LeftShoulder:rz</td>
</tr>
<tr>
<td>51</td>
<td>LeftShoulder_To_LeftArm:rx</td>
</tr>
<tr>
<td>52</td>
<td>LeftShoulder_To_LeftArm:ry</td>
</tr>
<tr>
<td>53</td>
<td>LeftShoulder_To_LeftArm:rz</td>
</tr>
<tr>
<td>54</td>
<td>LeftArm_To_LeftForeArm:rx</td>
</tr>
<tr>
<td>55</td>
<td>LeftArm_To_LeftForeArm:ry</td>
</tr>
<tr>
<td>56</td>
<td>LeftArm_To_LeftForeArm:rz</td>
</tr>
<tr>
<td>57</td>
<td>LeftForeArm_To_LeftHand:rx</td>
</tr>
<tr>
<td>58</td>
<td>LeftForeArm_To_LeftHand:ry</td>
</tr>
<tr>
<td>59</td>
<td>LeftForeArm_To_LeftHand:rz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel Name</th>
<th>Channel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>LeftHand_To_LeftHandThumb:rx</td>
</tr>
<tr>
<td>61</td>
<td>LeftHand_To_LeftHandThumb:ry</td>
</tr>
<tr>
<td>62</td>
<td>LeftHand_To_LeftHandThumb:rz</td>
</tr>
<tr>
<td>63</td>
<td>LeftHandThumb_To_LeftHandThumbEnd:rx</td>
</tr>
<tr>
<td>64</td>
<td>LeftHandThumb_To_LeftHandThumbEnd:ry</td>
</tr>
<tr>
<td>65</td>
<td>LeftHandThumb_To_LeftHandThumbEnd:rz</td>
</tr>
<tr>
<td>66</td>
<td>LeftHand_To_L_Wrist_End:rx</td>
</tr>
<tr>
<td>67</td>
<td>LeftHand_To_L_Wrist_End:ry</td>
</tr>
<tr>
<td>68</td>
<td>LeftHand_To_L_Wrist_End:rz</td>
</tr>
<tr>
<td>69</td>
<td>Spinel1_To_RightShoulder:rx</td>
</tr>
<tr>
<td>70</td>
<td>Spinel1_To_RightShoulder:ry</td>
</tr>
<tr>
<td>71</td>
<td>Spinel1_To_RightShoulder:rz</td>
</tr>
<tr>
<td>72</td>
<td>RightShoulder_To_RightArm:rx</td>
</tr>
<tr>
<td>73</td>
<td>RightShoulder_To_RightArm:ry</td>
</tr>
<tr>
<td>74</td>
<td>RightShoulder_To_RightArm:rz</td>
</tr>
<tr>
<td>75</td>
<td>RightArm_To_RightForeArm:rx</td>
</tr>
<tr>
<td>76</td>
<td>RightArm_To_RightForeArm:ry</td>
</tr>
<tr>
<td>77</td>
<td>RightArm_To_RightForeArm:rz</td>
</tr>
<tr>
<td>78</td>
<td>RightForeArm_To_RightHand:rx</td>
</tr>
<tr>
<td>79</td>
<td>RightForeArm_To_RightHand:ry</td>
</tr>
<tr>
<td>80</td>
<td>RightForeArm_To_RightHand:rz</td>
</tr>
<tr>
<td>81</td>
<td>RightHand_To_RightHandThumb:rx</td>
</tr>
<tr>
<td>82</td>
<td>RightHand_To_RightHandThumb:ry</td>
</tr>
<tr>
<td>83</td>
<td>RightHand_To_RightHandThumb:rz</td>
</tr>
<tr>
<td>84</td>
<td>RightHandThumb_To_RightHandThumbEnd:rx</td>
</tr>
<tr>
<td>85</td>
<td>RightHandThumb_To_RightHandThumbEnd:ry</td>
</tr>
<tr>
<td>86</td>
<td>RightHandThumb_To_RightHandThumbEnd:rz</td>
</tr>
<tr>
<td>87</td>
<td>RightHand_To_R_Wrist_End:rx</td>
</tr>
<tr>
<td>88</td>
<td>RightHand_To_R_Wrist_End:ry</td>
</tr>
<tr>
<td>89</td>
<td>RightHand_To_R_Wrist_End:rz</td>
</tr>
</tbody>
</table>
7.2 User Guide for the Tool

The zip file in the DVD contains the digital asset for the tool and all the necessary texture files, geometry files, and motion files. To install the tool, please refer to the following steps:

1. Extract the content from stadium_agent.zip to the working directory.
2. In Houdini, go to File > Install Digital Library. Select $HIP/stadium_agent.otl for the digital asset library.
3. Hit the Tab key to bring up the menu. Click Stadium Crowd under Digital Assets.
4. Adjust the parameters for placement and motion controls. The detailed description of each parameter is listed below.

![User interface of the stadium crowd tool for placement.](image)

*Figure 7.1  User interface of the stadium crowd tool for placement.*
<table>
<thead>
<tr>
<th>Parameters for Placement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent Scale</td>
<td>The uniform size of the agents.</td>
</tr>
<tr>
<td>Agent Position Offset</td>
<td>The distance away from the center of the seats that the agents will be positioned.</td>
</tr>
<tr>
<td>Upper Ring Translate</td>
<td>The translation for the upper boundary of the stands.</td>
</tr>
<tr>
<td>Upper Ring Scale</td>
<td>The scale of the upper boundary of the stands.</td>
</tr>
<tr>
<td>Lower Ring Translate</td>
<td>The translation for the lower boundary of the stands.</td>
</tr>
<tr>
<td>Lower Ring Scale</td>
<td>The scale of the lower boundary of the stands.</td>
</tr>
<tr>
<td>Arc Angle</td>
<td>The angles of the two rays in degrees that comprise the arc. Arc Angle from 0 to 360 forms a circle.</td>
</tr>
<tr>
<td>Number of Columns</td>
<td>The total number of columns in the stadium.</td>
</tr>
<tr>
<td>Number of Steps</td>
<td>The total number of stairs in a column.</td>
</tr>
<tr>
<td>Rate of Empty Seats</td>
<td>The probability of a seat not occupied by an agent.</td>
</tr>
</tbody>
</table>

*Figure 7.2 Structure of the stadium stands generated by the tool.*
Figure 7.3  User interface of the stadium crowd tool for motion controls
<table>
<thead>
<tr>
<th>Parameters for Motion Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Motion to be performed by the agents: Still, Clapping or Cheering.</td>
</tr>
<tr>
<td>Stand/Sit</td>
<td>Options for the agents to stand or sit.</td>
</tr>
<tr>
<td>Motion Scale</td>
<td>The strength of a motion. A scale value greater than 1.0 means the motion will be magnified. A scale value less than 1.0 means the motion will be diminished.</td>
</tr>
<tr>
<td>Stretch Range</td>
<td>The range of the time stretch for a motion. A stretch value of 0.5 means the channels are compressed by half; the speed doubles. A stretch value of 1.5 means that the channels are stretched twice as long; the speed is cut in half.</td>
</tr>
<tr>
<td>Frame Shift</td>
<td>Shift of channels in frames along the timeline. For a greater value of frame shift, the agents will perform sit to stand more randomly with a greater time gap.</td>
</tr>
<tr>
<td>Action Transition</td>
<td>The transition time for one action to move onto another.</td>
</tr>
<tr>
<td>Stand/Sit Transition</td>
<td>The transition time before “sit to stand” starts from the previous action.</td>
</tr>
<tr>
<td>Overall Shift</td>
<td>The final motion offset along the timeline for randomization.</td>
</tr>
</tbody>
</table>