TAMPERE UNIVERSITY
OF APPLIED SCIENCES

Fire Simulation for a 3D Character with Particles and Motion Capture Data in Blender

Ville Hoikkala

Bachelor's thesis
May 2016
Degree Programme in Media

ABSTRACT

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

HOIKKALA, VILLE:
Fire Simulation for a 3D Character with Particles and Motion Capture Data in Blender

Bachelor's thesis 41 pages
May 2016

The object of this thesis was to study particle based simulations in 3D and how they can be applied to motion capture data. The theoretical part examines the history of 3D and computer graphics concisely and finally particles and simulations.

The practical part consists of the case-work involving a fire emitting particle simulation. The study examines two methods of creating a fire simulation in Blender and how they influence the outlook of the particles.

It was found that the fire simulation can be achieved and applied to motion capture data. The study suggests that creating a particle simulation to motion capture data is only the beginning in the visual effects workflow. Completing a visual effects work involves multiple work stages.

On the basis of the thesis, it is possible to comprehend the fire simulation workflow and to apply it to motion capture data in Blender.

Key words: Blender, simulation, motion capture data.

CONTENTS

GLOSSARY

CGI	Computer-generated imagery, CGI for short is creating pictures or movies using computers.
Blender	3D computer graphics software.
Software	Collection of instructions or data that the user is able to interact with a computer.
Mocap	Short from motion capture, technology of recording the movement of objects or people.
Emitter	The entity that creates the particles.
Smoke Domain	The containment of a simulation.
Particle system	A group of particles, where the particles are treated as one entity.
3D Rendering	Computing final images out of mathematical data.
Mesh	Collection of vertices and edges that define the shape of a 3D model.
Volume	Volume represents the light interaction inside the mesh.
Shader	Shaders define the light interaction at the surface of the mesh.
Voxel	A unit of graphic information that defines a point in 3D-space.
Armature	Armatures mimic skeletons and are made of bones.

Bones	The base element of an armature. They have three points, the root, the body of the bone itself and the tip.
Inverse Kinematics	Form of animation process allowing to position the last bone in the bone chain and the other bones are positioned automatically.
Forward Kinematics	Manual positioning of each bone.
Cache	Storing simulation data to disk to speed up processing times.
Baking	Pre-calculating the simulation.
CPU	Central processing unit.
GPU	Graphics processing unit.

1 INTRODUCTION

For CGI artists and animators, motion capture has become one of the most important tools of use (Liverman 2004, 13). Motion capture has been a well-used tool for creating fascinating creatures and worlds for the cinema, gaming and virtual reality for a while now. But in my opinion motion capture hasn't been used to its full extent in sports production, and my goal in this thesis is to find out how viable motion capture is in the subject. I believe with this study, I will successfully conclude the potentials and limitations in the field and how they can be utilised. I will be covering a practical case which I have worked on.

Why do I want to focus on motion capture in sports production? For the last two seasons, I have been creating player introductions for the hockey team Tappara. Introductions are 4-5 second clips of the players which are shown at the start of the game in the hockey arena's large-screen televisions, also known as a jumbotron. The player introduction is also replayed for the individual player who scores during the ice hockey match. While working on these two seasons I have mainly been doing 2D motion graphics. I want to see how well motion capture can be used for motion graphics. The object is to further develop my skills with motion capture data.

The thesis is aimed at all who have interest in 3D and the technicalities involved in creating particle systems with a 3D software. Basic knowledge of Blender is expected, so the main focus is on the fire simulation and motion capture data.

2 DEFINING 3D

In order to understand particles and simulations, let us first define what 3D is and how it came to be. Computer graphics are still a fairly young art technique. But its influence in industry, societies and our daily lives has been significant.

2.1 3D graphics and applications

3D-graphics has become a standard for multiple types of design applications. Artificial models offer invaluable information in building-, industrial- and city planning. For example, it is much easier to examine a buildings model in 3D-space to see what works and what possibly doesn't before the time and money consuming projects begin. 3D enables users to visualize their plans before actual work. 3D-visualization has also become an important standard in healthcare. Being able to produce 3D scans of the human anatomy is an important way to demonstrate medical data in a way that is easier to comprehend. (Puhakka 2008, 24.)

Simulations are used for practising the operation of different kinds of machinery. For example, airplanes, helicopters, ships, trains and forest machinery. Simulators allow to practice in a safe environment and help to prepare for real situations. (Puhakka 2008, 24.) Virtual reality will utilize simulations and 3D-visualization even further. Soon the technology will be made available to the common user and time will show how much popularity it will gain in consumer applications.

Obviously one of the most noticeable uses of 3D can be seen in movies and television. Entertainment industry has gained a lot from 3D and the technology has progressed so far that sometimes it is hard to separate reality from a computer generated image. Other than movies and television, one of the biggest appliances behind advancing 3D is the gaming industry. Globally, gaming has become a larger business than the movie industry (Puhakka 2008, 24.). Due to the high demand, gaming industry is one of the central forces behind the advancement of 3D-graphics and standards.

2.2 Computer graphics history

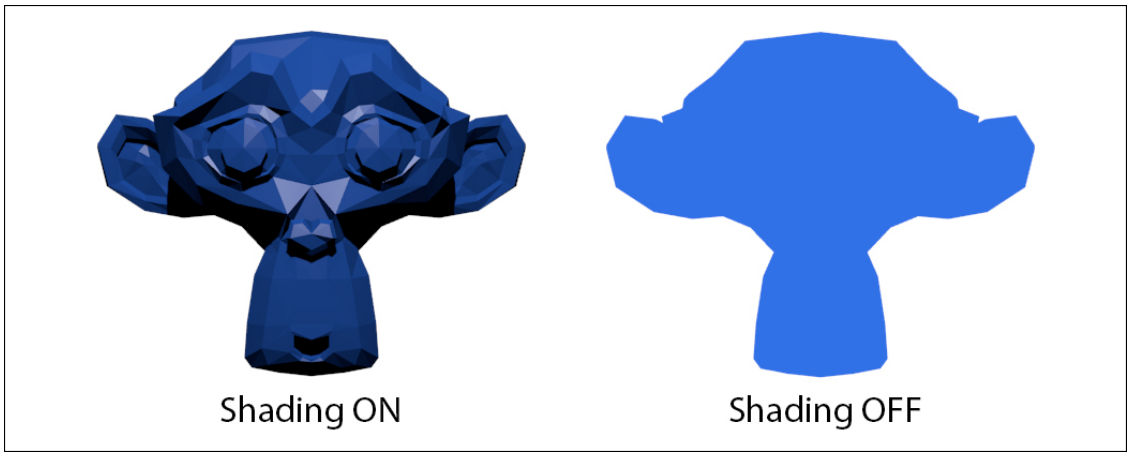
The start of computer graphics could be placed at the Cold War. The United States engineered a computer that could be used to operate a flight simulator for project Whirlwind. Based on this, project SAGE continued on the work and created an air traffic control system. The information received from the radars were drawn as vector graphics on the computer screen, and the system was controlled with a light pen. (Puhakka 2008, 24.)

The first time computer graphics were used in a film was in 1958 by Alfred Hitchcock in the movie Vertigo. The opening of the film displayed abstract vector drawn triangles while music played. During this period, at the end of 1950 and the start of 1960's, IBM created their first commercial CAD-system DAC-1. (Puhakka 2008, 24.) Considered to be the first real-time graphical videogame, Spacewar was programmed in 1961. It consisted of simple white-line drawings, but took several hundreds of hours to program. (Chopine, 2011, 1.)

The foundation of modern graphical user interfaces happened in 1963, when Ivan Sutherland created Sketchpad as part of his doctoral thesis for a computer system called TX-2. Sketchpad allowed the user to draw shapes on the screen using a light pen. Before this, a user could only operate a computer by punch card readers which had no displays. (Chopine, 2011, 1.)

PICTURE 1. Sutherland with Sketchpad on the TX-2 (MIT Museum, 1963)

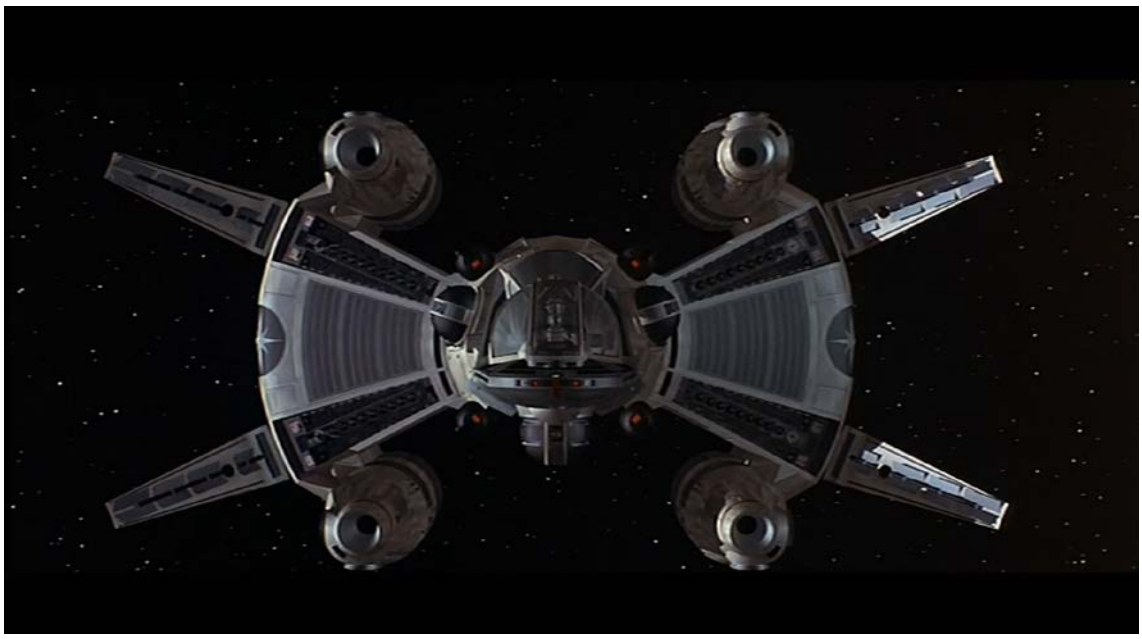
Sketchpad's 3D polygons were basically white wireframes that you could see through. It was not until Sutherland was hired for a program in Utah where researchers created an algorithm that would give the wireframes a solid appearance. Natural progression for solid appearances were shaders, which shade the colours on the objects surface. This was a big step but far away from the detail we have today. The forerunner towards more natural lighting was Bui Tuong Phong, who discovered that direct lighting on the objects caused more natural lighting. Phong created his own algorithm for this as part of his doctorate. Phong's shading created great results but took a long time to render. Jim Blinn continued Phong's work to create a quicker render. Both shaders are still widely used in today's 3D applications. (Chopine, 2011, 5.)



PICTURE 2. Same model with shading on and off (Hoikkala 2016)

Another pioneer in the world of 3D and especially animation is Ed Catmull, also a graduate from Utah and whom you might know today as the co-founder of Pixar Animation Studios. Catmull was working in Computer Graphics Laboratory (CGL) when he was approached by George Lucas, the man behind *Star Wars*. Lucas recruited Catmull and several other CGL employees to Lucas's own special effects division called Industrial Light and Magic (ILM). This is where they further advanced animation with features such as particles and motion blur. It was at ILM where computer-generated animations took the first steps towards becoming a part of the movie industry. But after the movie *Tron* failed at the box office Lucas scrapped ILM's computer-graphics division. Due to the poor success of *Tron*, most movie producers got scared of using computer-generated graphics and it wasn't until *The Last Starfighter* in 1984 when the interest rose again. *The Last Starfighter* used fully rendered spaceships as seen in the picture below and ended up saving money in production when all models were 3D compared to using practical effects.

Due to the cost-efficiency and the success of the 3D modelling, it revitalized filmmakers interest in 3D and computer graphics. (Chopine, 2011, 7.)



PICTURE 3. Spaceship scene from The Last Starfighter (1984).

3D graphics were on the rise and the technology kept advancing. In 1989, the movie Abyss featured a creature that had a surface that reflected the environment. The same technique was used in 1991, when Terminator II had a fully modelled 3D human character that had a metallic surface. (Puhakka 2008, 24.)

The first full length computer animated film was Toy Story (1995) by Pixar (Puhakka 2008, 24.). The first computer animated film that attempted to create a fully realistic world was Final Fantasy: The Spirits Within in 2001. Although not lacking from capabilities, the film fell short due to issues with the character's unrealistic movements and skin, making audiences feel uneasy. Much of this continues to be a problem of animation: getting the characters to move right. (Chopine, 2011, 7.) The development of motion capture has solved some of these issues.

3 MOTION CAPTURE

For CGI artists and animators, motion capture has become one of the most important tools of use (Liverman 2004, 13). Motion capture basically translates real world coordinates into 3D space. Each of these coordinates correspond to a point in the 3D space forming a unified action of the recorded model.

3.1 Types of motion capture

Motion capture can be accomplished with different methods. Each of these technologies have their own strengths and weaknesses, there is not a single technology that is best for all uses.

3.1.1 Magnetic

Magnetic motion capture uses sensors placed on the body. The sensors are connected to a cabled control unit which calculates the magnetic fields generated by the sensors and designates the positional and rotational values of the markers. (3D Bloom. Motion Capture Evaluative Essay 2013.)

Magnetic mocap is one of the most technical types of motion capture. The advantages in magnetic motion capture are that the sensors track in real time and all the data is absolute. The sensors are prone to interference due to magnetism and the transmitters and receivers on the body limit range and movement. (Lonkar, Types of Motion Capture.)

3.1.2 Mechanical

Mechanical motion capture utilises an exoskeleton that the user wears. Exoskeleton consists of a device on each joint that has a rod and a measuring instrument. The instruments are designed to measure the subject's joint angles when moved, transforming physical motion into computer data. (Kitawaga & Windsor 2012, 11) Different types of mechanical motion capture other than exoskeletons, can be gloves or mechanical arms. These are great in uses where you need detailed animation of each joints movement, a piano player for instance. (3D Bloom. Motion Capture Evaluative Essay 2013.)

The good thing about mechanical motion capture is that there are no range limits and little to none interference. However, mechanical motion capture does have its issues. Most common exoskeleton suits can only track basic joint movement, making it a bit unrealistic and they do not have the same flexibility as optical mocap for instance. Mechanical mocap also has no ground awareness, so all jumping and vertical movement has to be animated separately. (Lonkar, Types of Motion Capture.)

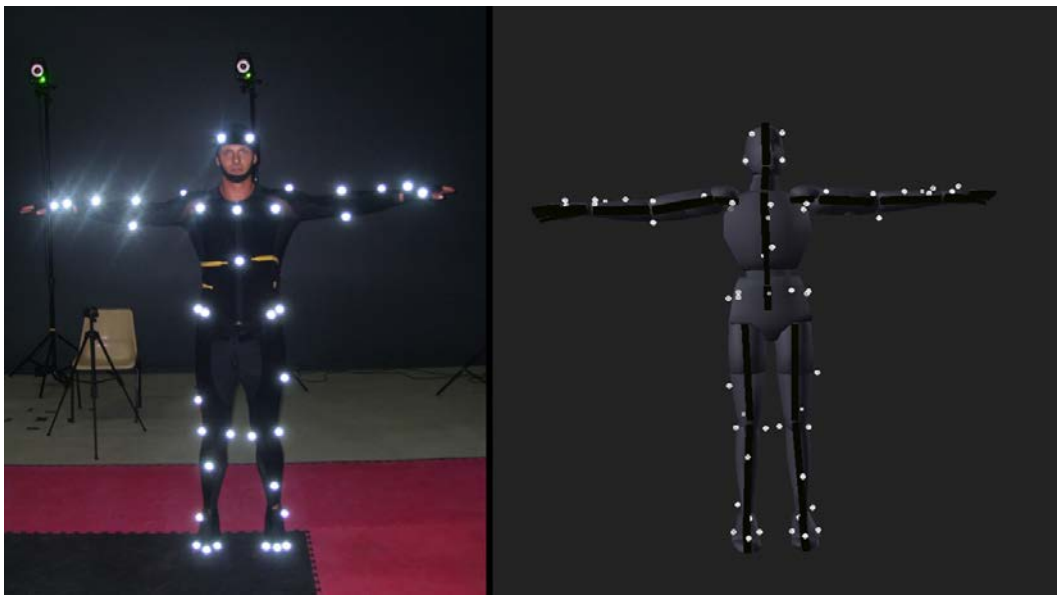
PICTURE 4. Mechanical mocap exoskeleton (Cronin)

3.1.3 Optical

Optical motion capture is the most popular form of motion capture technology due to its adaptability. Optical motion capture consists of reflective or pulse LED markers that are placed on the actor's body. The data is captured digitally via multiple cameras from different angles. (3D Bloom. Motion Capture Evaluative Essay 2013.)

Biggest advantages in optical mocap is the suit which is skin-tight and light, allowing maximum freedom in movement. With multiple cameras triangulating the light dot sensors, optical mocap also allows the capture of multiple actors at once. Optical mocap relies on visual connection with the camera, this causes the data to be accurate but also makes it prone to lose the trackers at times when the tracker is obscured by something,

this causes errors in the data. Other interference can happen from light sources. (Cronin, Systems and Applications of Motion 2014)



PICTURE 5. Optical mocap setup (Bedford, 2014)

4 PARTICLE EFFECTS

4.1 Use of particle effects

The demand for particle and dynamic physics is often in enhancing the visual effects or creating something that could not otherwise be done. Particle-effects are most commonly used as special effects in film, television series, animations and games. Almost every one of these mediums uses particle-effects at some point. Particles can be used to imitate natural phenomenon's, put fur on a character, add rain or smoke to a scene, simulate cloth or make wind. These are just some of the examples the particle-effects can create. (Manrique 2015, 167.)

4.2 Particle systems

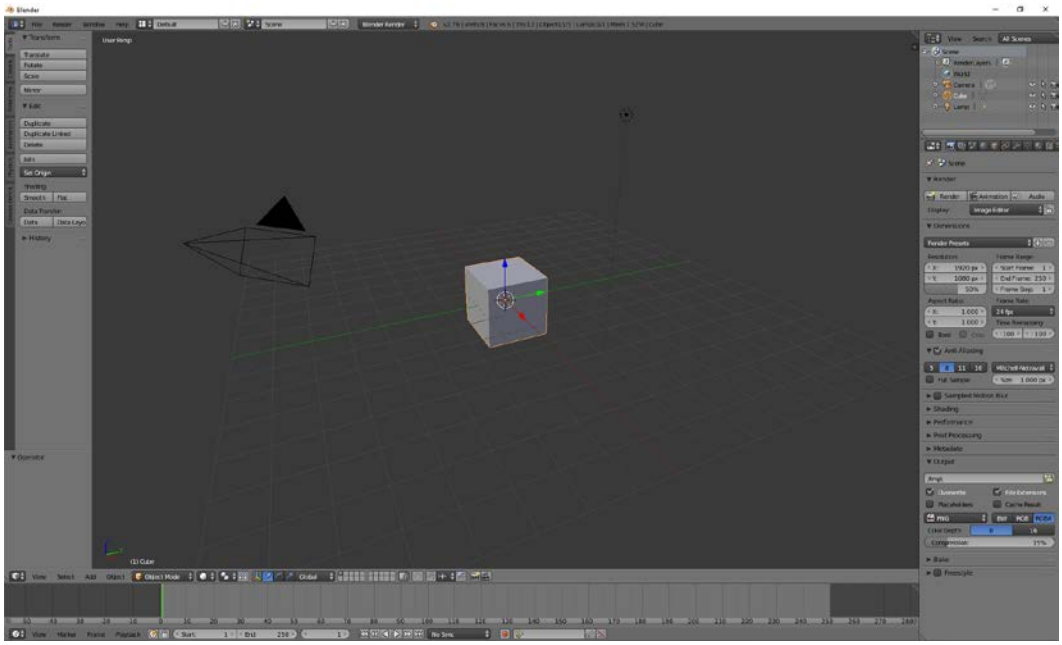
A particle system is a specialized group, that handles the particles as one unified object. The particles data is processed as straightforward as possible to make the calculation of hundreds of thousands of particles even possible. By combining all this data into one, it allows us to manipulate its parameters with more ease.

Particles are created from an emitter, which can be any type of 3D object or point. The emitter source and particles create a particle system. The particles emit from the objects or points surface and will move in a given value which can be influenced. Typical settings of modifying an emitters particle system is the starting speed, direction, flow and lifetime of the particles.

5 BLENDER

I chose Blender as my 3D software because of my previous experience with it, and because in my opinion it is one of the most powerful 3D toolsets available.

Blender first came to be as an in-house application, then for a short while it was released for free online until it was sold for commercial use. In 2002 it became completely free and open source through the Blender Foundation. (Manrique 2015, 9.) To some it has a bigger learning curve than other 3D programs, but the UI has seen good improvements from its previous versions.



PICTURE 6. Blender's default interface (Hoikkala 2016)

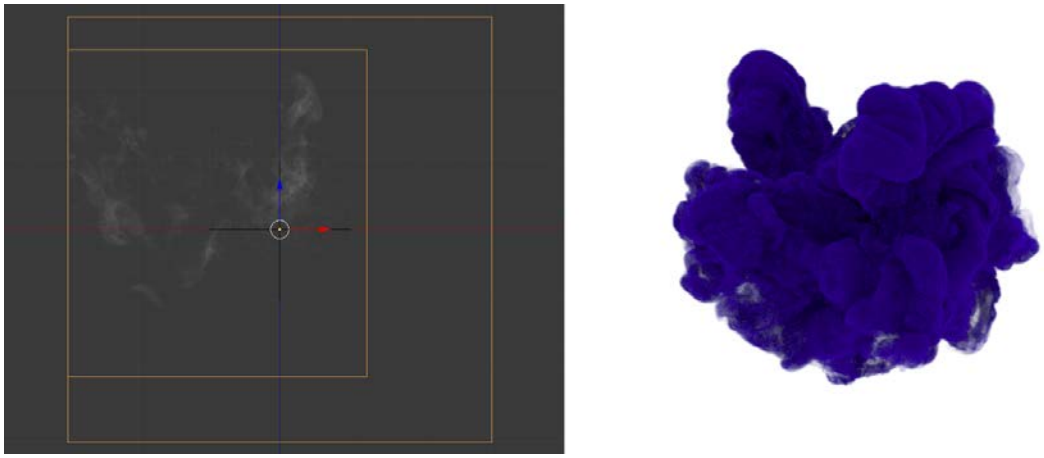
Typical uses of Blender are modelling, animating and effects.

5.1 Particle systems in Blender

Particle systems could be placed into two groups, emitter and hair based. As stated before, emitter based particles are dynamics such as fire, smoke, mist and other similar ones. Hair type simulations can be used to create strands, fur, grass and of course hair. According to Manrique (2014, 177) because we can add multiple particle effects to a mesh, we can for example add one particle system for a rat characters' fur, then add another one to make the whiskers.

5.1.1 Emitter

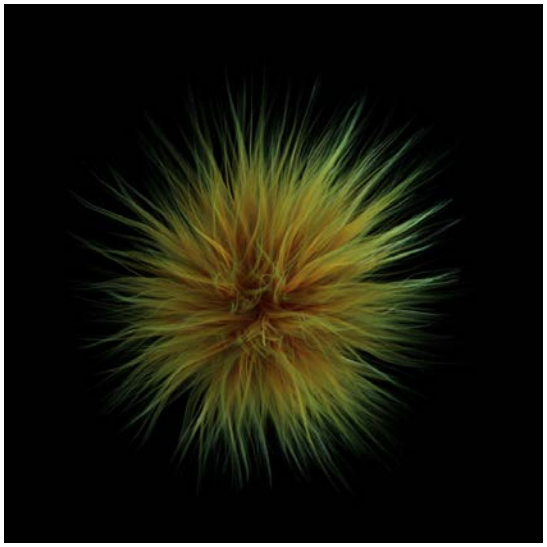
Emitter based simulations are usually chaotic by nature and have a certain lifespan, velocity and physics that can be tweaked endlessly. As it is seen in the picture below, the smoke emitting in the domain does not appear very thick, but when given density using material settings it will render looking like ink.



PICTURE 7. Smoke emitter and rendered image (Hoikkala 2016)

5.1.2 Hair

Hair based particles are static and much like real hair, you must first grow it and then style it. In the world of 3D this means setting the amount of hairs, their length, animating if needed – and finally rendering.



PICTURE 8. Fur made from particles (Blender 2.77 Manual)

5.2 Creating fire

A fire simulation can be referred to as a particle system, but in reality it consists of physics. Fire simulation can be created with a particle system, but in my opinion smoke domain has surpassed it.

5.2.1 Workflow for fire and smoke simulations

For a fire simulation, a flow object and a domain object are needed. Flow object is the object that will emit the fire and smoke, domain object are the bounds for the simulation. Here is a typical workflow for fire and smoke simulation.

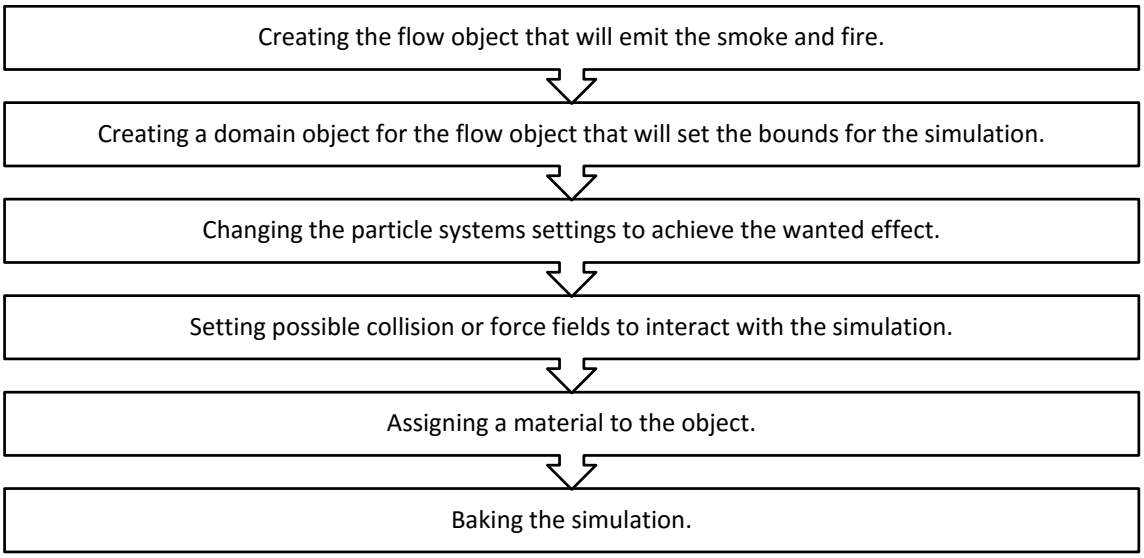


FIGURE 1. Typical workflow for a fire simulation (Hoikkala 2016)

5.2.2 Smoke simulation

Although called smoke, smoke simulation can be utilized to simulate great many things such as fire, dust, mist and other similar effects that flow through air (Manrique 2015, 195). Smoke simulation generates animated voxel textures which act like 2D but in 3D space. Movement of the smoke inside the domain is controlled by settings, smoke can also be influenced by force fields, other physics simulations and collisions from other objects. (Blender 2.77 Manual)

6 CASE: FIRE SIMULATION

6.1 Introduction

The case-work for this thesis is a fire simulation and the final implementation will be applying the fire simulation by using motion capture data based on real human action to create an abstract 3D render of a character on fire.

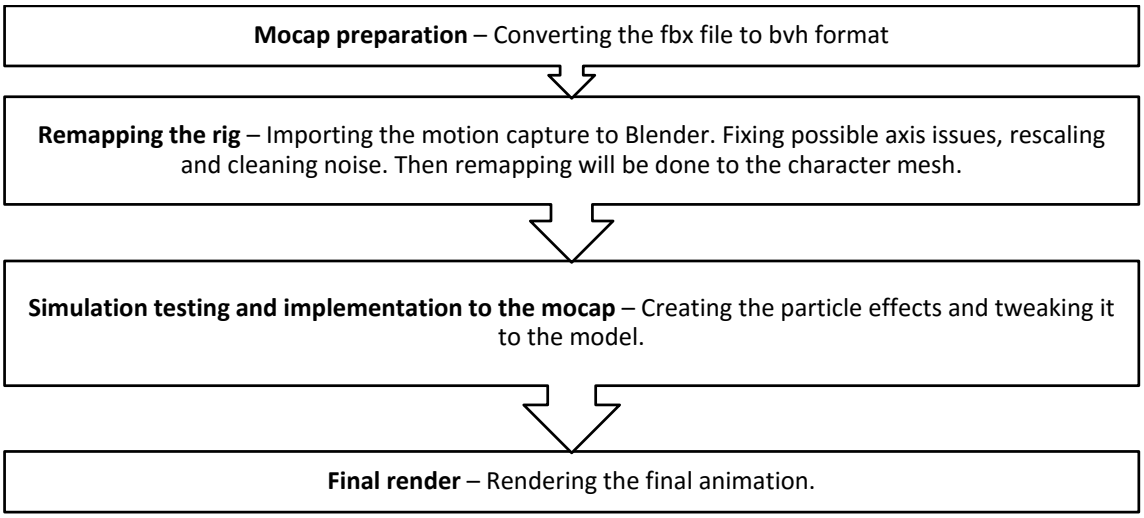


FIGURE 2. Project workflow (Hoikkala 2016)

I have been involved in sports production for two seasons by creating player introductions for Tappara's hockey team. Trying to look for new ways to produce visuals is vital. Tappara's colours are orange and blue. Symbolically, orange and blue can be portrayed in fire and ice. Fire and ice are natural elements that fit into ice hockey well. Due to these reasons I have chosen to interpret fire as a human character in my project work.

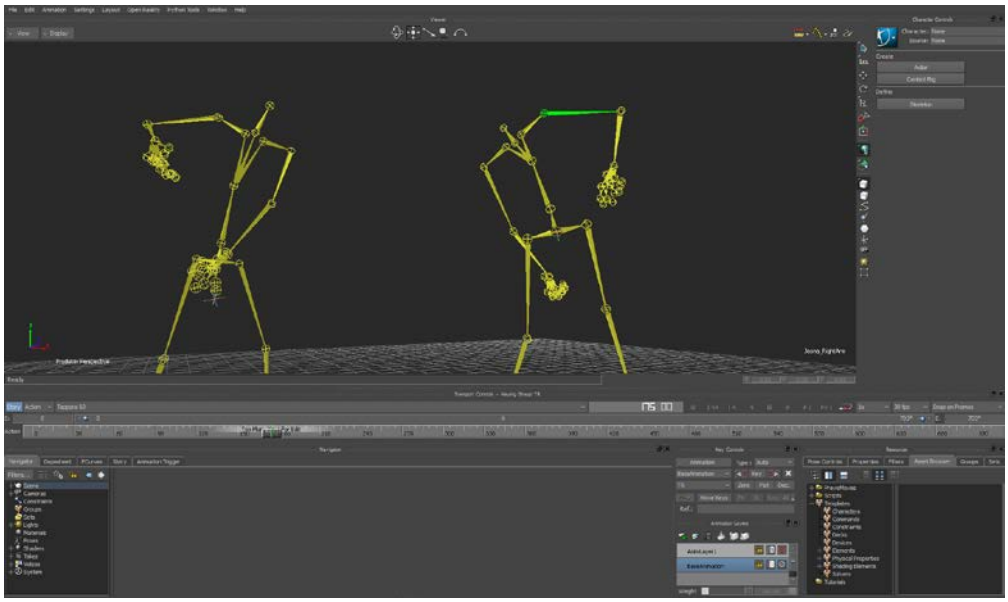
6.2 Motion capture data

The motion capture data used in this project has been shot early 2015. Some of the footage was used in material for Tappara in 2015-2016, but in my opinion it was never used as much as it could have been. The mocap is of two characters performing typical actions related to ice hockey.

6.2.1 Conversion

The shot footage uses FBX (Filmbox) file format. However, due to issues involved in importing FBX to Blender such as wrong bone names, bones missing and overall issues in animation, they needed to be converted into FBX into BVH file format. In my experience, BVH is an easier format to work with. BVH has been created specifically for motion capture and skeletal information data (UW Graphics Group. 2016.).

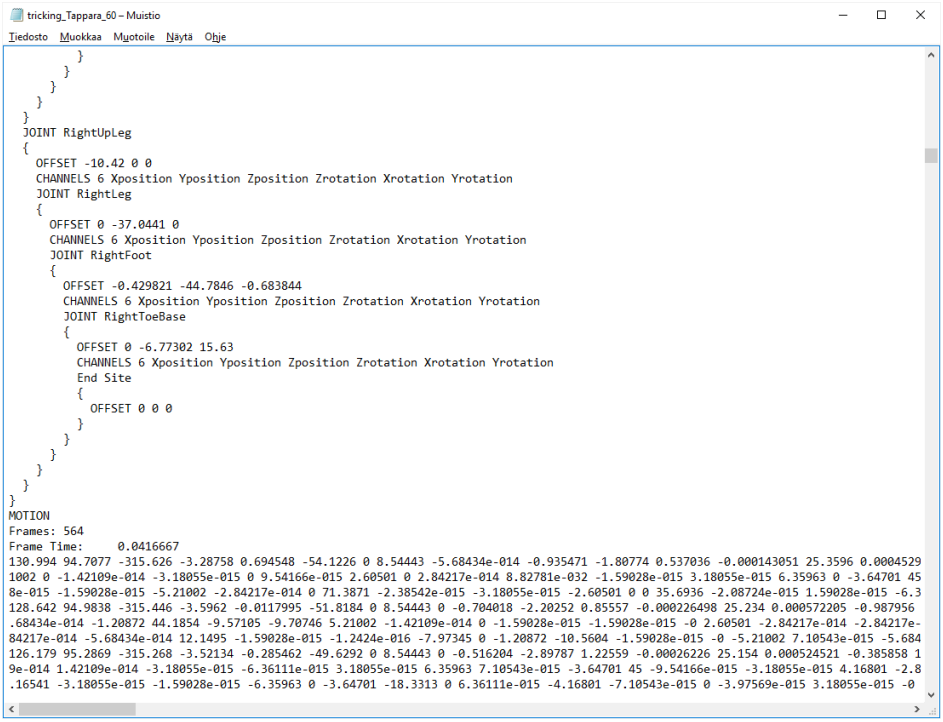
Autodesk MotionBuilder is used to convert the file. In Autodesk MotionBuilder the FBX file is first imported into the program, then exported as BVH. In the export options, the sampling rate can be changed. In this case, the original sampling rate was 60 frames per second, this many frames are not necessary in this case so the sampling rate will be set to 24 fps. The program will then export the mocap as BHV, but an additional edit to the file needs to be done before it can be imported into Blender.



PICTURE 9. Mocap skeleton in Motionbuilder (Hoikkala 2016)

The actor skeletons were named during the mocap shooting, which causes problems in Blender's interpretation of the footage. One of the mocap actor's name was Joona, so each bone in that skeletons group starts with "Joona_", for an example in the picture the green highlighted upper right hand bone is named "Joona_RightArm" instead of just "RightArm". This means that when we import the motion capture data, Blender arranges the bones by name, but when they all start with the name "Joona_", the order becomes disarrayed and some of the bone data becomes missing.

Before importing into Blender, the BVH file will be opened in a text editor and all “Joona_” will be replaced with a blank. In a text editor the mocap data consists of the bone hierarchy and after that there are the motion coordinates. When the name is removed, this solves the issue with bone data and the bones are named properly such as: hips, neck, spine, shins, thigh and arms. Now the mocap data is ready to be imported into Blender for rigging.



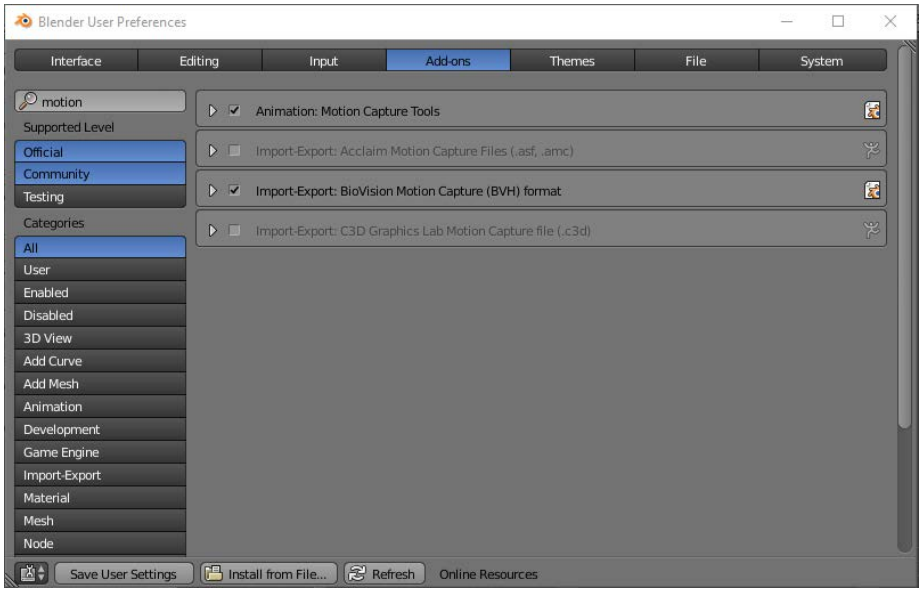
PICTURE 10. Mocap data opened into Notepad (Hoikkala 2016)

6.2.2 Rigging

In order to be able to use the mocap, something to animate to, a rigged mesh is needed. This is where armature, or skeleton comes into play. The mocap is an animated rig, but it needs a mesh to give the motion to. In this case, MakeHuman software was used to create a basic human mesh, it is a free open source software that allows to create 3D characters for different types of uses. The mesh rig is first imported into Blender, the mesh itself can be hidden for now, so the bones can be selected with more ease.

6.2.3 Retargeting

For easier retargeting process, Blender's built in addon tool for motion capture is used. The mocap rig will then be imported into Blender.

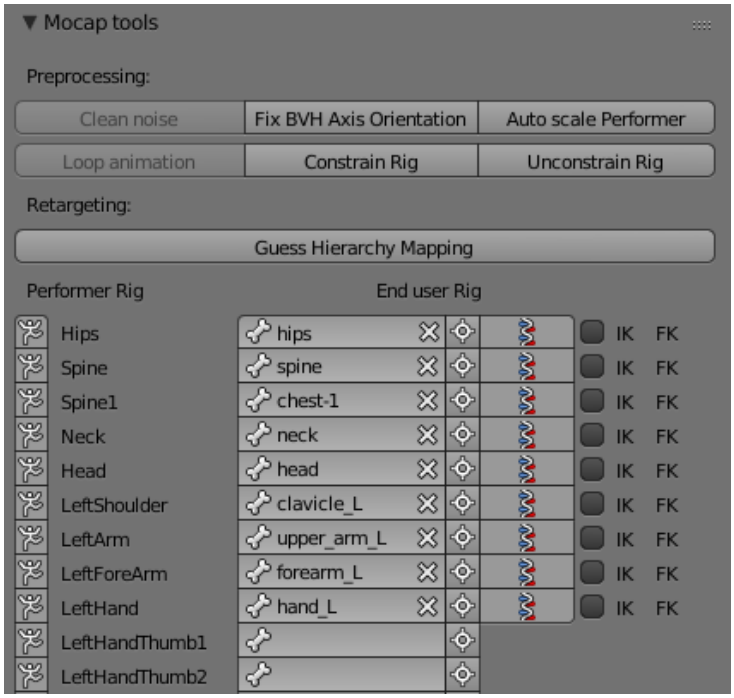


PICTURE 11. Blender's Motion Capture tools addon enabled (Hoikkala 2016)

Mocap tools is used from the object tab. Suggested workflow for rigging is first rescaling the armature, fixing axis orientation if necessary and then cleaning noise from the rig, then the retargeting can begin.

Retargeting is telling the system the relation between the two armatures, what bone mirrors to which in each armature. Easing the process, the "Names" display option set on in the armature settings. Mocap tools has an option to "Guess hierarchy", this feature tries to guess bones hierarchy if they have any similarities. However, most of the time the feature does not work perfectly, so manual mapping is required.

Once both of the armatures are selected and in pose mode, the UI opens up a list of all the bones. There are two ways to map a bone, next to each bone is a selection box, which sets the mapping of that bone to which is currently selected. It is also possible to type the name of the bone to the field.



PICTURE 12. Mocap tools UI with bones mapped (Hoikkala 2016)

6.2.4 Inverse kinematics and forward kinematics

Inverse kinematics “IK” is simplifying the animation process. IK retargeting is setting a parent for a chain of bones. So for instance, tip of the hand is the root bone and the rest of the bones up to the shoulder are child bones. Thus when IK chain is set for the arm, when the tip of the hand is moved, rest of the hand will follow. This eases the animation process, if done by hand. However, when we are dealing with mocap data IK constraints are rarely needed, and motion capture works best when working with forward kinematics. Forward kinematics are not constrained to any point in space which works for mocap because each frame in the armature is already animated, which is the point of mocap.

6.2.5 Twisted bones

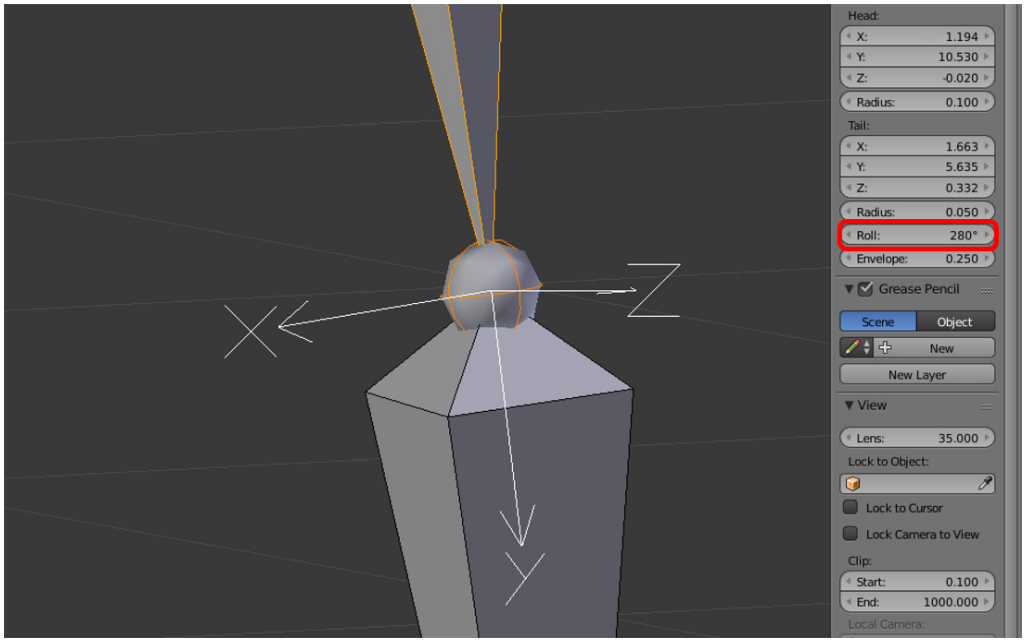
When all necessary bones have been mapped, the outlook can be previewed by checking the “Advanced Retarget” box. At this point, the mapped bones should mimic the performer rig. If the skeletons movement looks as it should, the mesh can now be revealed to check if it also is working.

Motion capture files and end rigs always have differences. After retargeting, deformities in the mesh can be expected. As seen in the picture below, there are deformities in the

mesh creating twists in the wrist, left leg and toes. Mocap tools offers an option to fix the twist, but this option is not without its faults so I find it best to fix the deformities manually.

PICTURE 13. Deformities in the mesh (Hoikkala 2016)

To fix the twisting, the roll of the bones need to be changed. The axis display is turned on from the armature tab so that in edit mode the axis can be seen. Opening the properties panel with hotkey “N” or from the bone tab, the roll of the bone can be altered. I found it best to try what works by going between edit mode and the pose mode to see when the mesh begins to look normal.



PICTURE 14. Bones axis roll fixed in edit mode (Hoikkala 2016)

The toes are fixed by going to bone constraints and deselecting the bones location copy from the x-axis.

PICTURE 15. Twisting in bones has been fixed (Hoikkala 2016)

The twisting issues are now fixed, there could still be finer attention into detail but for this case it does not matter since we are not looking for a hyper realistic human and the mesh itself won't be visible.

6.2.6 Finalizing the rig

The mocap tool has a specialized tool for finalizing the retargeting, however, I found it more simple to set a bone constraint to the hip bone. Using the retargeting action has caused some errors and seems unreliable at times, so this is why I chose another method. Selecting the end rigs hips bone, a bone constraint is set for location. The performer rig is then as the target and the head/tail is set to hips. Now when the animation is played, the end rig is working as intended.

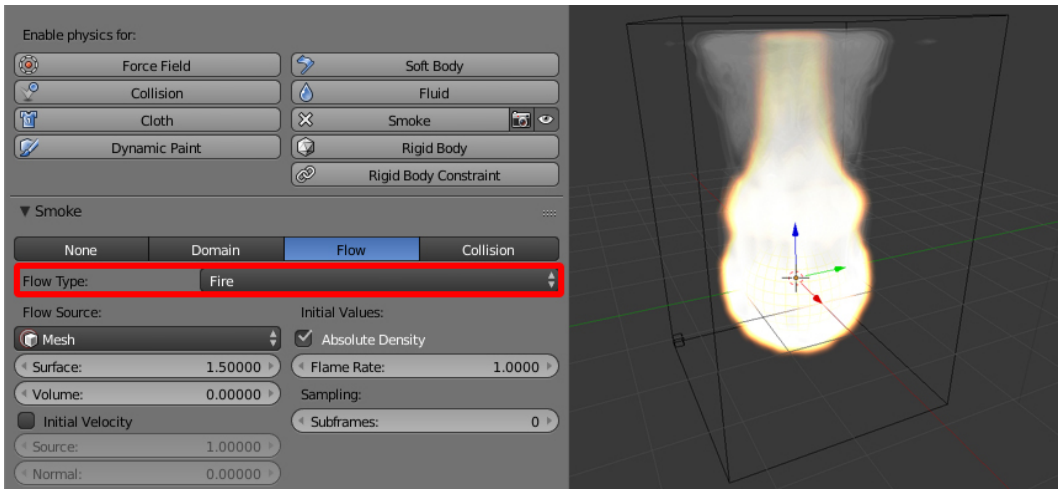
In creation of the fire simulation, I did testing with two methods; the Blenders internal render engine and Cycles render engine.

6.3 Building a fire using Cycles render engine

Blenders Cycles engine is the second default render engine in Blender. Cycles render engine tries to simulate real world lighting which is good for achieving more realistic scenes.

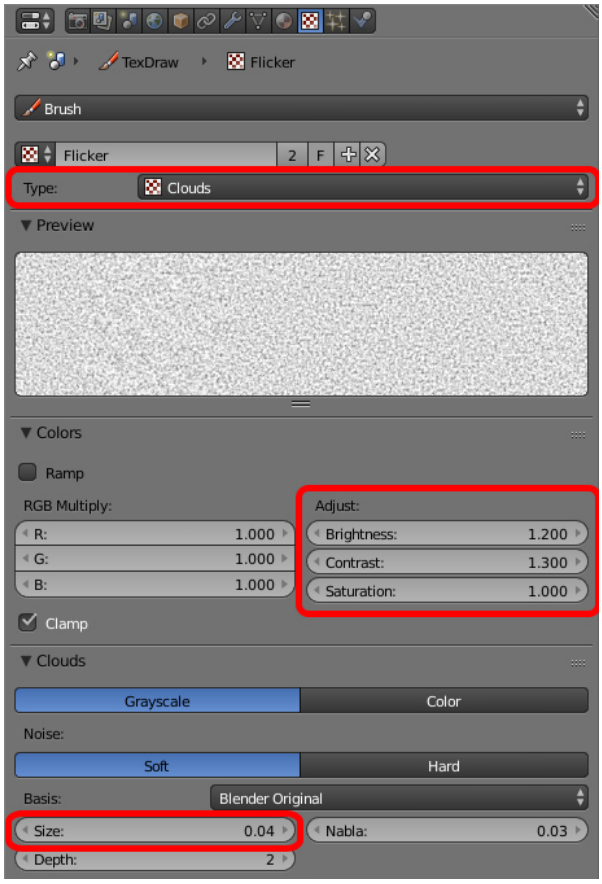
6.3.1 Fire emitter settings

Starting to create the fire simulation, a basic UV sphere is created to which the fire simulation is applied to. Blender's quick effect is selected from the object menu and from there the quick smoke is selected. Quick effects give a good base to work on. Quick smoke turns the selected object into a smoke flow object and also gives it a smoke domain around the object so it is ready for tweaking, thus removing the need to create the domain manually. By default, the object's flow type is smoke but it will be changed into fire.



PICTURE 16. Flow type of the object (Hoikkala 2016)

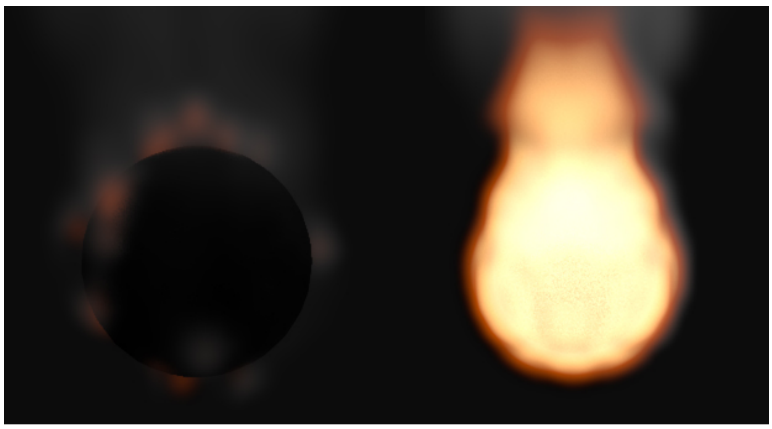
If the animation is played now it already resembles fire, but wanting a more realistic end result more is needed. Adding to the chaotic element of fire, a fractal texture will be added to the fire which is then animated. In the physics settings of the fire emitter, under Smoke Flow Advanced, texture for the emitter is enabled and from the dropdown menu "Tex" is selected. Proceeding to the texture tab "Tex" is selected and the type is changed to "Clouds". From there, the brightness and contrast settings are adjusted to bring out a high contrast to the texture. Size of the texture should also be changed. When this is animated it will mimic a flames sort of chaotic flickering. The name of the texture is changed to "Flicker" to tell it apart from other textures.



PICTURE 17. Texture settings (Hoikkala 2016)

To animate the texture, going back to physics settings and under the Smoke Flow Advanced; the timeline should be set to the first frame, then a keyframe is set to the offset slider. After this the timeline is set forward and the value in the offset is set up and another keyframe will be set so that the texture will animate giving the flame some flickering effect.

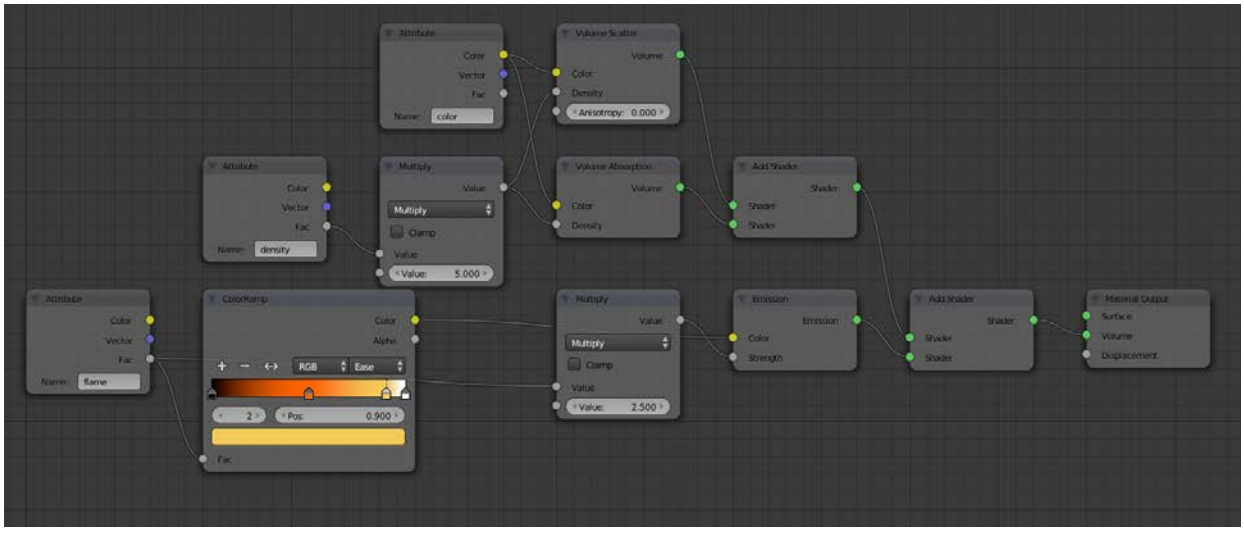
Something to note about the flickering texture, it can have a huge impact on the simulation depending on the settings. A low brightness and contrast means that the flame will die out very quickly and never reaching its highest burning point. Then again, a really bright texture with high contrast and larger size will make the fire very volatile and rippling. Best settings for a steady flame is so that the flame is consistent, with a little life of its own. The texture setting can also be utilized for animation, for instance if one would want the flame to die out at a certain point.



PICTURE 18. Texture differences (Hoikkala 2016)

6.3.2 Creating the fire material using nodes

Trying to make the fire look more realistic, a material will be added to the fire. Cycles uses nodes which gives options to define materials, lights and backgrounds limitlessly. Cycles needs to be turned on from the dropdown menu, otherwise the render will not show at all. When the engine is changed to cycles, the scene needs an assigned material to render to. To start working on the material, the domain is selected and in the materials setting “Use nodes” is enabled. After this the node editor is opened into the viewport the adding of shaders can begin. A guide for the fire and smoke was used for the material setup as seen in the picture below.



PICTURE 19. Node setup for the fire and smoke material. (Blender 2.77 Manual)

Attributes are important to name, because they are what the system is referring to when it looks for information on how to portray the material in question.
Simulations take large amounts of memory from the computer. To ease the calculation and final rendering process, Blender is able to pre-compute the simulation to save time on rendering, this is called baking. (Blender 2.77 Manual) Before baking the simulation, the cache for the data has to be named first or Blender might lose all the cache if the program is restarted.

6.3.3 Domain settings

To simulate smoke, the domain is selected and in the physics settings “Smoke Adaptive Domain” and “Smoke High Resolution” are switched on and the strength of the noise sampling is set to 4. Adaptive domain shortens render times by calculating the needed space for the simulation and high resolution brings more detail into the smoke. The smoke settings were tweaked.

6.3.4 Results

Advantages in working with Cycles render engine is that Cycles can take advantage of GPU processed rendering instead of CPU. If equipped with a powerful graphics card, this can significantly decrease rendering times. The fire simulation turned out alright and I was able to get it to work in a satisfactory way. The process could be further improved by perhaps adding embers or something more to the simulation.

PICTURE 20. Rendered fire in Cycles (Hoikkala 2016)

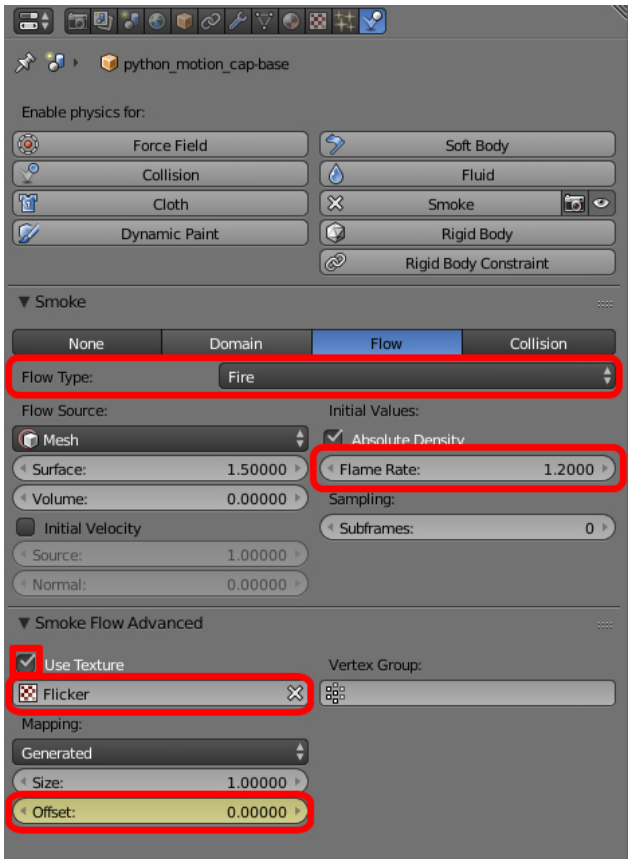
The fire is quite realistic although a little blurry.

6.4 Building a fire with Blender Render engine

Creating the fire simulation with blender render engine is quite similar to cycles in principal. The workflow is also similar, only the settings have variation. Biggest differences are in how the materials for the fire and smoke are created and the rendering process itself.

6.4.1 Fire emitter settings

Just like with Cycles render engine method, an object is needed, so that it can be turned into a fire emitter. Once again, a UV Sphere is added to the scene and quick smoke is added to it. Now the scene has an emitter and a domain. Flow type of the UV Sphere is changed into fire from the physics settings. The flame rate is the amount of fuel for the fire, so the higher the number the bigger the flames, the amount is risen to 1.2 from the default 1.

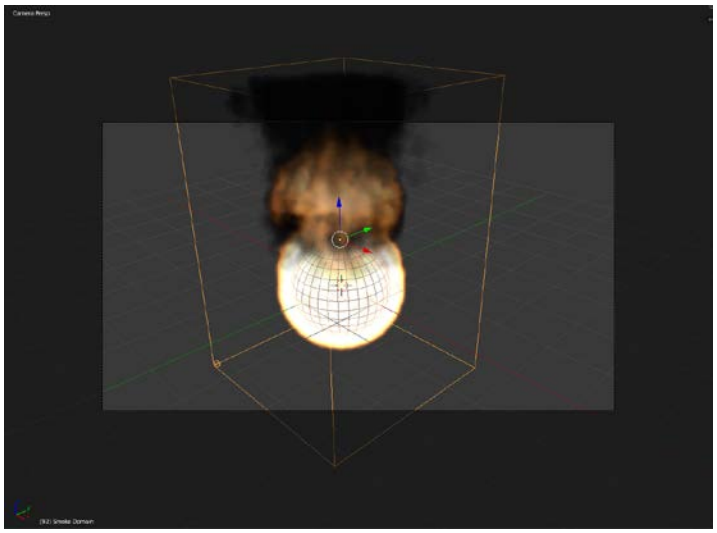


PICTURE 21. Emitter settings (Hoikkala 2016)

Same as the fire for cycles, a new texture for the flow of the fire will be added and then animated (picture 17). The texture is then added in the physics settings and the offset is animated. The amount of offset to animate depends on the length of the simulation. I found that offset mapping value of 0.1 per 10 frames worked quite good. Meaning that a simulation that lasts for 250 frames would have an offset of 20.00 in the final frame.

6.4.2 Domain settings

The domain settings are changed from the physics tab. Resolution of 78 was used for the render but a smaller amount is fine for testing, final render can go even up to 128 or more. Smoke flames settings are tweaked with increasing the amount of smoke created by the fire emitter. Smoke adaptive domain is also turned on for quicker rending times. For testing high resolution, smoke is not necessary to turn on, but for the final render it is good to have it more detailed.

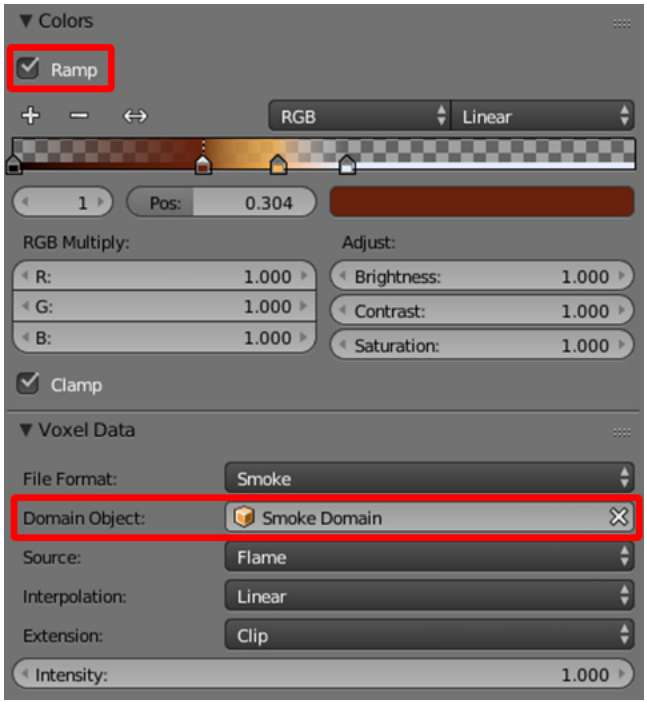


PICTURE 22. Emitting fire (Hoikkala 2016)

6.4.3 Creating the fire material

As mentioned before, creating the material varies in Blender render engine from Cycles render. First a material is added for the domain with the density scale set to 4. Then moving on to textures, a new texture is added from the texture settings tab. The fire material is added by adding a new texture and setting the type to voxel data which is blender's default for smoke source. The colouring of the flame will be made with ramp colouring

which will try to mimic a fires brighter starting point to the darker tip surrounding the flame before disappearing. Transparent colours are then added to each side to control the flames disappearance.



PICTURE 23. Colouring of the flame (Hoikkala 2016)

In the voxel data, the domain object should be smoke domain and the source as flame. Interpolation method means how the data between voxels is handled, linear seemed to have the best results. Influence options set how much the texture affects the object, so from this setting the density is wanted to be set to 1 and emission to 6.

Another texture is added, this one will control the smoke density. No colouring will be added to the smoke. Voxel data is set to same as the flame setting. For the influence, density and reflection colour should be turned on.

6.4.4 Results

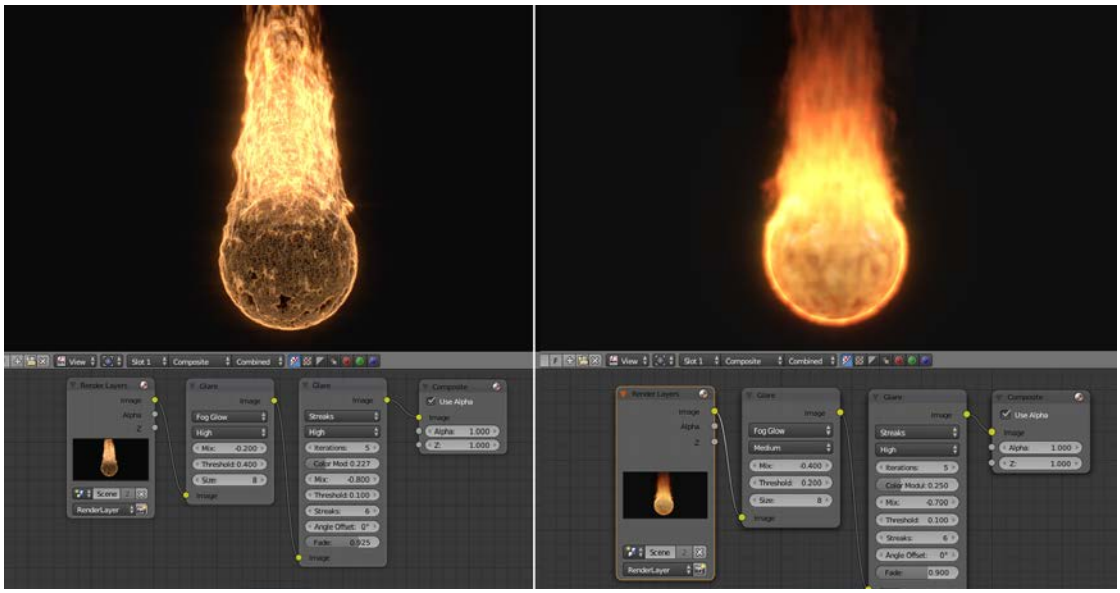
I was surprised at how well fire could be simulated with Blender Render and the end result was very pleasing. Compared to Cycles, Blender Render simulation has more sharpness to the flame but the colouring could use improvement.

PICTURE 24. Rendered flame with Blender Render (Hoikkala 2016)

6.5 Further improvement with post-processing

This applies both for Cycles and Blender Render methods. Compositing nodes allow to add post-processing effects to the scene. Typical Compositing effects include tweaking elements like: colour balance, colour tints, contrast, blur or shine.

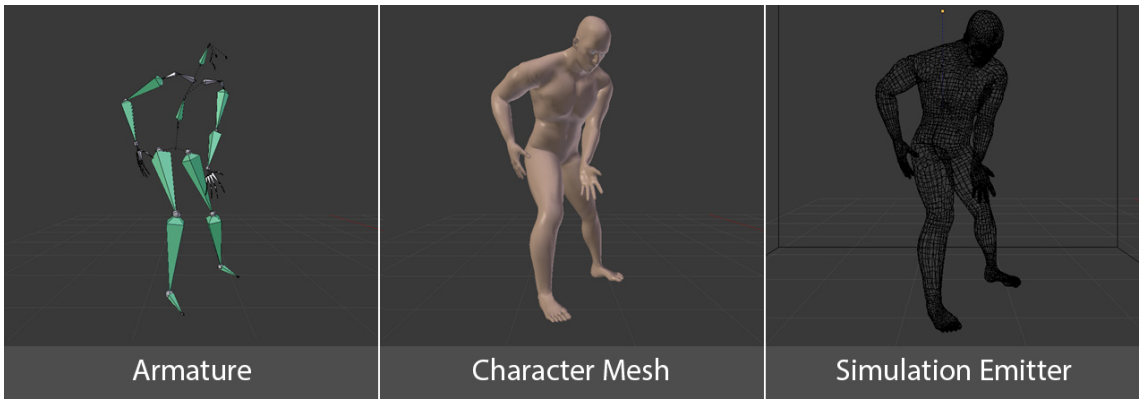
A shine is going to be added to the scene to give the fire a bit more glow. Going to the node editor and selecting the Compositing Nodes with Nodes enabled, two Glare nodes are added between Render Layers and Composite node. The glare types are changed to Fog Glow and Streaks and then modified. As it can be seen, post-processing helps bring out more glow and colour from the renders.



PICTURE 25. Blender Render and Cycles with post-processing (Hoikkala 2016)

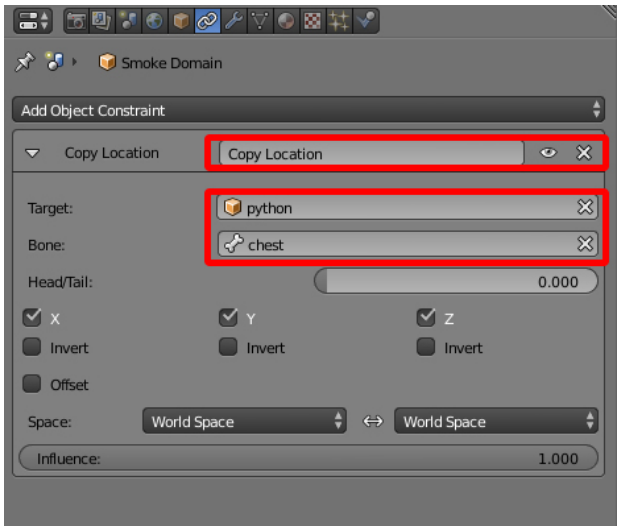
7 FINAL IMPLEMENTATION

For the final implementation, the fire simulation is applied to the mocap data. For this section the parts that differ from the simulation workflow will be explained. Otherwise the settings are to be expected to be as they were in the previous part.



PICTURE 26. The rigged mesh with applied effect (Hoikkala 2016)

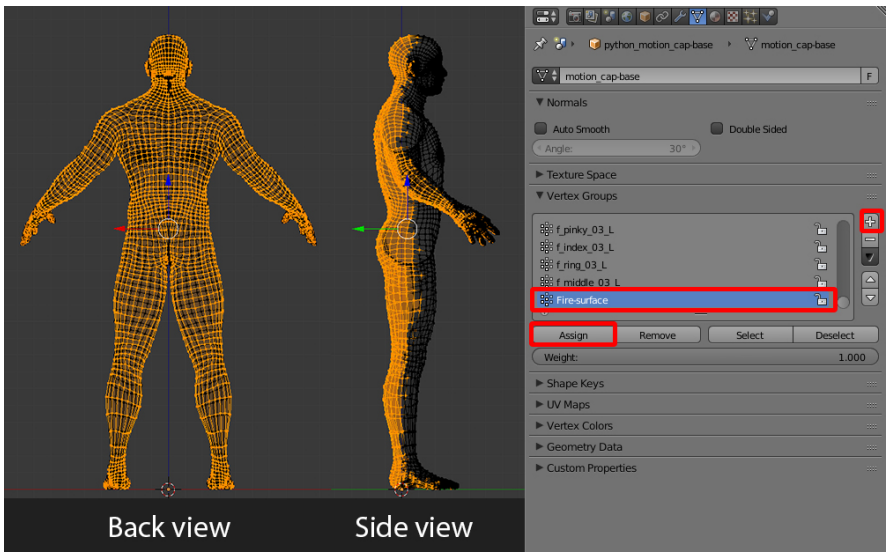
The animated mesh is to be treated as the UV Sphere would be, but with some differences. The smoke domain is scaled so that the fire has enough room. Because the character is moving, the domain needs to follow with it. The domain does not do this automatically, so a constraint is added to it. The constraint is set to copy the location of said target which in this case is the armature called “python” by default. The bone target is then set to a central bone in the armature, the chest bone for instance.



PICTURE 27. Constraint for the domain (Hoikkala 2016)

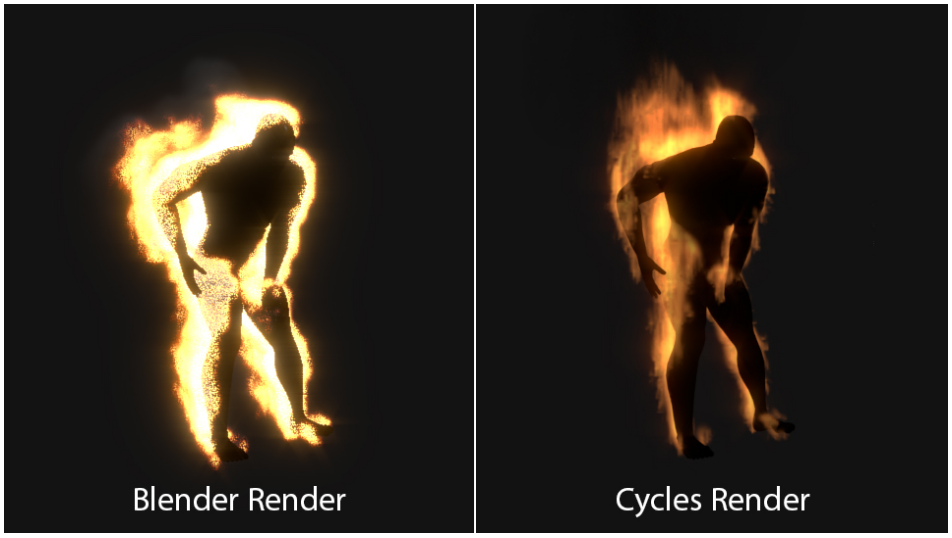
When an object is made into an emitter, the whole object will emit particles. During testing I found that if the whole character will emit fire, the fire is a bit overwhelming and the movement of the character will not show as well. Improving the simulation, emission is changed so that fire will only emit from the characters back.

Accomplishing this means that the area that is wanted to emit fire is selected, made into a vertex group and enabled in the emitter settings. The character mesh is selected and interaction mode is set to “Edit mode”. In edit mode, the mesh should be in T-position and the vertices can be selected. The perspective is changed with the numeric keypad to show the back view of the character and then the back is selected using the box-select tool. With the back selected, in the data settings there is a menu called vertex groups. A new vertex group is added by clicking on the plus sign, the vertex group is renamed to “Fire-surface” for recognition and finally it is assigned. The vertex group can now be added in the characters mesh physics settings, under the smoke flow advanced menu and added from the vertex group selection. When the simulation is baked, the fire will now emit from the selected vertex group.



PICTURE 28. Adding a vertex group (Hoikkala 2016)

The final renders of the animation worked out alright



PICTURE 29. Rendered fire character (Hoikkala 2016)

7.1 VFX

Creating a fire simulation in Blender is only a part in the pipeline involved in creating visual effects. A complete work of motion graphics includes many more aspects such as camera work, lighting effects, additional physics and environments around the character. This thesis' focus is on the fire simulation and motion capture data, but an example is created to demonstrate possible uses of the final renders.

In this example, creating the final VFX would include improving the fire with adding effects that would make it glow and shine light brighter. A surface is created by adding an ice texture. Lighting for the background is added which would be animated. Foreground effect would be added with a snow fall. The VFX could be used in ice hockey production, for advertising, promotional material or player introduction intros.

As it is seen in the VFX comparison picture below (picture 30), VFX can make a simple picture something completely different.

PICTURE 30. VFX comparison (Hoikkala 2016)

8 PROBLEMS

Simulations can take a lot of time to calculate and render. The process can slow Blender down a lot and can sometimes result in crashes. Best practise was to lower the resolution and other heavy effects like the smoke to get a faster workflow. When working on the simulation, it is best not to drag the timeline to watch the animation, this caused Blender to freeze and crash at times. The simulation should also be baked before playing, otherwise the simulation might not work as intended. Before a full render multiple tests should be made and at times compromises must be made between quality and time. Depending on the scenes length and resolution, a single clip can take multiple hours in rendering time. Good thing about Blender is that when an animation is rendered, it is saved as an image sequence. Image sequence means that the render process can be stopped and continued later from the frame it last rendered. The image sequence is then compiled in Adobe's After Effects where post-processing effects are added.

9 CONCLUSION

The subject of the thesis was quite challenging and took a lot of self-study. Especially the motion capture involved aspects that could not be learned from research or tutorials and involved a lot of trial and error. For the fire simulation, I don’t think I could have achieved the same results if I would have started them from scratch and the tutorials from Blender Guru helped me a lot with it. I’d say in general 3D takes a lot of studying and for a particle effect creation such as this project, it is good to know the basics of 3D simulation creation. During the process, I came to understand a lot more from Blender and its simulation tools and materials. Working with numerous settings might feel overwhelming at the start, but come with more ease the more they are experimented with.

The case-work turned out decent and I think it forms a good idea of what fire simulation creation in Blender is like. Besides the simulation, I was pleased to learn about motion capture data and the workflow involved in making it work in Blender.

I would suggest that Blender is a viable and effective tool in creating visual effects using simulations with motion capture data. Given more time on the subject, I believe that even more realistic fire effect could be accomplished.

REFERENCES

Bedford, A. 2014. Focus: Sports Science Motion Capture. Released 13.10.2014. Read 25.04.2016.
https://andrewbedford.com/2014/10/13/369/

Blender Guru. Tutorials, How to Render Fire in Cycles. Read 27.4.2016.
https://www.blenderguru.com/tutorials/make-fire-cycles/

Blender Manual 2.77. Render, Cycles Render Engine, Materials, Volume. Taken 2.5.2016.
https://www.blender.org/manual/render/cycles/materials/volume.html

Chopine, Ami. 2011. 3D ART ESSENTIALS: The Fundamentals of 3D, Modeling, Texturing, and Animation. Elsevier.

Classical MoCap Part 2: Systems and Applications of Motion. Read 18.1.2016.
http://stringvisions.ovationpress.com/2014/10/classical-mocap-part-2/

Manrique, M. 2014. Blender for Animation and Film-Based Production. Boca Raton, FL: A K Peters/CRC Press.

MiikaHweb. Blender 2.5: Realistinen tuli savusimulaattorilla. Read 11.5.2016.
http://www.miikahweb.com/en/articles%2Fblenderfire

Kitagawa, M., Windsor, B. 2012. MoCap for Artists: Workflow and Techniques for Motion Capture. Elsevier Inc.

Motion Capture Maestro Andy Serkis on 'Dawn of the Planet of the Apes' and Revolutionizing Cinema. Read 25.1.2016.
http://www.thedailybeast.com/articles/2014/07/08/motion-capture-maestro-andy-serkis-on-dawn-of-the-planet-of-the-apes-and-revolutionizing-cinema.html

Liverman, Matthew. 2004. Animator's Motion Capture Guide: Organizing, Managing, and Editing. Hingham, Mass, Charles River Media.

Puhakka, A. 2008. 3D-grafiikka. Talentum.

S. Corazza, L. Mündermann, A. M. Chaudhari, T. Demattio, C. Cobelli, T. P. Andriacchi. Annals of Biomedical Engineering. Volume 34, Issue 6, pp 1019-1029. A Markerless Motion Capture System to Study Musculoskeletal Biomechanics: Visual Hull and Simulated Annealing Approach.

Sturman, David J. A Brief History of Motion Capture for Computer Character Animation. MEDIALAB. Read 18.1.2016.
http://cose.math.bas.bg/Sci_Visualization/compAnim/animation/character_animation/motion_capture/history1.htm

Stokstad, M., Cothren, M. 2011. Art History. Volume One. Fourth Edition. Pearson Education Inc.

Silverman, Matt. 2004. History of Rotoscoping. Phoenix Editorial.

UW Graphics Group. Course article: Biovision BVH. Read 11.4.2016. http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html