PHYSICALLY-BASED ANIMATION FOLLOW
A Comparison of Player Experiences

Bachelor Degree Project in Informatics

30 ECTS
Spring term 2019

Henrik Lennartsson
Lina Wijkander

Supervisor: Marcus Toftedahl
Examiner: Niklas Torstensson
Abstract

Animations in the game industry are considered a bottleneck today, and a way to speed up the development in this area is by letting the computer procedurally generate the animations. This research has explored blending physics with *keyframed animations* in games, to create interactive and responsive characters, and to find out how these animations are perceived by players. This way of animating characters in real time has lately been used in a blossoming game genre called *fumblecore*. By letting physics do part of animation in real time, unpredictable and unique animations can be created.

Two demos of a *fumblecore* game was created and tested on players with various gaming backgrounds. One had a traditional *keyframed animation*, and the other had a *physically-blended animation*. The results showed that the majority of participants of the study preferred the *physically-blended animation* over the traditionally *keyframed* one.

**Keywords:** animation, procedural, physics-based, animation follow
Table of Contents

1 Introduction ........................................................................................................ 1

2 Background .......................................................................................................... 2
   2.1 Procedural Generated Content in Games ..................................................... 3
   2.2 Game Animation Techniques .................................................................. 3
       2.2.1 Keyframing .............................................................................. 4
       2.2.2 Forward Kinematics ................................................................. 4
       2.2.3 Inverse Kinematics ................................................................. 4
   2.3 Procedural Animation .............................................................................. 4
       2.3.1 Fully Procedural..................................................................... 5
       2.3.2 Blended ............................................................................... 5
       2.3.3 Physics-Based ...................................................................... 6

3 Problem ................................................................................................................ 7
   3.1 Method ....................................................................................................... 7
       3.1.1 Artefact and Evaluation .......................................................... 8
       3.1.2 Playtesting and Interview ....................................................... 8

4 Implementation .................................................................................................. 11
   4.1 Character Setup ...................................................................................... 11
   4.2 Animation Follow ..................................................................................... 15
   4.3 Finite State Machine .............................................................................. 16
   4.4 Character Balance .................................................................................... 17
   4.5 Implementation Summary ....................................................................... 21

5 Evaluation/Utvärdering ..................................................................................... 23
   5.1 Presentation of Study .............................................................................. 23
       5.1.1 Pilot Study ........................................................................... 23
       5.1.2 Final Study ........................................................................... 23
       5.1.3 Study Results ....................................................................... 24
   5.2 Analysis ..................................................................................................... 25
       5.2.1 Humoristic and Fumibly Feel .................................................. 26
       5.2.2 Satisfactory Character Control .............................................. 26
       5.2.3 Interaction & Reaction – Unpredictable & Unique Animations ... 27
   5.3 Conclusion ................................................................................................. 28

6 Concluding Remarks .......................................................................................... 29
   6.1 Summary ................................................................................................... 29
   6.2 Discussion .................................................................................................. 29
       6.2.1 Social Gains .......................................................................... 29
       6.2.2 Gender .................................................................................. 30
       6.2.3 The Importance of Responsive Characters ................................ 31
       6.2.4 Tools to Affect Mood ............................................................... 31
       6.2.5 Credibility of Results, and Risks ............................................. 31
   6.3 Future Work ............................................................................................... 32
       6.3.1 Short Term ............................................................................ 32
       6.3.2 Long Term ............................................................................ 32
       6.3.3 Broad Perspective ................................................................. 32

References .............................................................................................................. 33
1 Introduction

In the entertainment business, the gaming industry has surpassed all other media, and is now the largest entertainment media in the world (OppenheimerFunds Distributor, Inc 2018), with customers of various backgrounds, interests and needs. To meet the demand for innovation, entertainment and novel experience from the customers, the industry is looking for ways to speed up production or reduce its costs. Animation in games is costly, and creating animations for character movement, swaying trees or cloth that flies in the wind takes time and many talented artists and animators to create results that are believable and realistic. A method to try to combat the costly animations is to generate them procedurally, meaning that the computer stands for most of the work through algorithms, instead of being hand made by an artist. Research in this area has been ongoing for over thirty years (Geijtenbeek & Pronost 2012; Mahoney 1997), but in a pace that only has picked up in the recent 10 or so years (Geijtenbeek & Pronost 2012). These techniques have branched out into several categories, such as animation repurposing in form of inverse kinematics (Johansen 2009), motion graphs (Chen & Steed 2011), and blending (Edsall 2003). This recent increase in interest has made more tools and techniques available, and this thesis focus on exploring a part of the area.

There is still a gap between the research that has been done, research that is currently being performed and the use of these techniques in the game industry (McAloon 2019). There could be possible gains for the gaming industry with fast and easy solutions, but researchers often give solutions to complex problems, in a way that is not always user friendly. A rare example of where the fields of research and practice met, is the physics engine Euphoria, developed by NaturalMotion and used by Rockstar Games in their game engine RAGE. These tools are used in games such as Grand Theft Auto V (Rockstar Games 2013) and Red Dead Redemption 2 (Rockstar Games 2018). -It required substantial time and effort, but the results came out successful (McKeand 2017). Bridging the gap between research and practice could give more successful examples like these, but since it is a complex area, mutual benefits needs to be identified – such as the cost reduction from a production perspective. With better, more user-friendly tools and products, more developers could get creative, finding innovative solutions to real-time animation, and thus boost the progress in this field.

This research is targeting the area of physics-based animations, an area that lately has bloomed with a new genre of games called fumblecore. Fumblecore can be described as a game genre where physics play a big part of the game, especially on playable characters, making them have a fumbly appearance. The aim of this research is to find, and bridge typical techniques used in games today with physics-based techniques used in the fumblecore genre. Many games require that a certain style, mood or personality is kept for the character (Geijtenbeek & Pronost 2012) and that can be a problem for physics-based techniques due to their unpredictable nature. But at the same time, they provide accurate real-time simulations (Zucconi 2017) for interaction with the game world. This research will aim to produce a responsive character with a wide range of animations driven by physics-based techniques, whilst maintaining the higher level of user control a traditional keyframed animation would give.

To reach this aim, an artefact blending physics with traditional keyframed animations is developed. The artefact, focusing on the playable character, is then used to compare the blended animation setup with a traditionally animated version. With this setup, the goal is to find out how players perceive the blended animation approach of fumblecore games.
2 Background

The video game industry has been growing fast (and still is) and is now the biggest economic force in entertainment (OppenheimerFunds Distributor, Inc 2018; Wijman 2018) (Figure 1). As the industry grows, the expectations on what it produces is raised, as consumers expect to be baffled by new and exciting technologies and novel experiences. The higher consumer expectations lead to a larger production, making the expenses of the production grow, both in terms of money and time (Connor 2018; Schreier 2017; Superannuation 2014). Improvements have to be made in areas where bottlenecks usually arise in the production and an area where research has been done in the past couple of decades is procedural generation of in-game content. This means that content is generated by algorithms instead of by manpower, effectively cutting down on costs in both time and money. Even though there are decades of research in the area, researchers and the industry have only been scratching the surface of what can be accomplished with procedural generation techniques (Smith 2014). The future of game entertainment is expected to have more procedural generated content to combat bottlenecks, improve production, and meet expectations of consumers (Yannakakis & Togelius 2015).

![Gaming’s Revenue Eclipsed All Other Major Entertainment Categories](image)

**Figure 1** Revenue of traditional media as of May 2018. Modified Image (OppenheimerFunds Distributor, Inc 2018).

An area that is currently of interest to improve in, is animation (Antonissen & Riff 2010), and procedurally generating animations can be done in a number of different ways, which will be described later in this work. Animation is an integral part of games and as graphics are becoming better and player expectations are growing, more work has to be put into animations to meet these expectations and keep players engaged and immersed. It is however not feasible to create animations for every single action that can be taken by a player, if the character is expected to interact with several objects in the surrounding game world. There would simply be too much animation work to cover all different poses and animations that occurs as part of interaction, in a media where you not only follow a character, but actually play the role of it. Animations in movies are different because of the fact that everything is decided beforehand (i.e. it is linear) whereas in games actions are taken in real-time (it is interactive). A game can be perceived as better if the player is immersed (Carr, Buckingham, Burn & Schott 2006), and this is something you strive for as a game developer. If animations are not keeping up with the technological advancement in graphics (Johansen 2009), they risk hurting immersion of
games as a result. There are different kinds of procedural animations and some of these techniques will be handled in this research.

2.1 Procedural Generated Content in Games

By using procedural generation, more diverse content can be produced at a faster pace. Various techniques and algorithms are used, depending on what kind of content is desired. One of the earlier techniques in procedural generation was pseudo-random number generation, which is an algorithm that produces a random sequence of numbers. One of the first games to use this technique, was Elite (Acornsoft 1984) (Hendrikx, Meijer, Van Der Velden & Iosup 2013), where the technique was used to procedurally generate a large number of galaxies for the player to travel to. This technique and other procedural generation techniques has since been used in a wide range of areas.

Some techniques require more or less human involvement for desired results. For example, a terrain can be generated and then be manipulated by a designer before generation continues or work on the content is finished. These assisted techniques (De Carli, Bevilacqua, Pozzer & D’Ornellas 2011), which requires human involvement, also requires resources in form of manpower and time, but has the potential to generate more detailed and meaningful content. Human assisted techniques are often used to counter content that ends up being too random, meaningless or repetitive (Geijtenbeek & Pronost 2012; Kenwright 2012). Non-assisted techniques, which means that the algorithm does all the work, can be solid groundwork for a designer to build on or if the content will not be subjected to closer inspection or be key part of the content as a whole (De Carli et al. 2011). It is important to be able to decide if and where procedural generated content fits in a game, and when it doesn’t (de Araujo & Souto 2017).

An advantage of procedural generation is the endless results that can be produced by less work. This means a large variation and replayability in games can be achieved, and as it doesn’t require as much time and resources, the cost of development drops. Despite this, Yannakakis & Togelius (2015) points out that procedurally generated content faces a risk of not being sufficiently varied, reliable and qualitative, as a designer might not always have enough control over the generated content. This can result in the player experiencing the content as repetitive despite the fact that this is what procedural generation is trying to fight. De Carli et al. (2011) also warns about the use of procedural techniques and mentions that they are only as useful as the game they fit in. de Araujo & Souto (2017) agrees that there is a risk with using procedural generation irresponsibly, as it tends to make the game repetitive and less interesting, but goes on by saying that a good balance between traditional and procedural generated content is needed for the opportunity of endless content.

In fumblecore games, procedurally generated contents are used in such a way that players control some specific parts of the playable character and then physics does most of the work on the animation, which often produces a comical effect. An example of this is the game Octodad: Dadliest Catch (Young Horses 2014), where the player controls the tip of an octopus’s limbs, and the rest of the specific limbs follow along as they are affected by physics, making them fly all over the place uncontrollably, resulting in a fumbly but comical movement.

2.2 Game Animation Techniques

As mentioned above, animations are an important area to improve the workflow in, as it risks not being able to keep up with the technological advancement in graphics (Johansen 2009).
Some animation techniques that are used as standards in the game industry is keyframing, Forward Kinematics and Inverse Kinematics, which are all used in different kinds of procedural animation.

2.2.1 Keyframing
A keyframed animation consists of a number of specific frames, called keyframes. These frames are specific and have certain conditions that makes a particular frame a keyframe. By interpolating between keyframes over a duration, you can create new frames that exists between these keyframes. This is called tweening. The keyframes describe the transformation of the object, be it position, scale or shape at that point of time, and the inbetween states are calculated by the interpolation algorithm between the keyframes to create an illusion of motion. For an artist, the control given by keyframes is important, but it comes at the expense of heavy memory usage (Ahlström, Holmqvist, & Goswami 2017).

2.2.2 Forward Kinematics
Forward Kinematics (FK) is the most common way to produce animations in games, and is achieved by changing the angle of the joints in a skeleton to their desired position, with the child joints changing according to their parent joint. The end effectors position can be determined from the angles and relative positions from the total amount of joints (Unity Technologies 2019). FK provides the point in space that the end effector (e.g. the hand of a character) reaches with the given movement. This position is reached by knowing the angles of the previous joints, meaning that changing the rotation angle of the joints will lead to having the effector reach different points in space (Zucconi 2017).

2.2.3 Inverse Kinematics
In Inverse Kinematics (IK), the opposite situation as in FK is solved. IK targets a point in space and then solves how to move, typically a robotic arm, to have the end effector reach it by working backwards. With the given point in space, it finds a valid orientation of the character’s joints so that the end effector reaches the desired position. E.g. if it is desired that a character’s hands follow an edge as the character climbs along it, the hand is moved to the designated position and IK solves how the joints are forced to get the arm into the desired stance (Johansen 2009; Unity Technologies 2018; Zucconi 2017).

2.3 Procedural Animation
Just like other procedurally generated content, to animate procedurally means that algorithms or mathematical expressions do part of, or all of the work in an animation. (Mahoney 1997).

Animation is currently a bottleneck in game production as almost everything has to be animated in some kind of way. It is costly to create believable animations that require motion capture (Champandard 2012) or traditional keyframe animations. Motion capture requires a motion capture studio and actors, and keyframe animations requires a team of animators (Togelius, Champandard, Lanzi, Mateas, Paiva, Preuss & Stanley 2013). The increased public demands on games increases the need for more varied and interesting animations, which can be hard to meet, especially for small game studios with limited resources. Increasing the amount of animations or keyframes is not a very sustainable solution in these cases and therefore new, alternative solutions to this problem are being sought after (Johansen 2009).

Procedural animation techniques are currently a of topic interest, both for researchers and the game industry (Antonissen and Riff 2010). The game industry is currently using procedural
animation techniques to solve a bottleneck that prevents projects to keep up with the technological advance in graphics, and the game industry is interested in finding better solutions to further improve or find new methods in the area (Champandard 2012; Johansen 2009; Samyn, Van Hoecke, Pieters, Hollemeersch, Demeulemeester and van de Walle 2012). These procedurally generated animation techniques are well suited to replace time-consuming and repetitive tasks in traditional animation techniques, such as keyframing animations (Chen and Li 2016), FK and IK.

One of the advantages of procedurally animated content is that, just like other procedurally generated content, a versatile variation can be achieved by relatively little work. With few keyframe animations, you can achieve a similarly looking result as if an artist had made more keyframes than they actually had, or a physics engine can handle the generation of keyframes in real-time. If the generated content is not subject to direct inspection from the viewer or player, procedural animation can be used to faster create content and move the production along. The animation artists team in the movie Moana (Walt Disney Studios Motion Pictures 2016) used procedural animation techniques to create some of the waves in the movie. Traditional animation was used for waves that were in focus and needed a great amount of detail, while procedural techniques were used otherwise, or in some cases a combination of both techniques (Byun & Stomakhin 2017).

Procedural animations are however, at a disadvantage when it comes to the artistic depth and detail that a professional artist can produce. As such, they are typically avoided when creating specific animations such as those used in cutscenes or movies, where the producer might intend for some very specific actions to be taken by characters. If used on a character in a game, procedural animations might be successful when it comes to reacting and adapting to ongoing scenarios, but it does take away control from professional artists. It does however give control to the player.

In fumblecore games, the disadvantages of procedural animations are used to the genre’s advantage. Instead of hiding the results produced by physics, the genre embraces it, as the result is often humorous. As such, it does not aim to hide itself from direct inspection from the player.

2.3.1 Fully Procedural
Procedurally generating a motion is done by creating general rules or procedures that control how an animation proceeds. This means that when an algorithm creates changes in movement on an object without using pre-defined keyframes, it can be called procedural animation (NonLinear Educating Inc 2003). Procedurally generating a snowstorm would for instance have a randomization procedure that decides the motion of a snowflake in the animation. This saves an artist’s time from having to decide motion for every single snowflake in the scene and speeds up the animation process.

2.3.2 Blended
Blending is when an already existing animation is repurposed for a specific scenario (Samyn et al. 2012). For example, if a character has to crouch more or less deep depending on the gap it is trying to enter, the artist would have to create several different kinds of crouching animations. By using blending, a general crouching animation is combined with procedural techniques, and the character can crouch more or less depending on how big the gap is. A benefit of blending traditional keyframe animation with procedural animation is the reduction in amount of keyframes needed (Ahlström, Holmqvist, & Goswami 2017). By blending keyframes with procedural animation through interpolation, less workload is put on the artist, and more diverse animations can be produced. This is important in games where an in-game character needs to adapt to surroundings as it is moving around or performing
specific actions. An example of this, is how the main character in *Uncharted 3* (Sony Computer Entertainment 2011) climbs into moving vehicles, adapting the animations based on where the enters the car. A typical technique used to implement this kind of movement is IK (Johansen 2009). This allows the animation artist to focus on work that requires more detailed animations, such as cutscenes.

### 2.3.3 Physics-Based

Another common way to procedurally animate is through simulating physics. One example where this is common is animating water, such as in the previously mentioned *Moana* (Walt Disney Studios Motion Pictures 2016) example. Having an artist animate water would take an enormous amount of time as the artist would have to keyframe all the points of the surface of the water. Procedurally animating the water instead, by letting a physics system with fluid dynamics take care of the work, lets the artist focus on other work. (Mahoney 1997; Zucconi 2017). Physics-based animation is most commonly used for simulating natural phenomena (Geijtenbeek & Pronost 2012), such as gases, water, fire and clouds (Mahoney 1997).

Physics-based animation is also used for characters to simulate authentic fall and death animations, through the use of ragdolls. Ragdolls are a means of doing procedural animations, by imitating the effects the laws of physics exert on a body. An example where physics-based character animation works well, is if you want a character to do a dangerous stunt that is not suitable for motion capture (Geijtenbeek & Pronost 2012), such as diving off an edge, landing head first on the ground. A ragdoll is created by connecting limbs with rigid bodies through joints. A rigid body is considered a continuous distribution of mass that does not start to deform when pressure it applied to it. These joints have a certain degree of freedom for how much the limbs can rotate and twist. Through the use of rigid body physics, the ragdoll will simulate a body falling to the ground (Zucconi 2017). As such, putting limbs and joints together, and then applying physics, does not cause a character to walk; it will simply fall due to gravity. Moving specific limbs then has to be done through code. By manipulating rigid bodies through the use of forces and torques, muscles can be imitated (Geijtenbeek & Pronost 2012; McKeand 2017), which in turn gives physically accurate, but also unpredictable animations with lack of control (Geijtenbeek & Pronost 2012). This kind of animation has lately been adopted by the *fumblecore* genre in games such as *Gang Beasts* (Double Fine Productions, Inc. 2017), *Surgeon Simulator* (Bossa Studios 2013) and *I Am Bread* (Bossa Studios 2015).

The *fumblecore* genre embraces how silly this kind of animation looks where physics take care of most part of animations. As control of physics-based animation is hard to achieve, and often looks sluggish, game companies tend to stick to kinematic-based animations for more aesthetically appealing animations (Geijtenbeek & Pronost 2012). The fact that a whole new genre (*fumblecore*) of games are built around physics-based animation, there might be something more here than just a novelty factor (Hash & Isbister 2011). If it is here to stay, the lack of control, personality and mood of the animations are problems to address in order to bridge the gap between currently standard animation techniques and physics-based animations.
3 Problem

Pre-recorded animations require a knowledge of what will happen in the game in order for the artist to create an animation for that situation. The animations are thereby predictable, repetitive and unresponsive to unpredictable events (Kenwright 2012) and it is not feasible to make animations for all possible scenarios and interactions in a game.

As part of this thesis work, the authors have been in Shanghai, China producing a co-operative game in-house at a local game studio. This gave insights in industry related development complexities and made said authors more interested in the field of procedurally generated animations. -It was decided to explore this area by making a fumblecore game to understand how this works in practice. The characters in the game are two robots with flexible, fumbly and humoristic movements; just like in any fumblecore game. The twist is that it is desirable to maintain good user control (as to not lose the feeling you are a robot in full control), while allowing physical interaction with the environment. The floppy look is merely on the aesthetic front for the robots, while still being responsive to physics interaction like a physics based animated character would. Hence, the problem statement is how to maintain the fumbly, comical feel of a physics-based character, but still maintain satisfactory control of movement.

One option was to blend between normal animation during the more predictable parts of the game and swap to full ragdoll physics during more unpredictable interactions (Geijtenbeek & Pronost 2012) such as being belly flopped down a slope by your friend while solving a particularly complicated task. The goal of the game project was to have the best of both worlds – both control and goofiness, at the same time.

The approach would be to attempt maintaining ragdoll physics and mimicking a pre-recorded keyframe animation. This way making sure to maintain a certain, sought-after feel and personality of the character, but still maintaining the unpredictable, unique animations.

It was desired to explore different aspects of how the balance between ragdoll physics and constraint rules of the animation tracing, affects the floppy feel during certain states of the character in the game. There would be periodic constraints of the animation, but the ragdoll physics would never be turned off completely, maintaining unique animations and unpredictability, thus making sure the content remains interesting for the player (McKeand 2017). Since a game was produced, it was of interest how players would actually think of this solution. Was it a viable one, considering the criteria of the game? Would this approach result in a satisfactory experience for the player?

The problem area would boil down to the question:

*How is a physics-based procedurally generated, animation following character, perceived by players?*

3.1 Method

To investigate if it is possible to have a fumbly and humorous feel, combined with precise controls, an artefact was created. The artefact takes the form of a game with two playable robots interacting with each other and the environment in a game world. Separate maps in the game consists of either the physics-based procedurally generated animation with animation follow, or the one with traditional animation made up by keyframes. Playtests will be conducted on a target audience of players to evaluate the artefact, focusing on the animations. It is of particular interest to find out the players opinion regarding the comical appeal and “moreness” or ”juiciness” of the animations. To find out how the animations are perceived by the players, two game sessions with the different versions of the animations will be played.
Afterwards the players will be asked questions regarding the play session. The format for these questions will be a qualitative study in the form of a semi-structured interview. This means having more room for the interviewee to speak their mind with open ended questions, but also control enough to lead them into the right track to be able to provide the data needed to answer the questions brought up in this study.

What is considered humorous and comical is very personal and subjective and is something that will have to be considered when analysing the answers.

3.1.1 Artefact and Evaluation

The game is created using Unity software (Unity Technologies 2019), a game engine that the group creating the artefact was comfortable with. The game’s aesthetic style is based on similar games from the fumblecore genre, which are characterized by a low amount of polygons for models and light concise colours to get a simplistic look. A traditional animated character, provided by a 3D animation artist, has been implemented, and from this, a ragdoll has been created with rigidbodies and joints for specific parts of the ragdoll. The game is a co-operative game, and has players complete physics-based tasks to advance in the game, such as throwing things or the other player around, stepping on buttons or using magnets to travel around. The procedural part of the animation is based on looking at the current state of the keyframed animation and trying to match that state at every frame, while still exerting physical forces on every rigidbody on the character. This gives an animation that responds to the environment, but still aims to keep a specific control action.

The specific artefact for this thesis project will be extracted from the produced game and lend itself to our evaluative material, in the form of two demos with either the physics-blended animation or the traditional animated character that can be played. The actions the player can perform are walking, running, belly flopping, jumping, giving thumbs up, and picking up and throwing things. Further description of how this works technically can be reviewed in the implementation chapter of this work. Hopefully a playtest like this, followed up by a quick questionnaire to get background data and an interview, will give more detailed answers and opportunity to go back and review the answers for additional data. It also allows for follow-up questions if there is any confusion or unclearness.

The data collected will be compiled and analysed to provide material for further development and improvement of physically-based, responsive character animations.

The playtests, questionnaire and interviews will be piloted to stake out errors and identify potential risks.

3.1.2 Playtesting and Interview

The target audience of the playtest and interview is gamers. This is because it is of more interest knowing how gamers, who are more experienced in this area, perceive the animations. This gives data that is relevant to be able to answer the research question, as the intended field of the study is targeted towards the gaming industry. At the playtest sessions, computers and controllers will be available and the participant will not need any additional equipment of their own to participate. The participants will playtest the game alone and the interview will be a one-to-one interview (Denscombe 2014).

The participants will be briefed shortly in what animation and procedural animation in games is, how the artefact is constructed, how the playtest session and interview will be conducted and how long the session will be. Each demo will be played for 5 minutes each, a total of 10 minutes playtesting. The participants will informed about that the session is completely voluntary and that consent to participate can be withdrawn at any time without risk of
The participants will also be informed about what the data collected will be used for.

The playtesting will start with more detailed information on how it will happen. The player will be given a controller and instructions for input controls, and will then be left undisturbed for the playtest. There is no right or wrong in how to play, it is more of a playground, and it is encouraged to try and play around with things to see how it feels to play. The demos will be played in different order, to prevent unforeseen factors that might provide dependencies.

Before the playtest, the questionnaire will be handed out and after it has been answered and the playtest is done, the interview will take place. The participants will be informed that they will be asked some questions about experiences of the demos. There is no right or wrong here either, the interest simply lies in the player's opinions. The interview will be of a semi-structured format (Denscombe 2014). This format has open ended questions to let the interviewee be free enough to speak their mind and express their opinion, but also has structure enough to allow the interviewer to control sidetracking and make sure the relevant topics are discussed. Data analysis will take longer time than a fully structured interview with more quantitative data, but with the relatively low amount of interviewees for this study, it should pose no bigger issue. The interview will take about 10 minutes and consist of the following areas:

- What did you think of the animations in the different versions?
- Which of the animations did you prefer, and why?
- How did you feel in regards of character control in the different versions?
- Which version did you prefer in terms of control, and why?
- What do you think of the mood/feel/personality of the character?
- Do you believe this kind of animation could work in other games, and if so, what type of games/genres?

Asking for clarification or giving nudges in the right direction will be done for certain questions if needed to make sure to get data to be able to answer the research question. The following clarification questions are planned:

- How did you feel in regard to the interaction and response of the character (If it ran into things or got hit etc., how do you remember the response was?)
- How would you describe the characters feelings? What did you feel when looking at it run around and interact?
- Why do you prefer said animation/character?
- Did the animations and controls fit for the game/genre?

These questions will aid in finding out more about player's experiences regarding abstract areas such as the earlier mentioned “morenness” and “juicyness” of procedural animation. Has the procedural technique succeeded in adding something more than just more varied and time saving animations? It is also of interest to find out how the players find the comical appeal of the two demos. And lastly to find out if the feel/personality/soul of a character is lost due to automatic computer created animations.

When performing an interview there is a risk of the social desirability factor (Denscombe 2014), meaning that the participants might be giving answers that are in line with what they perceive the interviewers want to hear. Being aware of this, notes will be taken as the players try the game, as unconscious information such as exclamations or facial expressions can provide more pure and true opinions.

There is also a risk of the interviewer effect (Denscombe 2014) where the interviewee will answer differently depending prejudice regarding ethnicity, sex, age et cetera of the person asking the questions. Depending on the questions asked, the interviewee can feel uncomfortable, awkward or embarrassed and choose to answer what they perceive the
interviewer considers appropriate. Since the questions regard how the participants perceive the animations, they won’t be of any sensitive nature and no personal data will be collected. The target group of gamers is also similar to the one of the interviewers’ and hopefully these are reasons enough to not affect the quality of the data too much by this effect.

In accordance to the guidelines Denscombe (2014) brings up, the participants will be informed about:

- The purpose of the research.
- How the data collected will be used and that it is confidential.
- How long the session will be (approximately).
- Where the researchers are from, names and contact information.
- That participation is voluntary and anonymous.
- That participants can quit the session at any time without any reprisal.
- A thank you for participating.

This information will be given on the questionnaire paper, together with the following, to get the gaming background of the respondents:

- Age?
- Sex?
- How many hours do you estimate you play per week?
- Favourite genres in games?
- Do you consider yourself a competitive player?
- Have you heard of the fumblecore genre?
- Have you played any fumblecore games?
- What games do you play currently?
4 Implementation

As part of a bigger project being produced in ten weeks with a total for six members of different disciplines within game development, a game was produced. The game was created in Unity software, as there is a lot to gain from using an already existing game engine. It provided an already existing physics system, animation trees and state machines. It is also the game engine all six members of the project were most familiar with and the most widely used game engine for small scale game development.

To solve this research’s specific problem of maintaining control and goofyness of a playable character, it was decided to always maintain ragdoll physics on the character, whilst using animation follow at the same time.

To simulate how limbs and their muscles work, inner forces and torques were used, inspired by former research using this technique (Kenwright, Davison & Morgan 2013; Kenwright 2012; Kenwright 2013). Muscles’ capacity to change their length through contraction and relaxation are emulated with the use of spring via Unity software’s joint components. A spring, just like a muscle can pull together and reduce the distance between two attach points or loosen to elongate it. By using this technique with torques, unpredictable and unnatural looking results in form of external forces and “The hand of God” (Geijtenbeek & Pronost 2012) can be avoided. Further reasons for the use of animation follow, were that the project being developed had criterias for humoristic movement. As earlier mentioned, it is hard to portray different feelings for a character’s movement with the use of procedural animation. As such, having a traditional keyframe animation as a reference to try and follow, was a compromise to maintain a certain level of control of the character’s personality.

The process of creating the game can be divided into four subsections. Below follows a more detailed description of the procedure and design decisions being made during development. The subsections are:

- Character Setup
- Animation Follow
- Finite State Machine
- Character Balance

Character setup will give a description of how the character was constructed in the game engine, to achieve a ragdoll with limbs needed for this research. How blending an animation was done, with the use of animation following, will be handled in Procedural Animation. How different states of animation was controlled and transitioned between, will be described in the subsection finite state machine. Finally, a section handling the subject of Character Balance is included.

4.1 Character Setup

As the game was part of a collaborative project with a 3D-animator in the group, a character model was produced and animated by a person with knowledge in the field. The first prototype of our character was produced fairly fast with basic animations to go with it.
Unity software has a built-in function to create ragdolls, and as such, this function in the software was used. It requires a character mesh rigged up with bones (Figure 2) to be able to create a ragdoll. The different limbs are dragged to the respective components in the wizard and decide the total mass of your character (Figure 3), and the software creates all the colliders, rigidbodies and joints needed to make up the ragdoll. The Ragdoll Wizard also does minor setups of joints on the character limbs, linking the limbs together properly. While the game engine has this tool for setting up a ragdoll, manual tweaking has to be done if the character consists of more joints than the wizard allows. As this was the case for the model used for the research, extra joints where added to the ragdoll after the initial setup from the software, to properly match the model produced by the 3D animator. This meant that the connected rigidbodies in the models hierarchy had to be reassigned manually to make sure they were connected to the correct limbs. New colliders also had to be added and readjusted for the limbs to properly work and finally the joints created by the wizard had to be configured to restrict rotation between limbs (Figure 4).
**Figure 3** Setup of the ragdoll in Unity software’s Ragdoll Wizard.

**Figure 4** Joint being configured to restrict rotations of upper leg.
Since the Unity software creates a ragdoll out of the character, the body will just fall to the ground during runtime of the application, and as such, some kind of restraints or balancing are needed for the character to stand up. This will be handled later on in this work, as proper balancing of a character requires more work than one might expect.

When creating the ragdoll, a copy of the rigged character was saved to use for the hand made animations provided by the 3D animator in the group. This, to have the playable ragdoll character try to mimic the movement of the animated character at runtime. The mesh renderer of this character was later disabled to hide it from the players view.

When Unity software creates a ragdoll through the use of its Ragdoll Wizard, the component for joints used are character joints. This component is used for characters as ragdolls, but was not customizable enough for the needs of this research. As such, all character joints were replaced by the component configurable joints, as it incorporates functions from all other joint types the software has (Figure 5), and allows for more customization.

A script was then applied to all joints on the character and setup properly. This script is the heart of the animation blending and will be explained more in detail in the next subsection of this chapter.

Using joints on a character is mainly done to simulate muscles, as Unity software creates invisible springs between them. These springs try to pull anchor points together to the exact same position, where the strength of the pull is proportional to the current distance between the points (Unity Technologies 2019). By configuring the joints you can allow for different distances where the spring will or will not apply forces to the objects linked together by the joints.

An issue with joints in Unity is that they require a lot of tweaking to get a desired result, but the fact that they are so customizable also means there are many different end results, which in turn can be used to portray certain “feelings” or the mood of a character depending on how much a spring exerts forces on the joints.

A decision was taken to use an extra collider and rigidbody on top of the hierarchy for the character, to use for movement for gameplay and avoid affecting the other rigidbodies of the ragdoll with forces not meant for it. This means the character hips are connected to an invisible object acting as a marionette for the character (Figure 10). A more thorough explanation as to why this decision was taken will be explained in the balance subsection.
• Problem: Setup and tweaking takes a long time if having to be redone.
  o Solution: Make sure to have the final rig before starting.

• Problem: Joints have a lot of settings.
  o Solution: Save time by reading up on them thoroughly beforehand.

### 4.2 Animation Follow

A script was created to try and follow a target animation by the use of the joints on the character. This script was added to every joint of the character, and the script was assigned the corresponding limb of the invisible copy of the character, as this copy was playing animations as normal. Every frame, the joints on the playable ragdoll tries to match the target rotation of the animated character, but with the use of torque forces, to prevent outer forces. The joints then keep the character’s limbs together and limits their movement to avoid them rotating in directions not allowed.

![Figure 6](image)

This code is run in `LateUpdate()` as this function runs after Unity software’s internal animation update. It updates the target rotation for the joint to strive for. The variable `startingRotation` is the inverse of the target limb's local rotation at start of application. The complete script is available as Appendix 1.

As the group had a 3D animator for the project, animations were created with the purpose of blending with physics. The animator had never worked with physics blended animations and as such created animations trying to act out physics for the character. This resulted in a clash, in such a way that the animations had the character throwing his arms around, and with the script trying to mimic this with high follow values, the extra momentum given by the springs and joints made the character rotate out of control. The animations were then toned down to allow for the physics system in the game engine to handle the physics itself. This gave a better end result, but as the new animations ran with very few keyframes, the use of high follow values in the created script meant that the forces yet again would create undesired rotations of the character, as the movement was too “snappy”.

To circumvent this, values in the script were toned down, which had the unfortunate effect of making the character not having enough forces to follow the animation correctly. This means that the character felt more heavy than intended, making the movements “sluggish” instead.

It was realized that different animations would require different tweaking, and as such, to be able to control the animations and change follow values in runtime, a finite state machine was set up, where values could be changed whenever an animation transition would occur. How this works will be discussed more closely in the next subsection.

• Problem: Same animation follow values for every animation creates unnatural motion and makes it impossible to change “mood” of character.
  o Solution: Script allowing change of forces on limbs at runtime in a finite state machine.
• Problem: Free ragdoll mode on certain limbs create forces of their own resulting in lack of control of the character.
  
o Solution: A compromise of animation and ragdoll on different limbs to achieve a satisfactory result.

• Problem: Keyframe animations used as reference for the animation following affects procedural animation results. Fewer keyframes creates snappy movement due to a lot of force in a short duration of time.
  
o Solution: Create reference animation with the desired end result in mind, or increase values slowly over time in code.

4.3 Finite State Machine

The idea of a state machine is that a character is engaged in a particular kind of action at any given time. These actions can be different depending on the gameplay of a game, but they usually involve states such as jumping, moving, running, etc. By using these states with the script created for animation follows and controlling when the character transitions between states, values of muscle control can be changed at runtime. If, for example, the character is tipsy or tired, muscle values can be set in such a way that the springs controlling the joints of the limbs are more relaxed and less force is being exerted to try to keep them together.

To be able to alter specific limbs, a script was created to set new limb values for a state where needed (Figure 7). This script allows a more dynamic approach to the animations, as a state might want specific muscles to act differently, and not set all muscles to the exact same values. The script fetches the limbs specified to change and modifies them as the state is entered, but it also saves the old values for the limbs to set them back when exiting the state. The script also handles the limits of the joints rotation, to be able to restrict rotations when needed. With free motion, the joints are allowed to rotate in all directions to reach the target rotation. Limited motion means the joints are limited to the allowed angles set during the setup of the character, and locked motion does not allow any motion on the limbs at all. With the addition of this script, a state can change the feeling or mood of the character whenever needed. It does however require certain tweaking to find the correct feeling one is after, and to avoid snappy or sluggish movement in the animation follow.
Figure 7 Unity software’s inspector view of created script intended to change joints at runtime.

As the project evolved, so did the characters intended for the game. What was once two robots with good control and heavy feel was instead transformed into two characters with low amount of control of their limbs as it felt more appropriate for the game genre. This change meant that new animations were made, with even less specific animations, leaving the physics system to handle most of the movement of the characters limbs. As little time during the project was set aside for tweaking of animation follow values, the result was now jerky limb movement, yet again creating unwanted forces on the characters. A way to reduce this jerky movement would be to smooth out the animation curve, either by an animator or through code. This has however not been handled in this research.

With the change of characters, changes to animations follow was changed with them, to allow for more sloppy movement in the limbs. The run animation was changed and trimmed down to only move the legs of the character, and as such, the upper body of the character is all handled by the physics engine during the run state.

4.4 Character Balance

The initial idea was to create realistic balance since the game would be all about physics. The goal was to have a balance system that was physically-based, and most desirable would have been to have a more realistic balancing system without external forces like earlier research in
this field (Kenwright, Davison & Morgan 2013; Kenwright 2012; Kenwright 2013). Our first approach to get the character to stand on its own two feet was to have the hips obey the rule of always being above the character’s feet and having the chest perform corrective movements. A human is in a balanced state as long as the center of gravity is within the support area (Figure 8). If the center of mass tips over and ends up outside the support area, the character would start to lose its balance and counter measures would have to take place for the character to remain on its feet. With a balanced and unbalanced state the finite state machine could handle this and have algorithms search for a suitable location to place a balance-correcting step. This would attempt to extend the support area and regain balance. If this wouldn’t be enough, the character would fall.

Figure 8 Character balance. If center of gravity is within the support area (left), balance is maintained. If center of gravity deviates from support area (right), the character will start to topple over.

Real balance of a character is a difficult subject that requires knowledge and time. As time for further research in the matter was limited and the focus of this research is blending animations and physics, shortcuts were taken in regards to balance.

The first, temporary, solution to the above problem was to add pontoon-like colliders on the side of the hips, in the height of the feet (Figure 9). This kept the character standing upright, but having two long colliders sticking out from the character’s feet, results in other issues such as getting stuck in terrain when trying to walk up slopes or preventing a player from walking up to a wall. Hence it was a temporary solutions whilst working on the animation blending.
Another approach, that was recommended by a person at the company where the project was being created, involves using external forces to force an upright stance. With a constant force in the head upwards and a constant force downwards from the hips there is supposed to be a balance of forces that keeps the character in an upright equilibrium. If the legs are procedurally animated, forces can be added to force the feet back down after they have been raised up to take a step. The downside to this solution is that with the snappy animations used, and the forces created by those snappy animations, the character would start floating into the air and slowly fall towards the ground. This was not optimal since the project was aiming for a heavy feel, and was as such scrapped early on to focus on other solutions.

The approach that ended up in the artefact was using a control object and hinging the ragdoll to that (Figure 10). The control object is the parent object in the playable ragdoll’s hierarchy. It has a rigidbody, a capsule collider and the player script attached to it. The collider goes through the whole character like a spine. This is because the center of mass of a rigidbody is calculated based off of its colliders (Unity Technologies 2019). A hinge joint is attached to it, and the hips of the playable ragdoll is set as its connected body. The initial size of the control object was a small circle at the character’s feet, but that resulted in unrealistic behaviour with the character rotating around this object’s center of mass, which means the characters center of mass now was at its feet. The control object was then changed to go through the whole character as a spine, as mentioned earlier.

*Figure 9* Character rigged with pontoon-like *colliders* for balance.
Figure 10 Character connected to control object by the hips.

Since the character is now being moved through a control object, this means that the character itself is more of a marionette doll, trying to follow an animation and allowing other forces to act on the ragdoll for realistic behaviours of collisions on the character. The forces for movement of the character is however made through the control object, which is what is actually “standing” on the ground. In this sense, the ragdoll itself is floating just above the ground, so that the the character’s feet won’t hit the ground every time it plays the run animation. Without having the character elevated slightly above ground, the end result of the animation blending would be the character’s feet hitting the ground or being dragged behind the character.

As there is no need to reinvent the wheel when writing code, and the group decided on a 3rd person game with the character rotating with input relative to the camera, code for rotation in 3rd person games was found on the internet. With these lines of code, not only does the character rotate in x and z depending on the input, but it also keeps the character upright whenever there is input (Figure 11). This was later changed to only work if the character is grounded when using input, to prevent the character from snapping into an upright position whenever input was detected. This bonus of “cheat” balance was welcome, but as it only worked when there was input, another solution had to be used for when the character was standing still.

![Figure 11](image)

Figure 11 Code executed only when the character is running and considered grounded, keeping it upright. If there is no input from the player, the character falls with only this solution in place.

To keep the player balanced while standing still, a solution of locking the rotation of the control object whilst on the ground was used. Unity software has a built in function called raycasting. This function casts a ray in a direction, and if this ray hits an acceptable collider, a boolean value of true is returned. This was used to control if the character feet was touching the ground, and if it was and there was no input being detected, the rotation of the control object’s x and z
rotation would be frozen to prevent the character from falling over fully. This was later added to also being used whenever the character was in the jump-state and moving upwards in the y-axis, as frustrating gameplay where the character would constantly fall over after a jump had to be prevented. The rotations are however free whenever the character is falling downwards, to have some kind of state where the falling physics are playing out correct.

These solutions ended up being the end result of the project, as the gameplay felt less frustrating. There were however more ideas speculated and tested before coming to this conclusion, such as a “get up”-animation that the character would follow with assisting external forces pulling the character slightly upwards until in a balanced state. This suggestion were discarded in favor of more control to the player. Instead of having a state for when the character has lost balance and having it automatically stand up like in the game *Sumotori Dreams* (Gravitysensation 2007), the player would always be in full control and the act of standing up after falling down would be a skill the player gained through playing the game. Buttons to rotate the character forward and backwards were added, giving the player a more controlled way of getting up from the ground, but also resulting in a free gameplay mechanic of the character being able to do front flips and backflips.

Balance could also possibly have been solved by having specific limbs follow their target animation limb in world space instead of local space. If the character’s hips would do this, they would be forced in an upward position as the static animation always is upright. This could be added to the animation follow script but as the team was already content at the solutions given, this was never implemented.

- Problem: Self-balancing bipeds is an intricate task that involve several fields such as robotics, graphical simulation and biomechanics.
  - Solution: Cheat. Have a control object to balance instead of intricate ragdoll

- Problem: Character has unpredictable movement and falls over even when running.
  - Solution: Cheat balance by rotating control object upwards through input.

- Problem: Character falls over when standing still.
  - Solution: Lock or limit control object rotation in certain states.

### 4.5 Implementation Summary

The artefact was created as part of a bigger project consisting of 6 team members. One of these team members is a 3D artist, and as such, a model to work with was produced fairly fast. It was decided to use a technique with inner forces and torques for the the animation follow, inspired by earlier research (Kenwright, Davison & Morgan 2013; Kenwright 2012; Kenwright 2013). The bigger project had design decisions aimed towards silly and goofy movement of the characters, and this had to be taken into account when tweaking animation follow values.

Unity software’s function for creating ragdolls was used, but more work in the form of tweaking and adding new Unity components had to be done to the character to achieve wanted results. After character setup, traditional animations and a follow animation script were created. A script tells *joints* on the character to take aim towards a specific rotation, and the *joints* then strives to reach this rotation through the use of torques. Iterations on values for the animation follow script were done as the result either ended up being too stiff with jerky movement, or too sloppy with barely any movement at all.
A finite state machine was created to be able to set different animation follow values depending on which state the character is in. These values are then changed in runtime whenever a character changes animation, and specific limb values can be tweaked for animations, allowing different end results in animations.

As the project being worked on was supposed to be playable, effort had to be put into balancing of the character, as the project would not be playable with the character hovering in the air, only mimicking animations. Balance is hard to achieve and different solutions were tested to find one suitable for the game. One solution was to give the character pontoons to balance with, but this resulted in problem if the character tried to run up a slope or go close to a wall. It was also considered to have the hips of the character to try and always be above the character's feet, as a balanced state can be achieved if the center of gravity on a character is within its support area (Figure 8). Another approach of given the character constant external forces upwards in the head, and downwards in the legs, was tried. This resulted in the character floating and was scrapped for another approach. The solution for balance came in the form of an invisible control object, hinging the character in the air. This means the character is controlled through this invisible object while the character itself hovers just slightly over ground, mimicking animations and being interacted upon by the environment. The invisible object itself achieves balance through the use of different techniques such as forcing an upright stance while on the ground and moving through the use of code, but also by limiting the rotation in certain axes whenever the character is standing still or moving upwards.

The biggest challenge was to try and balance fumbly, goofy feel of the character and maintaining proper character control, as to not make the playing the game frustrating for the player. This was particularly evident in states where outer forces were used, such as jumping and belly flop, but also running, as this resulted in the character starting to rotate in unpredictable ways. The end result of the project being worked, was considered successful by the team in the form of a playable cooperative fumblecore game (Figure 12).

![Figure 12 Final look of the game.](image)
5 Evaluation/Utvärdering

5.1 Presentation of Study

5.1.1 Pilot Study

A pilot study was conducted to ensure that the final study could be executed without major errors or cause confusion for the participants. A pilot study assists in the collection of detailed enough answers for the problem statement (Denscombe 2014). This made it possible to stake out errors and implement improvements. Some of these improvements were to give clarification to questions that were confusing and unclear to the participants, such as providing examples to what is meant when asking about animations and interactions. The research consent and questions for the study in their final form can be viewed in Appendix B.

5.1.2 Final Study

The study needed to be in the form of an interactive playable demo in order to make it apparent that the animations are generated in real time and not pre-recorded. If the player gets immediate feedback to their interactions with the game world, it is more evident that the things happening on the screen are a direct response to the player’s actions.

The study consisted of two playable demos for the participants to try out. One of the demos had a character with keyframed animations and the other had blended procedural animations, both which took approximately the same amount of time to create. In each playable demo, the player was able to run, jump, give thumbs up, belly flop, put hands up, and grab things. Each play session lasted around 10 minutes divided into 5 minutes for each demo. The order in which the demos were played were randomised to avoid potential bias, e.g. in form of players having better recollection of the latest version played.

After the play session the players were asked some questions in an interview of a semi-structured format. The questions were open ended, but an interview guide was used to keep the participants on track. This format also enabled follow up questions and requests for clarification from the interviewee. The estimated time for the questions was 10 minutes. There were the two interviewers present - one handled the questions and the other took notes both during the play sessions and the interviews. The questions in the interviews revolved around how the player perceived the animations to give answer to the problem statement:

*How is a physics-based procedurally generated, animation following character, perceived by players?*

The assumption was that the participants would consider the procedurally generated animations of the character to look more humorous and goofy as well as be more fitting for the genre (fumblecore). It was hoped that this would entail that the participants would prefer the blended animation with the possible exception of players of competitive games or games where high precision of controls is needed. The fumblecore genre is not known for having precise controls, but the game produced aimed to provide sufficient control and precision, at the same time as maintaining fumbliness and humor. It was, though, predicted that the controls would not be good enough for some players and that they would perceive it as lacking. It is later in the report discussed if the results can be considered satisfactory enough for a fumblecore game.
5.1.3 Study Results

The study was conducted on 12 participants with varied ages and gaming experience. Before the playtest the participants were tasked with answering a few questions to gather some background information (Table 1) about their play habits.

- The age of the participants varied between 24-36 years.
- All 12 participants were male.
- Time spent on games per week varied between 3-50 hours.
- Favourite genres varied, with the most common being First-Person Shooter (FPS).
- 1 out of the 12 participants had heard of the fumblecore genre prior to the test.
- 9 participants had played a fumblecore game at some time.
- 4 participants considered themselves being competitive gamers.

In this study, the term competitive entails playing competitively in the way Caillois (2001) describes it - playing for competition’s sake. Enjoyment derives from winning over others or oneself and honing one’s skills.

Table 1 Background information of participants in the study.

<table>
<thead>
<tr>
<th>Preference</th>
<th>FC knowledge</th>
<th>Played FC</th>
<th>Favourite Genre</th>
<th>Age</th>
<th>Game Time</th>
<th>Competitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>RPG, FPS</td>
<td>36</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>MOBA</td>
<td>25</td>
<td>35</td>
<td>Yes</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>No</td>
<td>MMORPG</td>
<td>24</td>
<td>35</td>
<td>Yes</td>
</tr>
<tr>
<td>Blended</td>
<td>Yes</td>
<td>Yes</td>
<td>Simulation, RPG, FPS</td>
<td>30</td>
<td>15</td>
<td>No</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>Strategy</td>
<td>35</td>
<td>10</td>
<td>No</td>
</tr>
<tr>
<td>Keyframe</td>
<td>No</td>
<td>Yes</td>
<td>Competitive, FPS</td>
<td>25</td>
<td>21</td>
<td>Yes</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>RPG, Platformer, Manager</td>
<td>31</td>
<td>8</td>
<td>No</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>Hack’n’Slash, FPS, Fighting</td>
<td>30</td>
<td>21</td>
<td>No</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>Yes</td>
<td>MMORPG, RPG, FPS</td>
<td>32</td>
<td>5</td>
<td>No</td>
</tr>
<tr>
<td>Blended</td>
<td>No</td>
<td>No</td>
<td>FPS</td>
<td>29</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>Keyframe</td>
<td>No</td>
<td>No</td>
<td>MMORPG, MOBA</td>
<td>27</td>
<td>20</td>
<td>No</td>
</tr>
</tbody>
</table>

Regarding the character’s humoristic feel and “goofyness” - The different words used to describe the animations and characters during the interviews were gathered in a table (Table 2).

Table 2 Words gathered by participants of study, describing the character and animations in the different versions.

<table>
<thead>
<tr>
<th>Blended Animation</th>
<th>Blended Animation</th>
<th>Keyframe Animation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Happy</td>
<td>Cute</td>
<td>stiff</td>
</tr>
<tr>
<td>Goofy</td>
<td>Egger</td>
<td>Humoristic</td>
</tr>
<tr>
<td>Humoristic</td>
<td>Doesn’t understand</td>
<td>Happy</td>
</tr>
<tr>
<td>Flappy</td>
<td>Child-like</td>
<td>Goofy</td>
</tr>
<tr>
<td>Ragdoll</td>
<td>Dropped</td>
<td>Alive</td>
</tr>
<tr>
<td>Chaotic</td>
<td>Stiff</td>
<td>Believable</td>
</tr>
<tr>
<td>Brain dead</td>
<td>Funky</td>
<td>Classic</td>
</tr>
<tr>
<td>Flaccid</td>
<td>Interesting</td>
<td>Depressed</td>
</tr>
<tr>
<td>Jiggy</td>
<td>Jolly</td>
<td>Fumbly</td>
</tr>
<tr>
<td>Ridiculous</td>
<td>Playful</td>
<td>Jolly</td>
</tr>
<tr>
<td>Frisky</td>
<td>Charming</td>
<td>Boxy</td>
</tr>
<tr>
<td>Lively</td>
<td>Personality</td>
<td>Frisky</td>
</tr>
<tr>
<td>Fumbly</td>
<td></td>
<td>Mechanical</td>
</tr>
<tr>
<td>Merry</td>
<td></td>
<td>Neutral/No personality</td>
</tr>
<tr>
<td>Character</td>
<td></td>
<td>Not funny</td>
</tr>
<tr>
<td>Comical</td>
<td></td>
<td>Unfinished</td>
</tr>
<tr>
<td>Curious</td>
<td></td>
<td>Stoic</td>
</tr>
</tbody>
</table>
In regard to character control, 10 out of the 12 participants considered the controls of the character satisfactory enough. Although stating this, 4 out of the 10 mentioned that they at times wished for slightly more control, as some gameplay tasks felt hard to perform. The remaining 2 participants thought the controls were somewhat hard, but still good enough for the genre (fumblecore).

As for differences between versions, half of the participants considered the controls to be equally good in both versions played. Of the remaining 6, only 1 participant stood out, saying that the blended character was harder to control than the traditionally keyframed character. As such, 5 participants preferred the controls in the blended version.

The participants were also asked about differences in interactions between the character and environment in the versions, but no one reported noticing a difference.

When asked if the participants believed this type of blended physics-based animations could be used in other genres, 10 mentioned that it would not fit in the FPS genre, and 3 mentioned it could possibly fit in platformers. Out of 4 participants considering themselves being competitive, 2 considered it could work in the MOBA genre, which was the genre they played themselves. All 4 competitive players were amongst the 10 who mentioned that it would not fit in FPS games.

Finally, the interviews showed that 10 out of 12 preferred the blended animations over the traditionally keyframed ones. The 2 participants who preferred the keyframed version both considered themselves competitive and their favourite genre was FPS.

5.2 Analysis

The participants were all male and around the same age. This is probably due to the small sample of participants in the study. It would have been preferred to have a more diverse sample, more closely representing gamers overall.

Another effect of having few participants is that it is hard to draw any conclusions with confidence from the data collected, but a tendency can be seen when analysing the interview answers. 10 out of 12 participants preferred the blended animation, which seems to go in line with our previously mentioned assumption.

With a larger sample, possible correlations can start to be seen. Analysing variables such as the participants age, which demo they started playing, favourite genres, experiences with the fumblecore genre et cetera, could have revealed differences, similarities and possible explanations. With a larger sample, conclusions could have been drawn with more certainty.

The tendencies that could be seen though, was that those who had experience in the fumblecore genre gave answers that indicated a higher satisfaction of the character controls. It is possible that the fact that they had played this type of games before, means that they are used to the genres controls to be somewhat “bad”, and thus are more forgiving to it.

To find out if the problem statement can be answered, a more specific analysis of the study results is needed. The goal was to achieve both humour and control, to have a responsive ragdoll with unique and goofy animations to keep the player interested.

Below, the problem statement is split into the 3 different parts to be able to discern if it has been satisfactory answered as a whole.

- Flexible, fumbly and humoristic movements.
- Good user control.
• Feel and personality of the character, while still maintaining the unpredictable, unique animations.

5.2.1 Humoristic and Fumbly Feel
Procedural content does not always mix well with “soul” and control over personality. This was needed for our problem since a fumbly, humoristic character, was the goal. It was desired that there would still be some kind of control over its personality, even though it was procedurally generated. The compromise that was made was to attempt a blend of both keyframed animations, to have a certain character and feel as base, and procedural, physics-based animations to have a responsive character with unique and flexible animations.

To find out if the participants had thought the animations made the character look humoristic and goofy, the words used to describe the character in the demos were gathered from the interviews, classified and categorized (Figure 13). The word used for the top category in the hierarchy is humorous and the sub-category is goofy since it is synonymous to both silly and stupid, which is the feel the character was meant to have. The words that fit into this description were then put into their respective category.

**Figure 13** Categorized words from the interviews.

“The first version (keyframed) was stiff and had boring animations. The other version (blended) was more fun. The movements were smoother and more playful.” - Participant of the study

5.2.2 Satisfactory Character Control
Good control is often lacking in fumblecore, which is a result of physics playing its part on characters in these games. If physics-based blended animations are to be used in other genres, this is something that needs to be improved. As such, part of this project was to try and make sure players had good control of the character they played.

As mentioned earlier, 10 out of the 12 participants said that the blended animations would not work in the FPS genre. 2 out of these participants considered themselves competitive FPS players and they were also the 2 that preferred the demo with the keyframe animations. It is supposed that these players consider control to be one of the most crucial parts when playing competitively, and the FPS genre in particular, is a genre where randomness in any form
generally is avoided. As such, it is important that the character on the screen matches the input from the player. Discrepancies in this often means success or defeat in this genre.

It is however surprising how the other 2 competitive participants in the study pointed out that the animations could work in their favourite genre MOBA. A possible explanation to this is that it is easier to compare this project’s game, which has a third-person camera, to the first-person perspective that FPS games have, and therefore see the lack of control easier between these genres. It is hard to picture how characters would behave in a MOBA where the perspective is often a top down perspective. Most of the complaints on the controls in the playtest came from how the character behaved when jumping, which is also something you rarely see in MOBA games.

Participants who had played a fumblecore game before could be considered biased, as they are already used to the slight loss of control these games usually have. It can however also be considered valuable feedback, as they have knowledge of the genre.

An often-occurring answer though, was that the controls were considered acceptable for a fumblecore game, but some participants desired a bit more control at times, as some tasks in the game were hard to complete. With half the participants wanting more control at times or overall, this is most likely a good hint towards that the controls need more work and that the goal of good control over the character was not achieved in its full.

“When you failed in the first version (keyframed) you became frustrated. With the other version (blended) you didn't. You don't expect to have full control over the situation.” - Participant of the study

5.2.3 Interaction & Reaction – Unpredictable & Unique Animations

When asked about which demo was preferred, an expected answer was that the participants would mention that they preferred the blended animation because it gave better interactions with the physical world - That the animations gave that something extra with their unpredictability and uniqueness. No two animations are alike in this demo. Pre-recorded, keyframed animations have a hard time providing this since they are predictable in their nature. If there is supposed to be an animation for any given situation, it has to be created beforehand. With procedurally generated, physics-based animations, the animations are generated on the fly, as the situations occur in real time.

That the participants reported not noticing any difference in the interaction and reaction of the characters in the two demos is therefore surprising. An explanation might be how the questions were asked and that it wasn’t made clear what the questions actually meant. It is a possibility that the participants did not notice the difference if the character fell into a heap of jiggling limbs, or fell stiff as a rod when falling over, but that option is discarded as less likely. The difference in the interaction and reaction is deemed too significant to not be noticed at all. The interviewers are most likely at fault here and the erroneous assumption that the participants somehow would know all that the interviewers does after their research, and therefore not require any explanation to what interaction and reaction means in this case. If this is the reason, it shows how important a thorough pilot test can be to be able to collect qualitative data.

At further investigation most participants reported that they noticed a difference, but just didn’t think of it. It can seem strange to report not noticing or thinking about the differences, but still give the blended animation as preferred animation. The reasons given were for e.g. that it felt more fun and alive, which could mean that not only were more animations achieved, but also a personality/soul of the character.
“The character feels curious, happy and eager, like a child that doesn’t understand. It makes you want to go explore the game world yourself. The other one (keyframed animation) felt stiff and not fun at all. You didn’t really pay attention to him and didn’t feel interested in the game.” – Participant of the study

5.3 Conclusion

The table with the compiled words describing the character, shows that the goal of achieving a character personality can be considered to be reached.

The answers given regarding that the players wished for better control of the character, but keeping the flexible, humoristic movements they enjoyed in the blended animations, shows that the goal to combine goofiness and control wasn’t quite reached.

10 out of the 12 participants preferred the blended animation, despite not noticing any difference in interactions during gameplay, and gave the character attributes that were not credited to the animations, such as it feeling more “realistic”, “alive” and “credible”. This could indicate that the physics-based procedural animations added something more than was consciously observed - possibly providing that extra “juice” to the character animations. This can possibly be the end result of how an assisted technique (De Carli, Bevilacqua, Pozzer & D'Ornellas 2011) helps counter meaningless and repetitive content that Geijtenbeek & Pronost (2012) and Kenwright (2012) mentions is a problem with procedural generated content.

Even though the order the demos were played in were randomized, no real difference in data could be noticed by this, which might be that the sample is too small, and with more participants a possible difference could be noted. The biggest difference was the change of mood on some of the participants themselves when playing the blended version last, where some would turn around towards the interviewers and say “Oooh, this is so much better”, something that did not happen once when the order of the demos were reversed.

Some participants of the study mentioned that they could picture seeing this kind of animation in other game genres, but the majority believed it would not fit in FPS games. This was assumed to be because most FPS games tries to avoid randomness and the fact that the genre is competitive at heart, meaning anything that could impinge gameplay and decide outcome of success or defeat, should not be in the game.

The end result of the research shows that the following were achieved:

- The procedurally generated, physics-based character is fumbly and goofy.
- The controls were satisfactory for the fumblemonge genre, with room for improvement.
- The animations could fit in other genres, with a lot of work, possibly not in FPS.
- The elusive feel of something more than can be seen - “juice” - was achieved.
6 Concluding Remarks

6.1 Summary

Animations in the game industry are considered a bottleneck today, and a way to speed up the development in this area is by letting the computer procedurally generate the animations. This research has explored blending physics with keyframed animations in games, to create interactive and responsive characters. This way of animating characters in real time has lately been used in a blossoming game genre called fumblecore. By letting physics do part of animation in real time, unpredictable and unique animations can be created. The fumblecore genre aims to create humoristic and fumbly characters, which usually comes with loss of control because of physics constantly affecting said characters. This is generally accepted in fumblecore games, but if physics-based blended animations are to be used successfully in other genres, more character control might be desired. Another typical problem with procedurally generated content is that it often feels repetitive and meaningless, which is why an assisted technique was used, to counter this.

To find out how players perceive this kind of animation, if enough control can be maintained on a character who is constantly being affected by physics, and lastly, if character mood/feelings can be changed and noticed, a game was created. The game was typical for the fumblecore genre in aesthetics, but the typical character control of the genre was aimed to improve upon. Two versions of the game were tested on 12 participants. One of the versions had a traditional keyframed animation for the character, and the other had a physically blended version. After the gameplay test, a semi-structured interview was conducted on the participants, to find out if the animations and controls felt satisfactory, and if the character felt responsive and humoristic.

The study showed that participants preferred the blended animation, as they felt it was more humorous and unique than the keyframed one. Participants also mentioned that the procedural technique could find use in other genres, but a majority considered it to not have its place in the FPS genre.

Answers also showed that participants did not notice any difference between the two versions in form of interaction and responsiveness of the character, which the researchers assumed would be noticed. However, participants gave the blended character attributes that were not credited to the animations, such as feeling more “realistic”, “alive” and “credible”.

Even though the majority of participants felt the controls were good enough for the game created, and the genre (fumblecore) as a whole, answers also showed that most participants wanted more control when doing certain tasks in the game. This was interpreted as, that the character control sought after in the research, was not achieved.

6.2 Discussion

6.2.1 Social Gains

Procedural animations and techniques can help with the sustainability of a game company, both in form of economic and social resources. Since animations are almost always required in a game it is of importance to develop new, fast solutions and techniques that are easy to implement.

The result of this study show that the blended animation technique can produce a wider variation of animations within the same amount of time as a traditionally handmade
animation takes to create, and can be more sustainable in the long run. Given a few keyframe animations, a lively, humoristic character was produced. This technique does not fit in all genres as of today, but it does seem to help alleviate some of the pressure on animation artists. This goes in line with what Chen and Li (2016) states about procedurally generated animations and how they can help reduce or even replace time-consuming and repetitive tasks in animation.

Since crunch has been a problem in the game industry for a long time, and still is today (Wright 2018), it is an area worth looking into for labor-saving systems. Crunching means that employees are expected to put in more work hours in a project, the closer the deadline for said project is. Not only is the stress and fatigue from crunching detrimental to employee health, it is also bad for the company if their employees are mentally burned out and has to take time off of work. Lately, the problem of crunch has been raised again, and production teams in big titles such as Fortnite (Epic Games 2017) and Red Dead Redemption 2 (Rockstar Games 2018) have reported issues (Campbell 2019; D'Anastasio 2019; Schreier 2018). Although preemptive measures are taken to prevent crunch, such as reducing the workload with e.g. procedural techniques in different areas of game development, more can be done. The animation bottleneck is one of the areas to be improved. The research of this project takes its form in this problem.

With equal development time being put into a *keyframed animation* and a *physically-based blended animation* during this work’s project, the latter resulted in a more diverse and interesting amount of animations. This way, producing few animations can lead to a wide variation of animations, lessening the load on the animation artist. Reducing menial, tiring and repetitive tasks for an artist with computational power, enables utilization of the artist’s creativity in more suitable areas.

Even though the *blended animation* created in this research requires an animated character to work in the first place, it unlocks an unlimited potential of mood or personality change for the character in different states by tweaking muscle values. There are other procedural techniques that does not even require an animation to start with, reducing the load even further.

**6.2.2 Gender**

Reducing stressful production can also spread improvement to other areas of game development, such as diversity. On the topic of animation, there seems to be an issue of game studios leaving out a female character or proper animations with the argument that it is too costly to animate them and that it would mean double the amount of animations to be produced (Ray Corriea 2014). This has upset many, and several game developers have said that the logic in the statement does not hold and that it’s more a matter of bad planning to begin with than that women would be too hard to animate. Some mention that a female character should only take a day or two to animate instead of the workload some companies say. They admit that female characters don’t move identical to male characters, but to claim it would be double the animations is not correct (GameAnim 2014). Many animations should be reusable by just changing a handful of key animations to produce unique animations that give character (Farokhmanesh 2014).

No matter the reason to why the female animations were left out of the games, new technology should be able to help with this issue. Either by making it easier to produce quality female animations or to make it harder to hide behind potential excuses. With better methods of producing qualitative animations, this is something that could be improved.
Making efforts of dealing with this issue might help spread awareness and spark new efforts in the game development field overall where inclusion and so called “bro-culture” can cause problems for companies that wish to attract women. Some game development studios have current issues with company culture (D’Anastasio 2018), and with movements like #womenaretoohardtoanimate there seems to be an increasing movement for change.

Since the video game industry is the biggest media industry today (OppenheimerFunds Distributor, Inc 2018; Wijman 2018) it is important to keep up on these fronts. After protests and lawsuits (Takahashi 2018), efforts have been made to deal with inclusion and company culture in some studios (Kerr 2018) and perhaps these efforts can spread positive effects to the field of animation. Attracting new players and maintaining its current ones are of great import to a game development company. One part in this is making sure women feel included, in both the games and companies.

### 6.2.3 The Importance of Responsive Characters

Since there is often a need for the player to be able to interact with the environment in a realistic way, there is need for proper animations for these scenarios. The physics-based techniques can interact more natural to unanticipated scenarios which is important to be able to maintain the perceived realism and immersion (Geijtenbeek & Pronost 2012). This is why a responsive character is such an important field for continued research in the future. With physics-based methods we can achieve a character with movements that are interactive, responsive and non-repetitive (Kenwright 2012), giving control to the player.

Physics-based techniques does however have the issue of controllability. Using forces and torques to control the character leads to imprecise and slow controls (Geijtenbeek & Pronost 2012). This is an issue for many games and another field to find improvements in. The data driven method of motion fields (Lee, Wampler, Bernstein, Popovic & Popovic 2014) is one promising approach that takes us a step on the way with faster responding characters.

### 6.2.4 Tools to Affect Mood

Even if a responsive character is desired, there is often a need to be able to control the style, mood or personality of a character in games as well (Geijtenbeek & Pronost 2012). As Yannakakis & Togelius (2015) points out, without good ways for a designer to control the generated content, it risks being repetitive or of not good enough quality. It is therefore important that more work is done in this area.

### 6.2.5 Credibility of Results, and Risks

The credibility of the results is not great due to the small sample size. Considering how positive the blended animations were received, it does however give an indication towards its uses.

There’s also a risk of the researches having interpreted answers during interviews as confirmations of the assumption in the problem statement. There’s always a danger of attributing the opinions of others to what easiest comes to mind - usually what is closest to oneself. In this case it would be to take answers given in the interview as proving that the project has succeeded instead of finding other possible explanations to why this could be the case. Putting on blindfolds and seeing only what one wants to see makes you miss out on potential important information.
6.3 Future Work

6.3.1 Short Term
The procedural technique used in this research can save resources in form of preventing repetitive work for an artist, but it might also lead to repetitive tweaking if the details of the character's mood are to be perfected. Short term goals for the project could have been:

- Easier tools for changing mood of character.
- Improvements to the control of the character

6.3.2 Long Term
If the project with the blended procedural animations were to continue for a longer period of time, it could result in a complete game with the chosen technique. A game where two or more players control a robot each to try solving tasks at hand, at the same time staying on guard for friendly sabotage from their robot counterparts.

A future possibility could be to implement the animation techniques in another genre. For example, in a platforming game or in the MOBA genre that some respondents mentioned as options. With more effort put into the controls, it would be interesting to see if the technique could work in these genres with a goofy profile.

An often-occurring answer among the respondents was that the techniques would not fit in an FPS game. With improvements to character controls with precision and responsiveness, this could be an interesting area to implement the technique in, to find out how players would perceive it.

6.3.3 Broad Perspective
This, and previous work, has indicated there are gains to be made in using procedural animations. It is already used in games and movies, and as a future step, it the technique might also work in robotics, e.g. giving robots more personality.

In the same way as the game For Honor (Ubisoft Entertainment SA 2017) uses a library of animations for an AI to decide a suitable animation match (GameAnim 2016), a possibility could be to use AI and blending techniques to adapt current animation libraries of male animations to female animation, a problem mentioned in the discussion of this research. This could be a compromise until more suitable techniques are implemented.
References


Bossa Studios (2013), Surgeon Simulator, Video Game, London, United Kingdom.


Gravitysensation (2007), *Sumotori Dreams*, Video Game, s.l.


McKeand, K. (2017) “Nine years later, one feature in GTA4 has never been bettered”, https://www.eurogamer.net/articles/2017-02-12-one-thing-about-gta4-has-never-been-bettered [2019-02-14]


Rockstar Games (2013), Grand Theft Auto V, Video Game, New York City, NY, United States of America.

Rockstar Games (2018), Red Dead Redemption 2, Video Game, New York City, NY, United States of America.


Walt Disney Studios Motion Pictures (2016), *Moana*, Motion Picture, Burbank, CA, United States of America.

Wijman, T. (2018) “Mobile Revenues Account for More Than 50% of the Global Games Market as it Reaches $137.9 Billion in 2018”,

36


Young Horses (2014), Octodad: Deadliest Catch, Video Game, Chicago, IL, United States of America.

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnimationFollow : MonoBehaviour
{
    [SerializeField] private bool invert;
    [SerializeField] private float torqueForce;
    [SerializeField] private float angularDamping;
    [SerializeField] private float maxForce;
    [SerializeField] private float springForce;
    [SerializeField] private float springDamping;

    // Variables to reset original values. Used for State Machine
    private float origTorqueForce;
    private float origAngularDamping;
    private float origMaxForce;
    private float origSpringForce;
    private float origSpringDamping;
    ConfigurableJointMotion origAngularXMotion;
    ConfigurableJointMotion origAngularYMotion;
    ConfigurableJointMotion origAngularZMotion;

    private Vector3 targetVel = new Vector3(0f, 0f, 0f);
    [SerializeField] private Transform targetLimb;
    private JointDrive drive;
    private SoftJointLimitSpring spring;
    private ConfigurableJoint joint;
    private Quaternion startingRotation;

    void Start()
    {
        joint = gameObject.GetComponent<ConfigurableJoint>();
        // Store starting values
        origTorqueForce = torqueForce;
        origAngularDamping = angularDamping;
        origMaxForce = maxForce;
        origSpringForce = springForce;
        origSpringDamping = springDamping;
        origAngularXMotion = joint.angularXMotion;
        origAngularYMotion = joint.angularYMotion;
        origAngularZMotion = joint.angularZMotion;

        // Starting values. To be overridden in the different states
        drive.positionSpring = torqueForce;
        drive.positionDamper = angularDamping;
        drive.maximumForce = maxForce;
        spring.spring = springForce;
        spring.damper = springDamping;

        // Store the above values in the joint's spring
        joint.slerpDrive = drive;
        joint.linearLimitSpring = spring;
    }
// One time settings
joint.rotationDriveMode = RotationDriveMode.Slerp;
joint.projectionMode = JointProjectionMode.None;
joint.targetAngularVelocity = targetVel;
joint.configuredInWorldSpace = false;
joint.swapBodies = true;
startingRotation = Quaternion.Inverse(targetLimb.localRotation);
}

void LateUpdate()
{
    if (invert)
        joint.targetRotation = Quaternion.Inverse(targetLimb.localRotation * startingRotation);
    else
        joint.targetRotation = targetLimb.localRotation * startingRotation;
}

// Function used by State Machine to set new force values for limb muscle
public void OverrideMuscleForces(float torqueForce, float angularDamping, float maxForce, float springForce, float springDamping, ConfigurableJointMotion angularXMotion, ConfigurableJointMotion angularYMotion, ConfigurableJointMotion angularZMotion)
{
    drive.positionSpring = torqueForce;
    drive.positionDamper = angularDamping;
    drive.maximumForce = maxForce;
    spring.spring = springForce;
    spring.damper = springDamping;

    joint.angularXMotion = angularXMotion;
    joint.angularYMotion = angularYMotion;
    joint.angularZMotion = angularZMotion;
    joint.linearLimitSpring = spring;
    joint.slerpDrive = drive;
}

// Function used by State Machine to reset last known values for limb muscle
public void ResetMuscleValues()
{
    drive.positionSpring = origTorqueForce;
    drive.positionDamper = origAngularDamping;
    drive.maximumForce = origMaxForce;
    spring.spring = origSpringForce;
    spring.damper = origSpringDamping;

    joint.angularXMotion = origAngularXMotion;
    joint.angularYMotion = origAngularYMotion;
    joint.angularZMotion = origAngularZMotion;
    joint.linearLimitSpring = spring;
    joint.slerpDrive = drive;
}
Appendix B - Consent Form and Interview Questions

Informed consent form for participants in the research of procedural animations, titled “Physically-Based Animation Follow”.

Name of Investigators: Henrik Lennartsson and Lina Wijkander
Contact Information: a16henle@student.his.se
                      a15linwi@student.his.se
Name of Organization: University of Skövde
Name of Project: Physically-Based Animation Follow

This Informed Consent Form has three parts:
• Information Sheet (to share information about the study with you)
• Certificate of Consent (for signatures if you choose to participate)
• Background Questions (to gather information about player background)

You will be given a copy of the full Informed Consent Form

Part I: Information Sheet

Introduction
We are two students at the University of Skövde doing research on procedural animation, as animations are considered a bottleneck in the gaming industry today. We are going to give you information and invite you to be part of this research. You do not have to decide today whether or not you will participate in the research. Before you decide, you can talk to anyone you feel comfortable with about the research. This consent form may contain words that you do not understand. Please stop to ask us any questions if there is something you do not understand and we will take the time to explain. If you have questions later, you can ask them of any of us.

Purpose of the research
This research aims to look into procedurally creating animations, which means that the computer does a lot of the work on the animations, to be able to reduce the cost and time it takes to produce handmade animations. We are investigating one of the many different ways to procedurally produce animations and we believe that your opinion about our work can help us decide if this is a valid approach regarding this subject.

Type of Research Intervention and Duration
You will be tasked with playing two different demos of a game, for 5 minutes each. This will be followed up with an interview, asking your opinions regarding the demos played. The interview will take approximately 10 minutes to complete. The way you play the game will not
be evaluated, as there is no right or wrong way to play the game. The total time estimated to participate in this research is 20-25 minutes.

Voluntary Participation and Right to Refuse or Withdraw
Your participation in this research is entirely voluntary. It is your choice whether to participate or not. You can choose to withdraw your consent at any time without consequence.

Procedures
Before playing the demos, you will be handed a paper with a few questions regarding your player background. After answering these questions the playtest will begin. You will be given a controller and we will go through the control scheme of the game with you. You will play the game by yourself, but we will be in the same room and available for questions if any should arise. After playing the demos, an interview will be held, to find out how you perceived the animations in the game. So try to pay attention to how the animations look and how it feels to play and interact with the world. The interview will be an open, semi-structured one and we encourage you to speak your mind.

Confidentiality
The data collected will be kept private, and you will not be asked to share any sensitive information. You will be assigned a number, which will only be used to enable withdrawal of consent, if so needed.

Sharing the Results
The data collected will be available in our bachelor thesis which is planned to be published at diva-portal.org.
Part II: Certificate of Consent

I have been invited to participate in research about procedural animations, titled “Physically-Based Animation Follow”. I have read the foregoing information and I have had the opportunity to ask questions about it. I consent voluntarily to be a participant in this study.

Date - Year/month/day

Name of Participant

Signature of Participant
Part III: Background Questions

Age
_______ □ Rather not say

Sex
□ Male □ Female □ Other □ Rather not say

How many hours do estimate you play per week?
__________________________________________________

Favourite genres in games?
__________________________________________________

Do you consider yourself a competitive player?
□ Yes □ No

Have you heard of the fumblecore genre?
□ Yes □ No

Have you played any fumblecore games (e.g. Human: Fall Flat, Octodad: Dadliest Catch, Gang Beasts, Surgeon Simulator)?
□ Yes □ No

What games do you play currently?
__________________________________________________
__________________________________________________
__________________________________________________
__________________________________________________
Interview Material - For Interviewers Only

Free answers are encouraged. Think aloud. There are no right or wrong answers. If we want to find out something more specific we try to clarify. Survey relies on participants opinions. Try to avoid leading the interview.

1. What did you think of the animations in the different versions?
2. Which of the animations did you prefer, and why?
3. How did you feel in regards of character control in the different versions?
4. Which version did you prefer in terms of controls, and why?
5. What do you think of the mood/feel/personality of the character?
6. (Mention which version was the blended one) Do you believe this kind of animation could work in other games, and if so, what type of games/genres?

Clarification Questions

- How did you feel in regard to the interaction and response of the character? (If it ran into things or got hit etc., how do you remember the response was?)
  - Achieved responsive ragdoll?

- How would you describe the characters feelings? What did you feel when looking at it run around and interact?
  - Achieved humoristic character?

- Why do you prefer said animation/character?
  - Why, if, is one character more interesting than the other?

- Did the animations and controls fit for the game/genre?

Did animations and controls fit the genre?