An Evaluation of Active Ragdoll Systems

COSE60654 GAMES TECHNOLOGY RESEARCH PROJECT

Harvey Milner
SUPERVISOR: BEN WILLIAMS
ASSESSOR: PETER COOPER

Contents

List of Figures

Abstract

For a long time, traditional 3D animation for video game characters has been a cumbersome process for developers and artists working on video games. Even with all the time and resources invested in ensuring that character animations cover all player actions, it is only possible to cover some eventual actions. The introduction of physics-based characters aims to fix these issues through a mixture of traditional animations and physics bodies. This study takes a look and evaluates multiple active ragdoll solutions in order to determine which ones are most suitable for different gameplay situations that may appear within a game, and to give developers a more informed looked at active ragdoll solutions that could be built into projects to reduce some of the issues that are presented with non-physics-based characters.

Introduction

Traditional 3D character animation has revolutionized immersion across tv and film industries but mainly in the video game field. Innovations in this field have allowed for detailed 3d characters with realistic human movements but at a large development cost due to the time and manpower needed in order to produce these animations. Classical 3D games used many lessons learned from tv and film where a 3D model of a character is made and then the bones are rigged, and animations are created through keyframes giving the character movement. This presented a unique challenge for games due to the many unpredictable interactions the player can have unlike with film where everything the viewer will ever see is highly curated, this led to games lacking in physical interactions with the surrounding world.

Early Ragdolls were created to solve key issues games had with physics such as when a character dies or falls over, early games would not consider the world and objects around them instead would just have a pre-set animation that would look out of place. Due to hardware limitations ragdolls where only turned on when needed and traditional animations would be used in the place of ragdoll animations the rest of the time, this was done in order to display these kinds of physics interactions as instead of having a premade animation the characters would directly react to the forces being exerted on them.

Ragdoll animations work differently to the traditional method of rigging and keyframe animations, instead of the character being rigged it is given a realistic bone structure with different weights as well as a centre of mass that are all effected by the games physics. All animations are created during runtime by directly reacting to the forces that the character is experiencing at the time. Many games have iterated on the basic concept of ragdolls in order to create a more realistic reaction to forces as without fine tuning a ragdoll will fall to the ground, this works well for certain situations such as the

previously discussed falling over and death animations as in these situations a character should fall to the ground while also reacting to the objects around them.

With the evolution of hardware early games character details became more realistic as these hardware advances allowed for higher quality textures and polygon counts. Although the visual elements of characters achieved a higher quality, characters where still using the traditional animation techniques which started to look more out of place not being able to interact with physics objects in the world such as when a character is opening a door. This is where the development of active ragdoll started. Active ragdoll works in a similar way to ragdoll where all the bones and joints of the character react to physics, the main difference is the ragdoll elements are not turned on and off as needed as active ragdolls react to all physics being exerted on them. The main issue with using active ragdolls over traditional character animation is the hardware cost as there is a lot of calculations needed in order to process the physics interactions on the characters, along with this hardware cost there is a lot of work needed for these active ragdolls to look and preform as realistically as keyframed animation.

Although active ragdolls have a high computing cost and large initial set up in order to look realistic this has not stopped many different developers from seeing the potential in them. Many developers such as EA have chosen to build active ragdoll technologies into the core of their engine in order to produce realistic interactions on top of their existing animations for all of their newest sports titles starting with the FIFA franchise. Due to many developers with large resources preforming initial development in order to increase performance and realism for active ragdolls this has led to it becoming widely adopted across many games as well as it now being more accessible for smaller indie developers to implement due to engines such as unity building their own solutions to aid developers implement active ragdoll. With open accessible engines like unity adding support for active ragdoll it has allowed for more unique titles to come out using active ragdoll such as gang beasts and fall guys, both of these games don't just use active ragdoll to add to their game play but both use the physics interactions as their core gameplay with both having gained a large amount of success from this unique style of gameplay as they are not aiming for realism but instead just play with the idea of the player being able to directly interact and be affected by everything in the world.

With the rise of active ragdoll technology in many current games it brings along many questions such as what the best implementation of active ragdoll is to use when developing a game and although the technology has become widely adopted and documented there isn't any relevant research into this question published over the past 15 years (Glimberg Stefan, 2007). This highlights a clear lack of research within this area of work, which may be problematic for developers implementing active

ragdoll systems into their games. For example, game developers may be basing their technical design and implementation of active ragdoll systems on a volume of work from early work in this area.

The evaluation of multiple implementations of active ragdoll systems within different gameplay situations will form the basis of this report. Such an evaluation could have significant benefits for game developers, especially those targeting implementations of active ragdoll systems, for example improvements in performance would not be considered when looking at older research due to later improvements within this subject area, along with the performance considerations the developer would benefit from breaking down which implementations of active ragdoll preform best in gameplay scenarios this includes which ragdoll system provides the most believability from its animations.

With this in mind, the main objective of this thesis will be to evaluate different implementations of active ragdoll systems to determine if there is a standard that both fits specific gameplay situations and environmental effects while also being both performant and upholding realism that would be found with traditional keyframe animation. The purpose of this work is to provide initial experimentation through a pilot study; to observe interesting effects which could inspire later work in this area. This is supported by the evident lack of exploration of the research domain, notably, the comparison of active ragdoll systems to that of a non-physics-based character. While looking at active ragdolls in specific situations it will also be considered if it is a technology that is even worth using outside of the scope of a game built with it as a core mechanic such as gang beasts.

Literature Review

Development of Ragdoll Systems

Some of the initial advancements in active ragdoll technology came from custom engines developed by studios inhouse such as the frost bite engine developed by EA. Basic Ragdolls where first used in skate for character deformation when physics interactions took place this technology was developed by EA Tech using the EA physics which would later be implemented into the frost bite engine when merging the EA tech team into the frost bite team (Ahmad, 2013). Due to EA developing many sports titles, active ragdolls were developed and implemented into frostbite by EA physics to allow for physics interactions between players in games such as FIFA, madden and fight night that could not be animated in a traditional way (GDC, 2018) . Early in development many different methods of active ragdolls were tested in order to give the best results for this kind of implementation. The resulting solution uses animation bones with physics layers added on top which is used to deform the animation bones from the given animation. Physics joints are also implemented to give a normal human range of motion for the given character. Using this method of active ragdoll torque is added to the bones to produce movement. Although this works in order to produce an animation a root bone is needed to allow for animations with respect to the world space this bone attaches from the hip to the world. One of the ways of optimising performance for this implementation is lowering the amount of physics bodies on the character by reducing the amount of physics bodies it adds the structural integrity of the character but reduces the range of motion of which the character has in the case of FIFA this was overcome by procedurally changing the position of the spine joints during run time to adjust to the physics operations.

Along with studios building their own inhouse active ragdoll technologies many public engines such as unity and unreal have started to develop their own solutions for active ragdoll making it more accessible for many developers that wouldn't usually have the resources to dedicate to developing active ragdoll technology. Both unity and unreal make use of importing a premade 3D model that has had its bones rigged in external software with unity making use of its skinned meshes technology to allow for the deformation of the character whereas unreal uses an inbuilt physical animation component (Binh Huy Le, 2014). The unity engine makes it easy for a developer to quickly have a ragdoll ready to use within a game, the process starts with a character being created and rigged in external modelling software and importing it into the project files once imported the user needs to deselect the mesh colliders then map out the bones from the rigging process to the ragdoll creation tool once this is complete the developer should be able to enter play mode and have a completely limp character which is the base for creating an active ragdoll (Unity, 2023). Unreal engine has a

different set up where a physical animation component needs to be added to the character blueprint from there the physical animation and its properties can be edited through the characters blueprint after selecting the root bone for the character (Unreal, 2023). Both of these methods give a developer the basics of an active ragdoll with a quick set up time although the character will need to be edited in order to make them more believable within the context of the game, they are in.

Through AI driven reinforcement learning, realistic animations that also have world interactions can be produced. This reinforcement model works like others by rewarding the AI for staying close to the target animation pose and taking points for failures such as falling over (Ariel Kwiatkowski, 2022). Over multiple generations of AI this leads to the decisions being made that allow for the character to follow an animation while also reacting accordingly to the physics operations happening in the world around them. Motion matching is a method more commonly used with in the robots space but can also be used when dealing with active ragdolls in video games, motion matching is used by training an AI active ragdoll character through reinforcement learning to follow another non ragdoll character (GDC, 2020). It starts with having a base animated character with no physics interactions like those present in most games this will be used as an ideal motion for the active ragdoll, from this multiple generations of the AI will be run in order to improve the performance. The AI will receive feedback based on the world and game rules receiving a positive score when preforming as intended and removing score when failing through a few generations this will produce movements the same as the non-physics-based character if set up correctly. This implementation can have some draw backs when not set up correctly as if the reward feedback is not distributed correctly then it can lead to less desirable animations, this can be detrimental due to the high time investment needed in order to run multiple generations of the character. Using motion matching along with producing the original animations from motion capture allow for quick results as from minimal animations the physics body can fill in gaps in animations in order to stay on track with the animation goal, this solution also means that animators don’t need to produce animations for all eventual outcomes in the game as the active ragdoll can physically react with the world.

When producing active ragdolls there are many variables that can be changed around and included in order to produce the desired results for the specific gameplay situation. Bones are the main component when building an active ragdoll system, these bones allow for the realistic movement of the character but in order to achieve this realism the correct number of bones and structure is required (Joe Booth, 2020). Due to the nature of each bone having its own independent structure it massively effects the way in which the active ragdoll will perform, each bone has its own position and size relative to the character being produced such as in a case of a humanoid character they would follow the same bone composition as a human in order to produce a movement result that matches

a human one, these different variations in the bones need to be tuned in line with the gameplay scenario in order to produce the best results. When working within a video game this adds a layer of complexity as this can lower gameplay performance due to the extra calculations needed to simulate the bones this is where a level of variation can be added to the creation of an active ragdoll, by reducing the amount of less important bones the rigidity of the character can be improved due to gravity effecting the character less as well as the number of calculations needed. Joints are used as the hinge that connects each of the bones with each other and are required to move from a simple ragdoll into a complex active ragdoll character. Bones can be held together through a number of connection points ranging from one to three with a higher number of connections giving greater structure between each bone. By the use of joints, it allows a developer to set constraints for the character such as not being able to move its arm in a 360 degree rotation that would not be physically possible for a human to perform, using joints also allows for a greater level of realism as due to the mixture of a correct bone structure and joints a developer can use inverse kinematics to determine the correct position for bones that are attached to one that has moved to a new position such as in the case of a hand deforming to reach a door the rest of the arm will move along with the hand in a humanlike manor (Kenwright, n.d.). Inverse kinematics takes the end point of a bone and works backwards through connected joints to find the correct orientation that matches with the end result based on the constraints given in the joints. The joints provide a huge impact on the level of realism that the resulting active ragdoll will provide as a character given fewer joint constraints will perform more loosely and in a cartoon like fashion whereas the opposite will happen when adding a greater number of constraints using both more and less constraints are viable options when creating an active ragdoll, the exact implementation comes down to the game that the active ragdoll is being designed for.

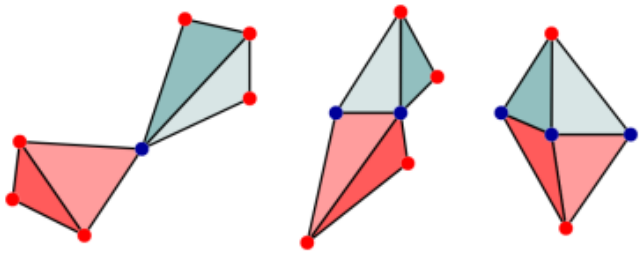


Figure 9: Three ways to model joints using particles. The first is a ball joint where the two boxes share one particle. Second is a hinge joint where two particles are shared. The last shows three shared particles which yield one rigid object.

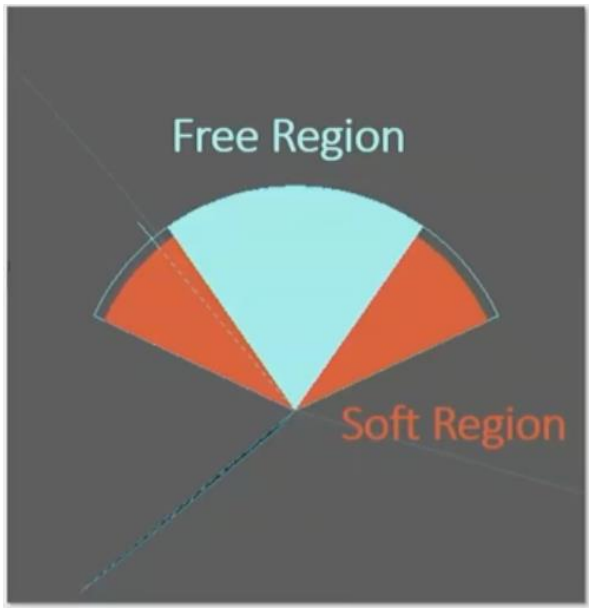
1 Joint Particles (Glimberg Stefan, 2007)

Physics Based Systems

Although using a physics-based approach to animation may present many down sides when it comes to gameplay performance and immersion for the player it allows for many interactions that are just not possible without using them. Performance issues can largely be reduced through turning off active ragdolls on characters when they are not in use and blending their current animation back to a standard one this lowers the overall physics calculations that need to be made, the number of joints that are used can also be reduced to increase performance this can result in lowered realism if not done correctly. Once performance issues are worked out active ragdolls can add lots to a game while also making the animation workflow quicker and adding new tools to an animator that would not usually be present when animating with keyframes alone. The main objective of implementing an active ragdoll to a game is in order to improve the way that the character interacts with the world and more importantly how the world effects the character as well as helping to fill in unforeseen gaps that animators would not be able to account for when creating animations. One gameplay issue that has troubled developers and designers for many years prior to active ragdoll is that of a character opening a door within the game world, this is due to many positions that a player can interact with a door from making it extremely difficult to animate. Prior to the use of active ragdoll many animators and developers would use basic arm movements to convey the action of opening a door without the character physically interacting with it or taking the player to a pre rendered cutscene of the character opening the door making it look as though the character had teleported from their previous position to that of the door these actions for many games broke immersion for the player, through the use of active ragdoll and inverse kinematics this issue can be solved as the character can deform its arm to the door handle position no matter the starting position and rotation as long as it is within human range of the character inverse kinematics can then be used to calculate the position and rotations of the connected bones this can be used alongside a motion matching system to overlay animation in order for an animator to fine tune the details of how the animation plays out. Physics based animation along with becoming a method in improving realism in video games is also branching out into its own genre of game where developers have lowered the joint constraints and used the physics interactions as a gameplay mechanic in games such as gang beasts and human fall flat these games use a less realistic active ragdoll but through the use of unique level design manage to create a game where the fun is based around controlling a character that still preforms like a ragdoll making these types of games only possible through the use of active ragdoll.

When implementing physical animation into a game, the world physics clearly play a huge role in the way that an active ragdoll will perform when looking at its believability. Even though the world physics is the main factor acting on the active ragdoll the different constraints of each join can be altered in

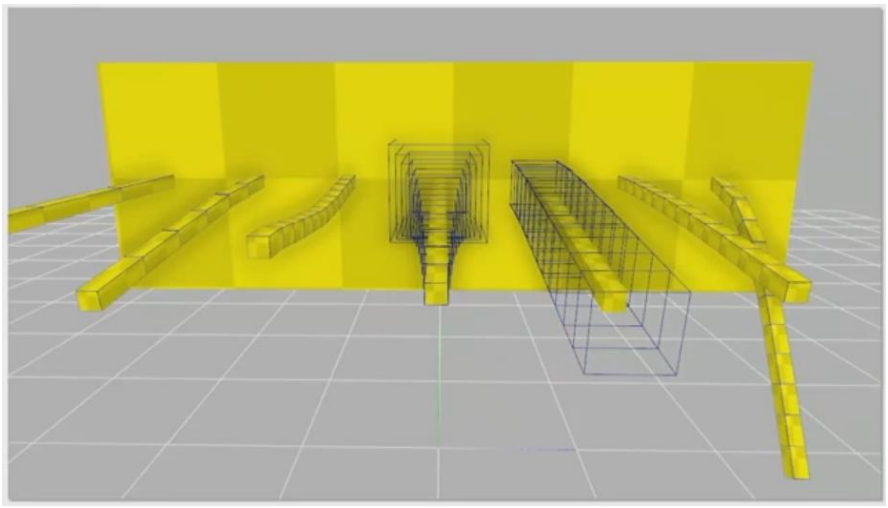
order to have the ragdoll perform as intended. Each joint on the character effects the range of motions that the character is able to perform meaning that even if the physics acting on the character is forcing it to act in an unintended way each joint can be tuned to ensure that the character acts as envisioned this also works with the bone structure of the character as they can be tuned in a way that gives the character greater strength in specific sections such as in the spine making the character less likely to fold in on itself. Extra realism can be added to active ragdoll joints to allow for a more accurate human range of movement this is achieved through adding soft regions to joints, when a character moves to a soft region they will use a spring to move back into the free region this allows a character to briefly move into a position that is possible for it to move into but only for a brief amount of time such as quickly jerking a character arm in order to throw a ball (French, 2012). These separate regions of each joint can be altered as needed for different types of implementations as a developer may want a character to have a greater range of motions than would usually be possible in this case the free region would be extended and the soft region made shorter.



2 Example of free and soft regions (GDC, 2018)

Stability of active ragdoll characters is one of the greatest problems when implementing the system into a game. Due to the nature of everything in the world effecting the character and its movement it means that a character can easily be knocked over or knocked off course when performing an action in most cases this is an unwanted result and something that would need to be fixed in order to ship the game (Sujeong Kim, 2013). Stability within an active ragdoll can be achieved through a few methods that centre around controlling the ridged bodies of the character, the ridged bodies present on the character can be altered in two main ways in order to improve the stability one method is

produced by lowering the mass of the ridged bodies along the bones such as in an arm the hand would have a much lower mass than the shoulder which connects to the body through using this method a greater force would need to be applied to the end of a bone to have the same effect that it would have if all of the ridged bodies had the same mass this however does come with the downside of being computationally expensive due to the amount of ridged bodies needed for a character (Pawel Wrotek, 2006). Another method is to keep the same mass of each ridged body but lower the total count of them being used on the character through using less ridged bodies it improves performance as less physics calculations are required in order to find the correct movements this method also improves the stability through the nature of there being less area for the physics operations to act on and can work well for many implementations, challenges with this approach appear when looking at bones such as the spine where stability is needed but a normal range of motion is required in some cases this issue has been solved through procedurally moving the joints within the spine to allow for movements along the spine but only when needed.



3 Effects of altering the number of ridged bodies being used. (GDC, 2018)

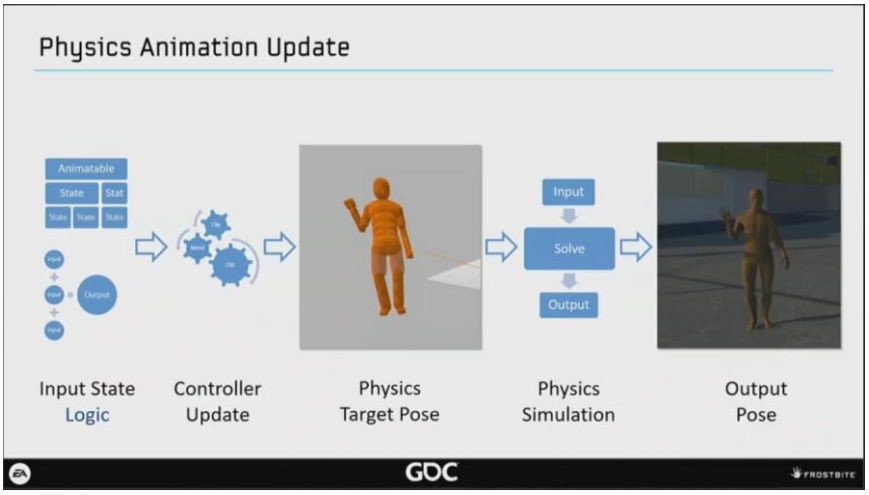
Alternative Methods

When looking at different implementations of active ragdoll systems there is a few to choose from such as a regular active ragdoll driven by torques but these can be expanded on in order to produce other variants like AI driven systems and motion capture active ragdolls (GDC, 2020). A basic implementation of an active ragdoll allows for the developer to manipulate the joints of the character in order to produce motion, to manipulate each joint a torque can be added to the current rotation to allow for the character to achieve a desired pose or movement these joint movements take in a torque vector which tells the character which magnitude and axis to move the joint along. The mass of the joints can be adjusted as needed in order to control the strength of movement in which a larger mass leads to stronger torque movement the main factor to keep in mind when producing these movements is to use as small of a torque as possible due to stronger torques increasing the likelihood of the active ragdoll failing.

AI driven active ragdoll systems build off of the basic torque movements but differ in the way that they are directly implemented. An AI driven system can take a few approaches one of the more popular ones being through reinforcement learning where an AI is trained over multiple generations to follow a predetermined animation these can be produced using neural networks in order to determine the best animations to use and using the torque inputs can change the characters joint rotations to match the animations provided, the AI will receive rewards the closer they are to matching an animation and will have the rewards taken away for failure allowing it to make better decisions for future interactions.

Motion matching is a technique that can be traditionally used without the use of an active ragdoll (GDC, 2020). Motion matching takes a large data base of premade animations usually sourced from motion capture from this data base a relevant animation is chosen depending on the player input in order to play the most suitable animation, this allows for realistic and reactive animations to be played in many different gameplay scenarios. This technique allows for an immersive experience for the player but falls short in a few different areas mainly being the lack of physics and the performance hit from the implementation of this system (Victor B. Zordan, 2005). By altering the system to take advantage of active ragdoll this does not offer a solution to the issue of performance but it does allow for each of the animations being selected to perform physics interactions with the world around them this adds another layer to the immersion offered by such a system as it allows for interactions such as the animation adapting to the player running while also altering the animation if say they hit another character it would deform the player character in response while also checking to see what the next best animation would be to recover from this deformation. When looking at the performance of the system the use of active ragdoll doesn't improve it, but the use of another active ragdoll technique

does, by using AI reinforcement learning with the system it allows the developer to train the character to perform each animation this cuts down on the performance as with the use of AI it removes the need to store as much data for the animations in the memory in turn freeing up the memory for other elements of the game.



4 An example of an update loop for a physical animation (GDC, 2018)

Research Methodologies

Experimental Methodology

The goal of the research is to define the most suitable active ragdoll systems for different gameplay conditions. In order to test the ragdoll systems multiple different tests will simulate different environments that reflect real world gameplay physics interactions. Each ragdoll will run through the test environments multiple times and the individual joint positions will be recorded in 0.2 second intervals in order to retain accuracy without bloating the results. A non-active ragdoll character identical to other characters will also be ran through the same tests the results from this character will then be used to compare results from the active ragdolls. Large deviations from the control will be recorded as less performant as the non-ragdoll character will be seen as the ideal movement for the ragdoll characters.

Three test environments will be conducted each of which test different physical interactions for the active ragdolls. The basic test condition will be moving the ragdoll across rough terrain as this will be a challenge in most games, this test condition will test the ragdolls ability to rebalance itself across a terrain of different elevations. The physical water test will test each ragdolls performance while swimming this test will physically interact with all of the ragdolls joints. The rope bridge environment will test the ragdolls ability to balance itself while moving on a constantly moving platform due to each panel of the rope bridge moving due to the force being applied from the ragdoll, it will need to have a reaction to this change in position in order to stay stable.

In order to collect the positions for the ragdoll characters a csv file will be written during the course of the test. The csv writer will take the position of key bones on a specific ragdoll character and output the position in local space to the csv. Once the test reaches the target frame the scene will be reloaded in order to collect the positions for 3 runs of the test. The csv writers will be set as an object in the hierarchy and three will be need in order to account for each ragdoll being tested.

Setting up the rough terrain condition requires setting up a basic unity scene and placing each active ragdoll in the same position in world space on a 3D plane. The plane object is set up with evenly distributed 3D cube and sphere objects to act as the obstacles for the ragdoll. With the ragdolls and ground plane in place the test is run by setting each ragdoll to be active individually along with the corresponding output object.

Unity hinge joints are implemented for the construction of the rope bridge for test 2 this is due to their interactions with unity's inbuilt ridged bodies as well as hinge joints being widely used by many developers for this situation. Along with the bridges construction a start and end platform are added to the scene for the ragdoll to ensure that each ragdoll starts at the same initial position.

The third test condition requires the external Water Float plugin from the unity asset store. Once importing this plugin, the float scrip is added to each ragdoll and the parameters for movement are set up the same for each based of the included documentation.

Technical Implementation

The unity engine has been selected in order to conduct the research within due to the wide range of active ragdoll systems that are available to be implemented into the engine as well as the wide adoption of the engine meaning the results can be directly acted upon by a wide range of developers as well as being produced in the universal render pipeline. Each test condition requires a different technical implementation in order to test the active ragdoll systems most of which can be built from standard unity elements. In order to produce a realistic and consistent water motion for the swimming test the water float unity package created by Pathiral (Pathiral, 2020) will be implemented as this package uses a standard function of creating physical waves, the packages uses sinusoidal waves (Phelps, n.d.) which can be overlapped with each other at different intervals in order to create some realistic water effects. Each of the other test environments will be created using in built unity technologies, the rope bridge will be produced using the hinge joint (Unity Technologies, 2021) built into the unity engine due to the nature of it being built into the engine it can be a widely adopted tool.

The project has a large part of the technical implementation from each active ragdoll characters and collection of data from each, the data from the ragdolls is collected from the position of each major joint every 0.2 seconds and is outputted to a csv file all physical calculations are made within unity's fixed update loop in order to achieve a fixed time step in order to produce consistent physics interactions. Each of the animations for the ragdolls will be identical for each and will be sourced from mixamo this gives consistency between the ragdolls and the control character as well as simulating a real-world use case as opposed to using procedurally animated characters.

The control character is a basic identical 3D model to the ragdoll character with a ridged body and capsule collider along with a basic script to move the ridged body forward at the same speed as the ragdolls and an animation controller that is shared across all characters in the test.

The projects first and most basic ragdoll will be the unity engines inbuilt active ragdoll character this was chosen as the most accessible option for most developers. The unity ragdoll is set up using configurable joints and ridged bodies on each bone and connecting them to the root bone in this case it's the ragdolls spine a small script is added to control the position and rotation of the child bones.

The final ragdoll implementation is from the unity package Root Motion Puppet Master (RootMotion, 2022) this is used to set up the 3D character as an active ragdoll that is propped up from the root bone

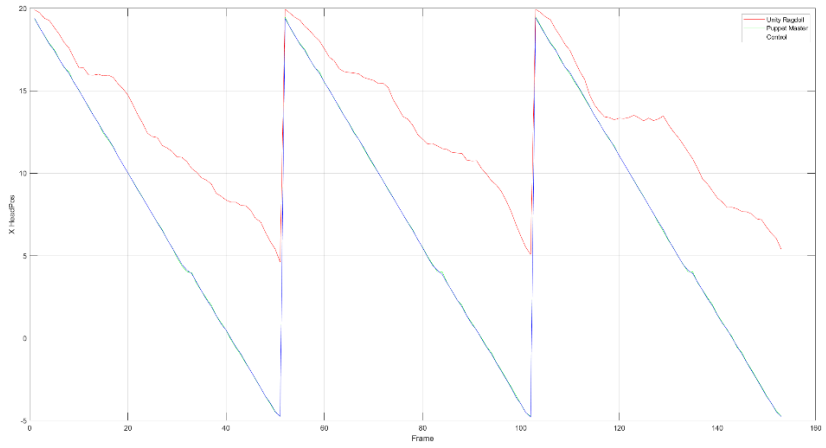
of the character. This package splits the character into two separate child characters one that controls the animations while the other controls the physics interaction acting on the character. Each of the settings will be kept in line with the documentation for the package in order to produce an easily reproducible result for other developers.

Results and Findings

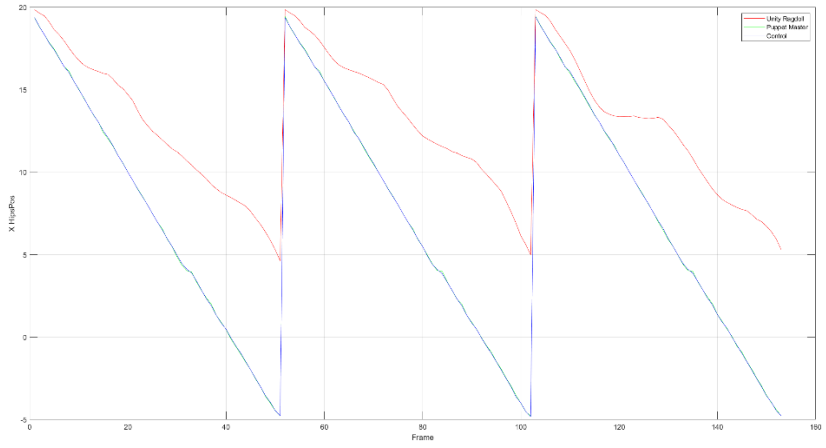
The results of each test scene have been collected and used to produce graphs comparing each of the x, y, and z positions of key bones on each character within the scene. Each of the graphs X axis displays the frame and the Y axis displays the corresponding coordinate for each character. As presented by the key the blue line displays the controls coordinates whereas the red displays a unity-built ragdoll and the green the puppet master plugin ragdoll.

Rough Terrain

The rough terrain data shows a clear connection between the bones that are directly connected to the spine as each of the graphs produced have a similar curve whereas the bones such as arms and legs that are not directly connected have a more unique curve. Throughout the rough terrain results the puppet master ragdoll performed the closest to the control but straying at certain sections whereas the unity ragdoll has clear deviations across each test run. The X Positions of each bone perform closest to that of the control with each x position across the board having a lower spread than other results. Whereas the Y and Z positions having a wider difference between each ragdoll.

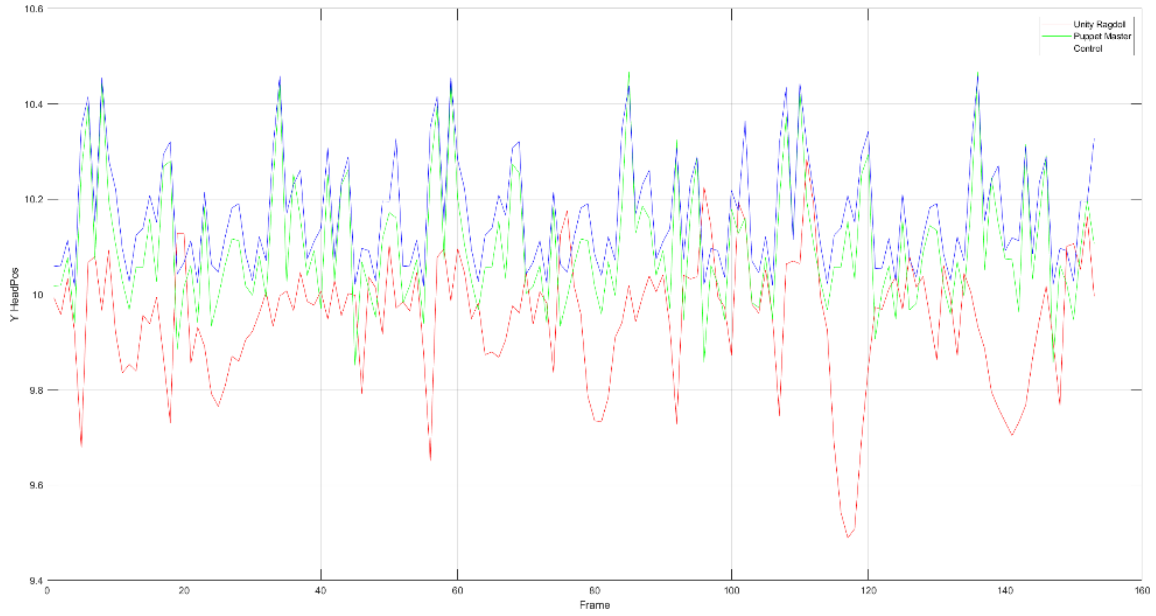


5 Rough Terrain Head X Position

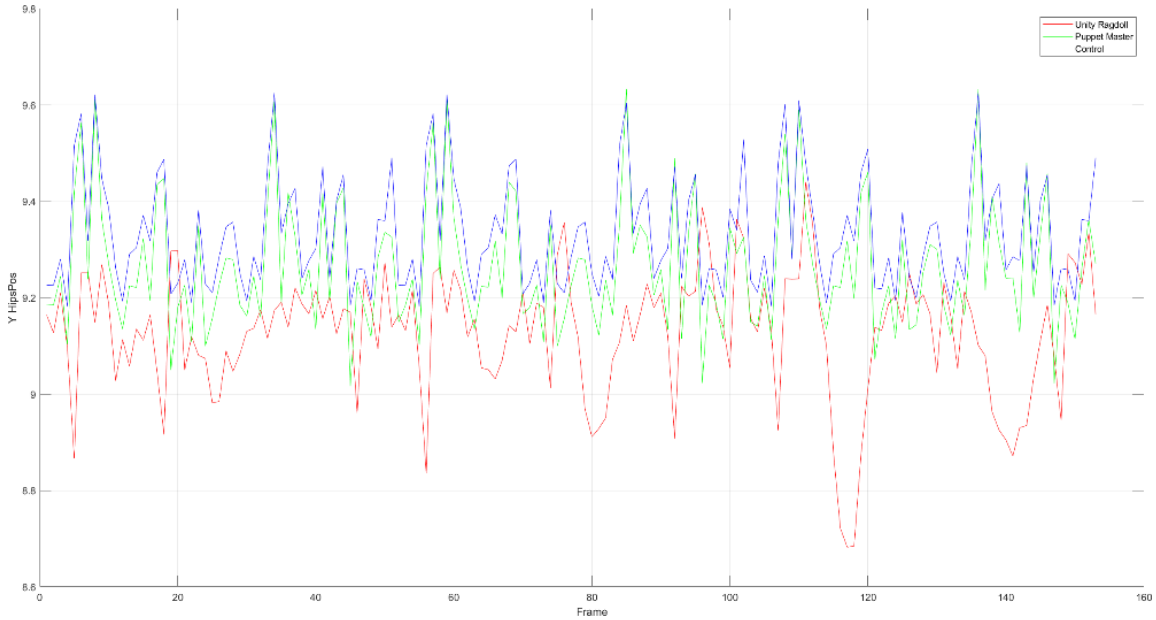


6 Rough Terrain Hips X Position

Each test condition reset the scene at around frame 53 this is shown in both figures with the spike in the position before falling again, with the control and puppet master falling at a much greater and steady pace whereas the unity ragdoll has a less steep decline which it levels out at points as seen in the final run of Figure 6.



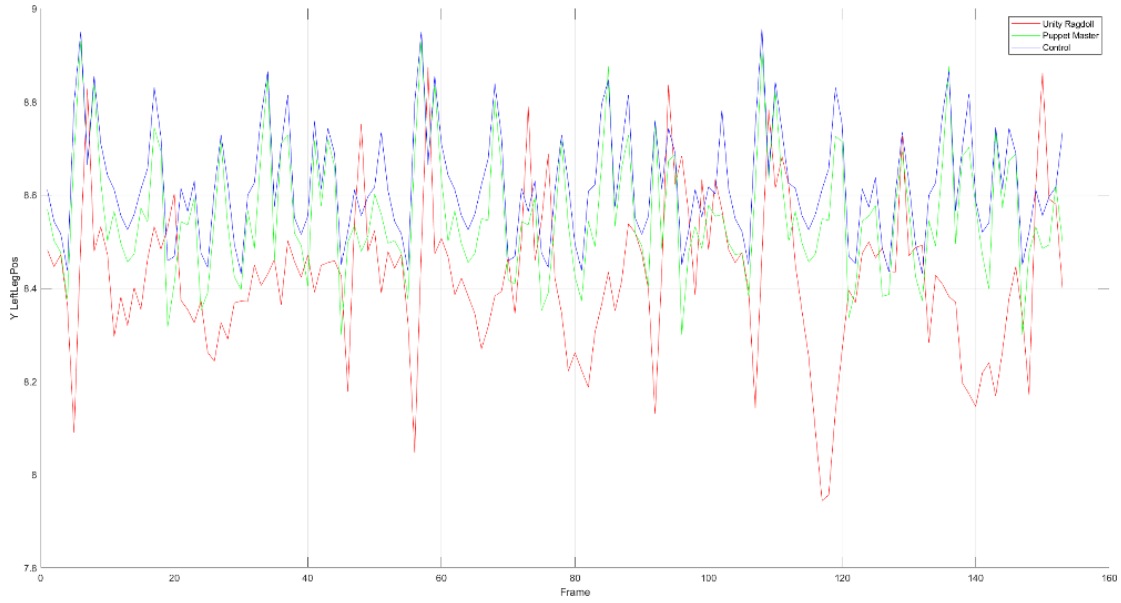
7 Rough Terrain Head Y Position



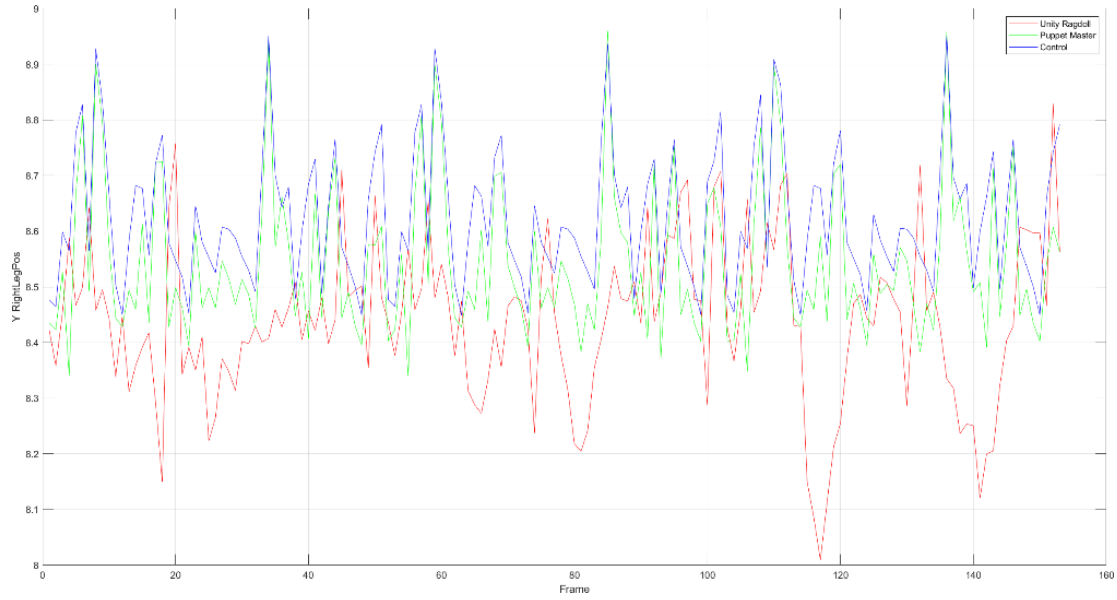
8 Rough Terrain Hips Y Position

The Y position of the bones have a much greater difference between each ragdoll in comparison to the x position. The puppet master ragdoll does stray from the control unlike the X position, but it does stay closer than the unity ragdoll which at points is invers to the control and puppet master ragdoll as seen in both figures 7 and 8 where the unity ragdoll dips to 8.7 Y just before frame 120 whereas the

other ragdolls positions are around 9.3-9.4 . A greater spread across each run of the test is also present within the results.



9 Rough Terrain Left Leg Y Position

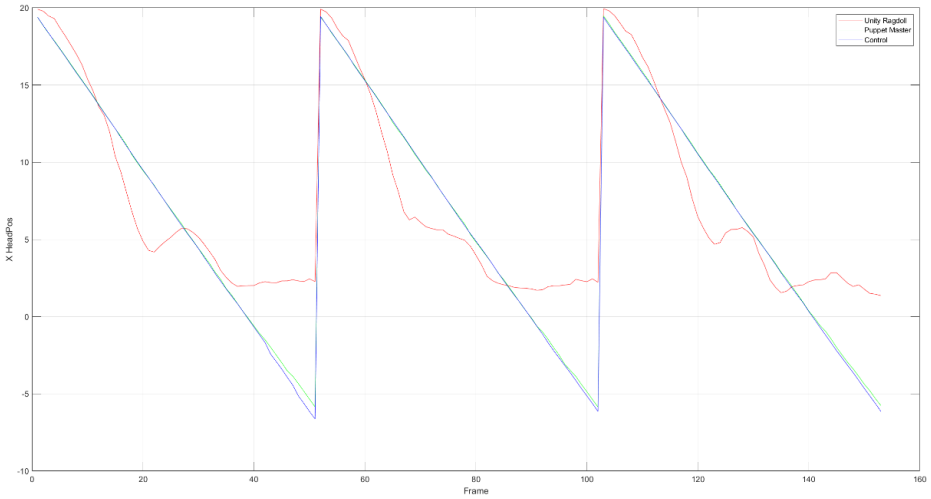


10 Rough Terrain Right Leg Y Position

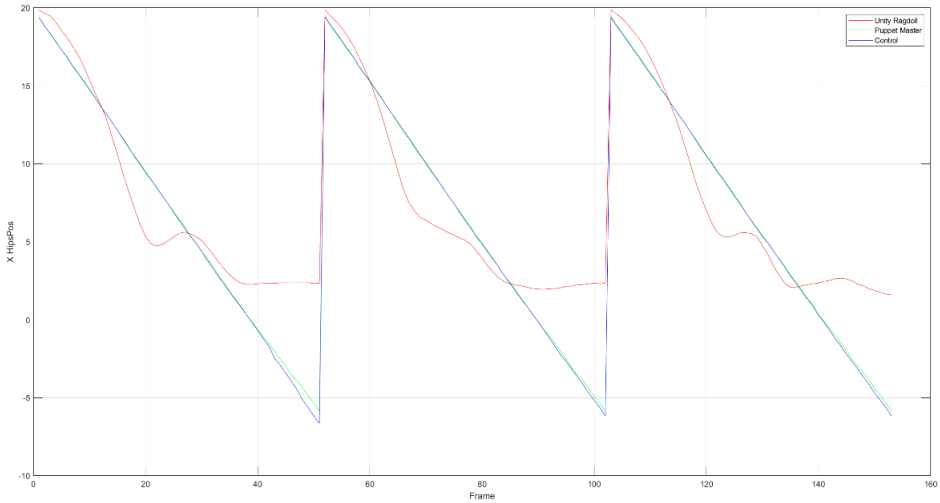
From the data gathered the legs show the greatest range in Y positions from the control. Each of the legs performs differently with spikes for the unity ragdoll being shown to catch up with that of the control in multiple sections of both figure 9 and 10. The performance of the puppet master ragdoll is at its widest gap from the control across all of the bones when looking at the legs.

Bridge

As with the rough terrain a connection in results is presented when comparing the bones directly connected to the spine. The results from the bridge show the puppet master ragdoll performing more in line with that of the unity ragdoll in certain bones with a wider gap between that of the control in most cases.

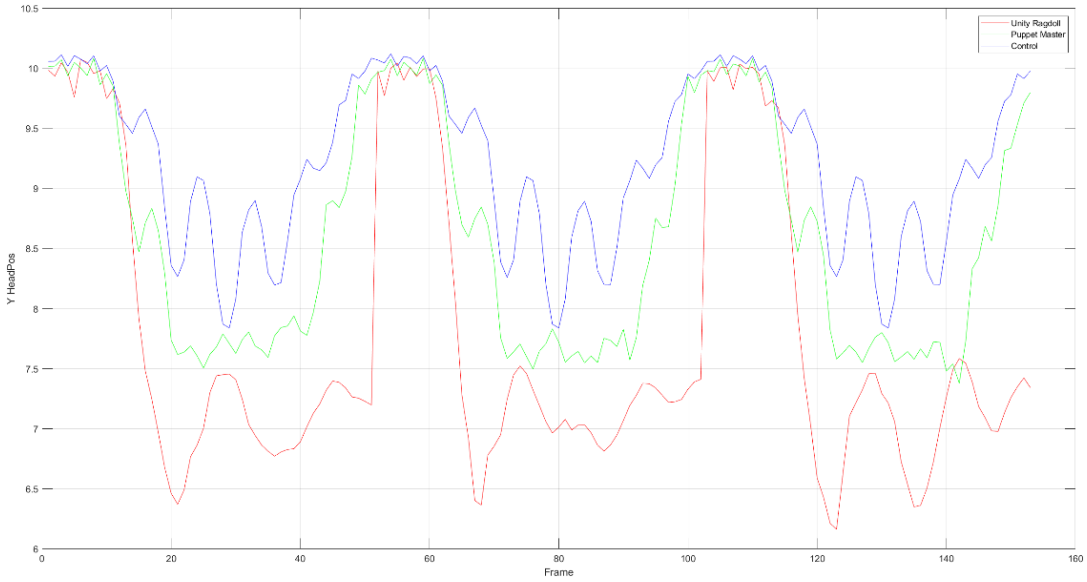


11 Bridge Head X Position

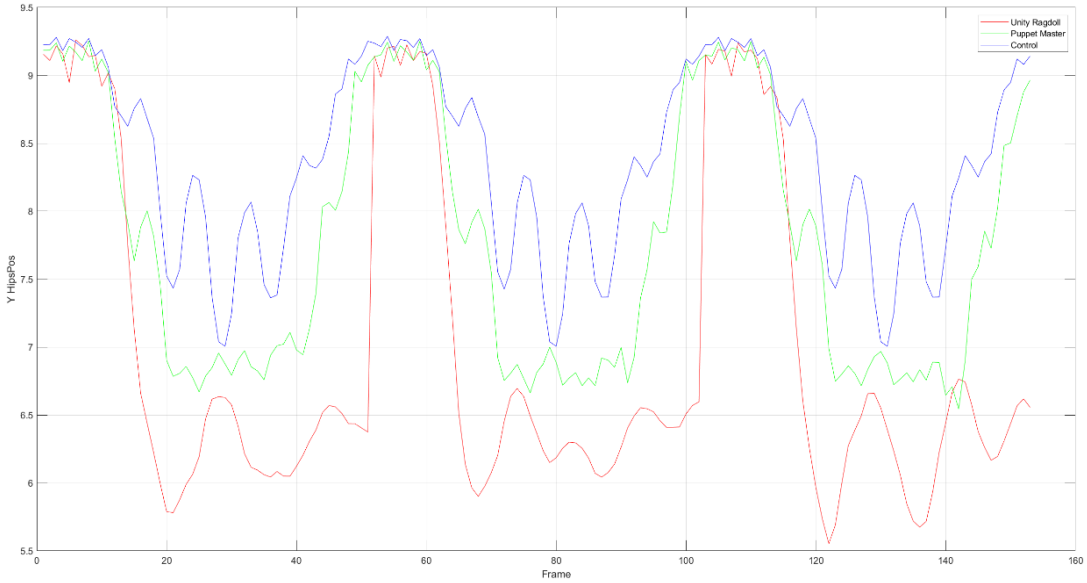


12 Bridge Hips X Position

Both of the X positions of the head and hips follow a similar curve to each other with the hips retaining a smoother curve in comparison to the head. The puppet master ragdoll preforms better than the unity one but still strays away from the control at certain points as seen around frame 40-50 in both figures 11 and 12. Unlike the control and puppet master ragdoll the unity ragdoll X position does dip below around 2.5 whereas both other ragdolls dip to -5 on the X axis. As seen in in both figures 11 and 12 each run through the test has varying results, each test restarts at around frame 53 and as shown both tests runs 1 and 3 follow a similar pattern having a large drop at the start before rising again at frame 20 whereas the second run is a smoother decline over the full run time.



13 Bridge Head Y Position

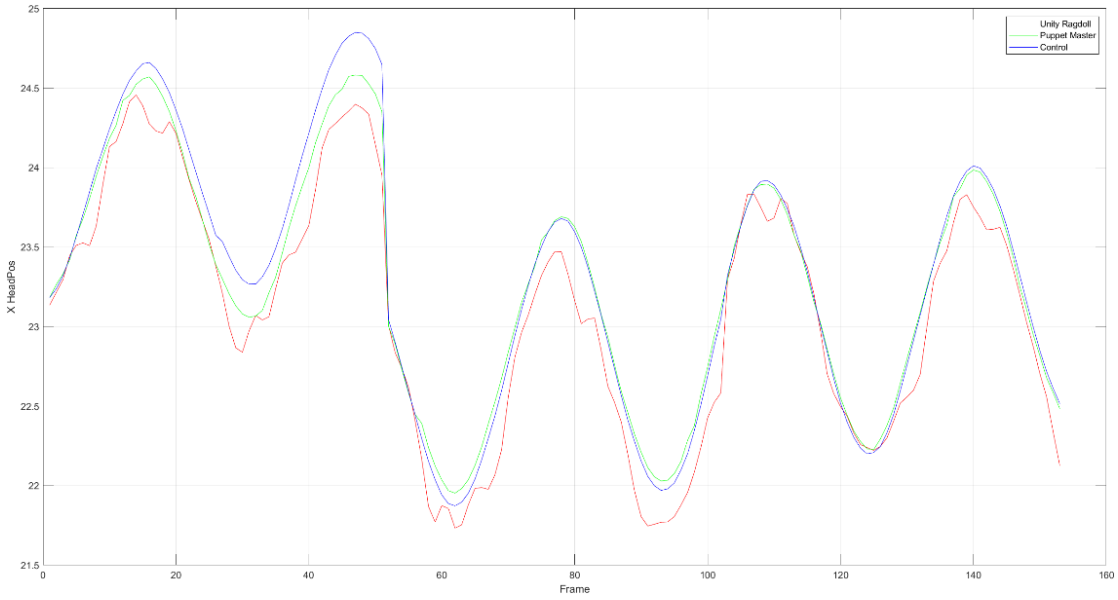


14 Bridge Hips Y Position

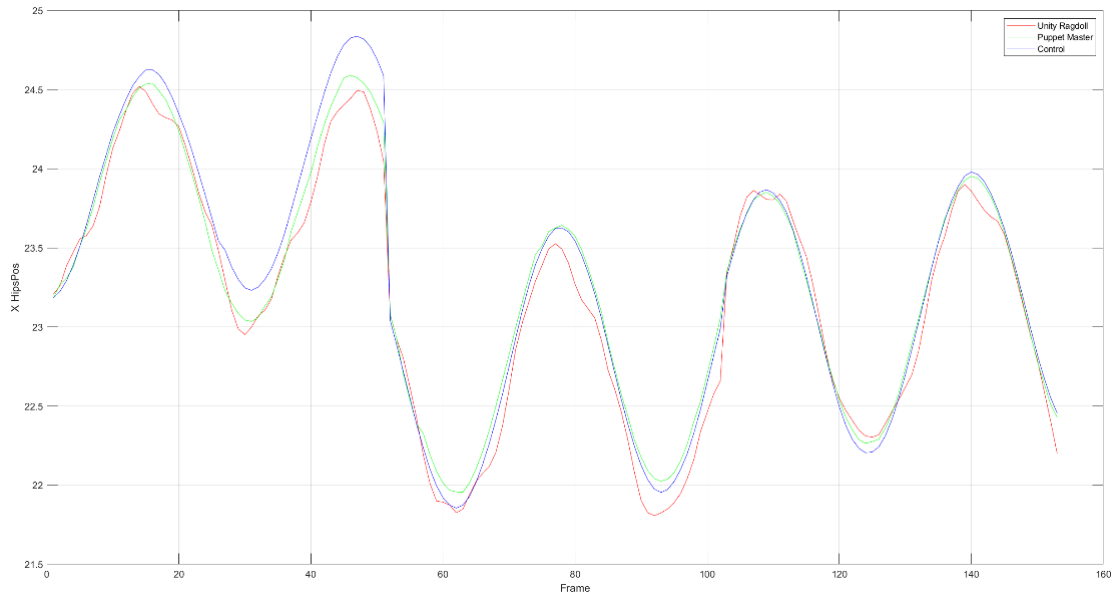
All three different ragdolls follow a similar trend across the y axis but with larger differences in performance to each other. Each of the ragdolls performs similar to each other at the start of the run as seen in figures 13 and 14 with the puppet masters being slightly closer in its results to the control in comparison with the unity ragdoll. Just before frame 20 all of the ragdolls start to dip with both of the control and puppet master having a slower decline in comparison with the unity ragdoll which as a large dip down to 5.7 which it doesn't build back up unlike the others which only dip to 6.7 but gradually build back to 9.2 by the end of the run this can be seen most at the end of both figures.

Water

The water has a noticeable difference from the other tests, each of the X positions follow a curve similar to that of the function being used to generate the waves with smaller differences in each ragdoll. Within these tests the puppet master appears to perform close to the unity ragdoll in comparison with the other test being run.



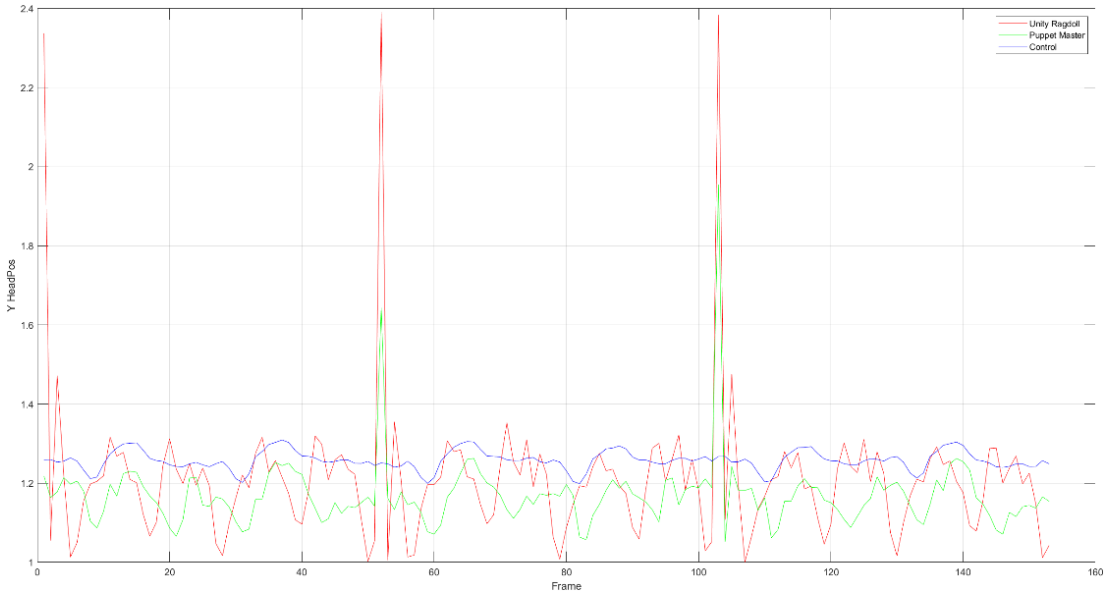
15 Water Head X Position



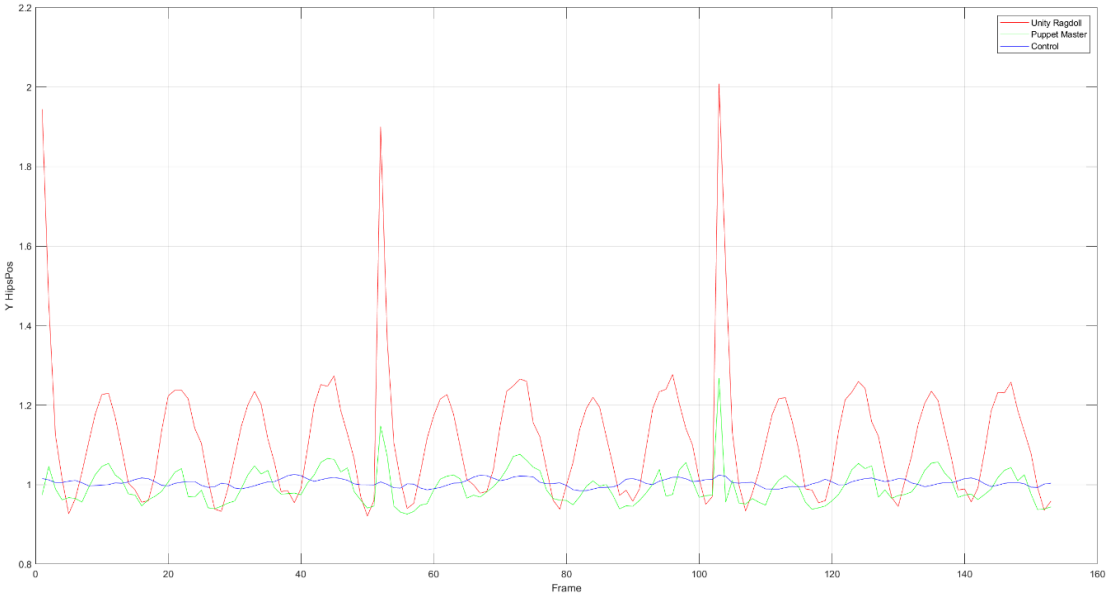
16 Water Hips X Position

With the head X position the curve being followed is similar that of the sinusoidal wave being used to generate the waves for the test this is most apparent after the first run of the test where the results appear to break from this trend. In the first run before frame 53 both the puppet master and unity ragdoll perform closer to each other than they do to that of the control these changes of the first run

where the puppet master starts to perform close to that of the control only tapering off close to the peaks of troughs of the curve. Although the unity ragdoll follows the same trend it is not as smooth when compared with the other two ragdoll in figures 15 and 16.

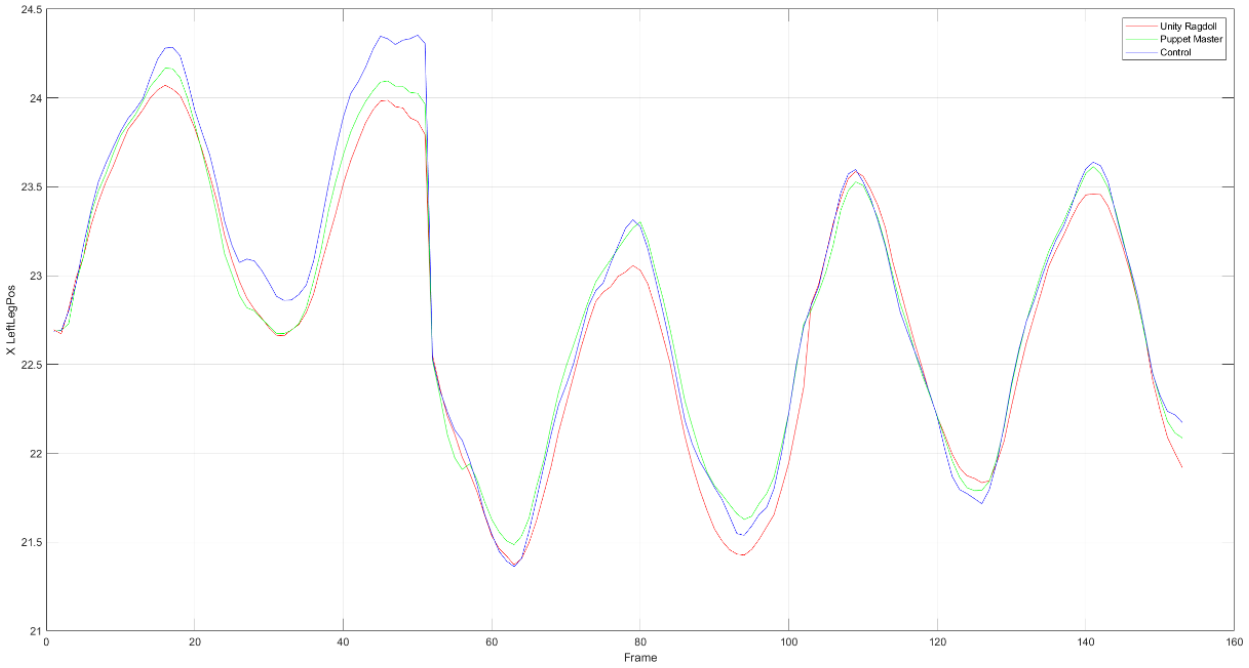


17 Water Head Y Position

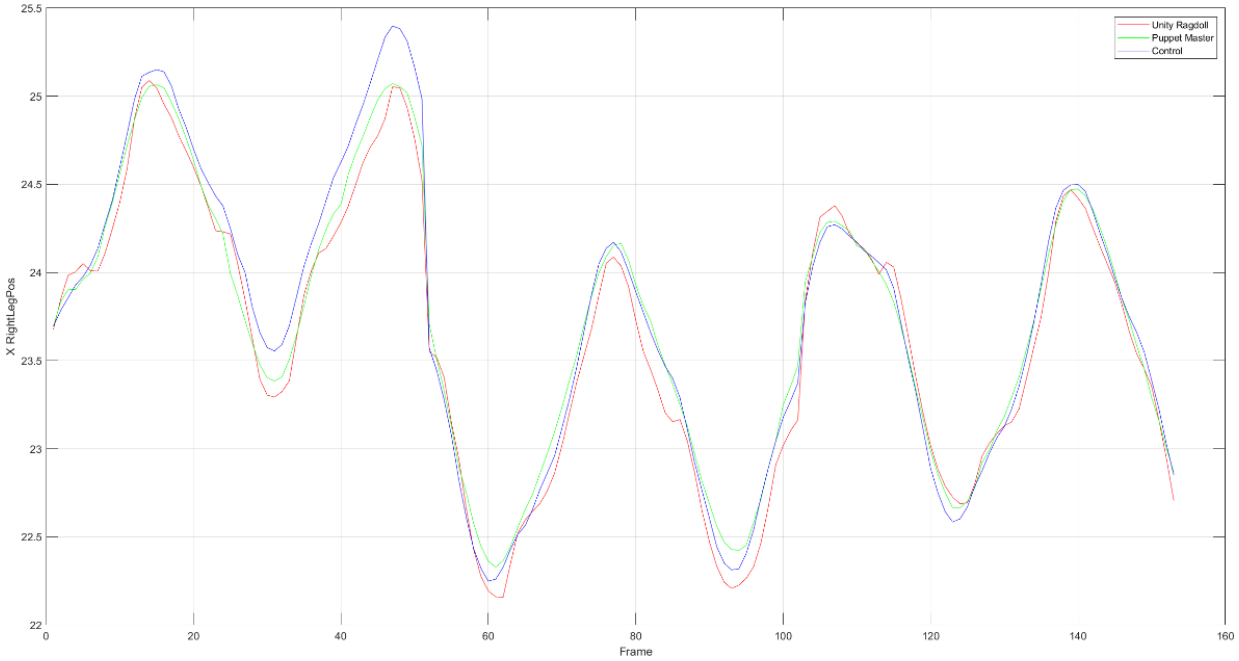


18 Water Hips Y Position

The Y positions of both the head and hip positions for each ragdoll widely differ from each other with more of a disparity appearing with in the in the head position and to a lesser extent in the hips due to the puppet master and unity ragdolls having a similar wave with the unity ragdoll having the higher peaks in comparison. Each of the ragdoll characters has a large peak up to 2 on the Y positions for the unity ragdoll and a smaller 1.3 on the Y position for the puppet master ragdoll as seen in figures 17 and 18 at the start of each run which does not appear in the control.



19 Water Left Leg X Position



20 Water Right Leg X Position

Both the left and right leg appear to follow a similar curve across all of the tests. Although the puppet master ragdoll still performs the closest to that of the control the unity ragdoll still keeps up well with the control only breaking off in a few places such as at frame 100 in figure 20 where it rises to 24.3 and in figure 19 the first and second runs the peaks and troughs stray away from that of the control.

Discussion and Analysis

Through the data collected from the tests many insights can be obtained for each of the ragdolls and test conditions. The data presents the puppet master as the most performant ragdoll in most situations but this not the case for all situations as well as the unity ragdoll appearing to stray further from that of the control although on the surface this may appear to be an issue this could become a benefit to the developer depending on the situation; these results collected will be elaborated further within this section of the report.

The results provided from the rough terrain test condition give a good example of how the active ragdolls would perform within most game conditions as most game environments will not be set on a flat plane with no obstacles to be avoided by the player. When looking at the X positions for the head and hips within the rough terrain presented in figure 5 and 6, test it appears that the puppet master ragdoll performs the best when compared to the control whereas the unity ragdoll appears to be less stable due to its position falling at a slower rate in comparison with the other two tests this could be caused by the ragdoll getting caught on the geometry whilst moving forward stopping it from moving for a brief period of time explaining the areas in both figure 5 and 6 where the unity ragdoll flat lines; the graph for the hips backs up this due to it having a smoother curve this could be caused by the hips being a more stable area of the ragdoll due to having more connected joints between it and the spine in comparison with the head which is a heavy weight only supported by the spine joint.

All of the Y positions as a whole for the unity ragdoll appear to be completely different to that of the other characters this could also be caused by the ragdoll getting left behind within the test leading to a delayed result further backing up this argument. Along with the unity ragdoll not aligning with the curve of that of the other 2 test characters it appears that in many sections of figures 7 and 8 the unity ragdoll is performing the inverse, showing that the ragdoll is falling over whilst the other ragdolls peak while moving over geometry. The puppet master ragdoll follows the exact curve of the control during in figure 5 and 6 whereas in figures 7 and 8 it follows a similar trend but slightly deviates from the control throughout, this displays that the geometry is having an effect on the ragdoll due to its performance in the Y position. Whilst not as significant as the effect on the unity ragdoll this could pose an issue in some situations. The difference between the performance in the X and Y positions displays that the puppet master has greater stability when walking over the geometry in comparison to that of the unity ragdoll, as the forward momentum gained does not appear to lead to the character falling over. Instead, walking over the terrain in a similar way to that of the control these claims are similarly backed up through the curve of the leg positions in figure 9 and 10 being similar to that of figures 7 and 8.

When looking at the results of the X position for the bridge section set out within figure 11 and 12 it is apparent that the unity ragdoll was not able to completely cross the bridge in the time allowed for the test, as at frame 35 it starts to level out showing that no more progress is being made with in the forward direction until the scene is reset. This behaviour is demonstrated across all three runs of the test with the ragdoll making quicker progress crossing the bridge in comparison to the other test characters until around frame 20 where in each run it appears to make negative progress this negative progress made within the rest of the run could be caused due to the initial momentum gained when stepping onto the bridge; after a while the momentum and weight of the ragdoll causes the bridge to bounce the ragdoll backwards. This is backed up when looking at the Y position of the head and hips in figures 13 and 14 where at the same frame that the negative progress is made there is also a spike within the Y position proving the at this time the character did make a large jump backwards on the bridge further testing would be required in order to provide a conclusive source of this movement but due to the sudden movement of the character at the start of the test it is likely that the momentum gained is the source of this backwards force, the Y positions of the character also back up the fact that the unity ragdoll did not managed to fully cross the bridge. With both the puppet master and control there is a sudden dip at the start of each run when the ragdoll steps onto the bridge which towards the end of each test the Y position begins to rise again as the character climbs back to the top of the bridge, this climb towards the end of the bridge is not present with the unity ragdoll . Across all three of the tests characters the Y positions presented in figures 13 and 14 have very sporadic movement with the graphs having large and frequent dips and rises in the position this is likely due to the physical effects of the bridge and the characters readjusting themselves in order to stay balanced due to the constant movement caused by the effects of their weight on the bridge.

The water test provide some interesting results especially when looking at the X positions of each of the ragdolls. Within figures 15,16,19 and 20 each of the positions follow a similar wave to that of the one being used to produce the waves effecting the character; within all of these results each of the test characters perform similar to each other. This could be due to the large amount of force being applied on them as well as being easier for the ragdolls to maintain a swimming position due to them not needing to maintain balance like they would if they were stood up like in the other test conditions. When looking at the Y position for the head in figure 17 it is clear that the unity ragdoll is less stable when compared to the puppet master ragdoll this can be seen in the greater differences in the highs and lows, although neither of the active ragdolls appear to follow the trend of the control. In both figures 17 and 18 both of the ragdolls do share the same outlying spike at the start of each one of the test runs with this spike the unity ragdoll does end up with the greater difference in position but it is still a noticeable spike with in the puppet master ragdoll; these spikes in the Y position could be an

issue with the way the tests are being conducted and could be caused by brief transition states in the animation between runs. This would explain why the less stable unity ragdoll is effected to a greater extent.

After taking the averages of each of the recorded bones for all of the ragdolls across all of the tests it is clear that within all the tests where the ragdolls need to balance themselves that the puppet master ragdoll has the smallest difference in its positions across the board with most positions being within a 0.1 difference of the control, whereas the unity ragdoll appears to struggle to keep up with both of the other characters being tested. The unity ragdoll displays the greatest difference in results with the control when looking at the head where the average across each of the tests except the water test are consistently off from that of the control and puppet master ragdoll. This could highlight an issue with the stability of the unity ragdoll as due to the lack of joints and weight of the head could lead to instability when moving around which is not present in the puppet master ragdoll. The results of both the rough terrain and the bridge section are completely different to that of the water level in which both the unity and puppet master ragdolls perform closely to that of the control on average this is most likely due to the character not needing to keep themselves balanced instead having them move with the water, this could present an issues within some development cases where extra control is needed over the character moving within the water as the data shows that both of the ragdolls are highly effected by the external added by the waters movement.

Table of Averages for Terrain Test

	Head X	Head Y	Head Z	Hips X	Hips Y	Hips Z
Control	7.182493	10.16711	30.2179	7.169213	9.332607	30.23337
Puppet Master	7.189104	10.09911	30.21423	7.170414	9.265056	30.23096
Unity	12.73574	9.947637	31.60932	12.75849	9.126291	31.61968

Table of Averages for Bridge Test

	Head X	Head Y	Head Z	Hips X	Hips Y	Hips Z
Control	6.489544	9.223603	30.19887	6.476423	8.389106	30.21433
Puppet Master	6.581433	8.618008	30.20961	6.562378	7.783973	30.22571
Unity	7.243757	7.793279	30.65697	7.344911	7.002978	30.67281

Table of Averages for Water Test

	Head X	Head Y	Head Z	Hips X	Hips Y	Hips Z
Control	23.30515	1.258013	-6.70029	23.27922	1.005087	-7.49658
Puppet Master	23.26723	1.169821	-6.69239	23.24975	0.994377	-7.50395
Unity	23.09578	1.213697	-9.16749	23.17202	1.116899	-9.98151

Although the data shows that the unity ragdoll does not perform to the same level as the puppet master ragdoll this does not mean that it is necessarily worse for all implementations as certain games may be trying to achieve a more cartoonish and less realistic physics-based character such as the ones seen in games like gang beasts the unity ragdoll appears to suit this type of environment much more than the puppet master ragdoll which through the data shows that it performs more in line with the expected results of the control. The puppet master ragdoll would be better suited for a more traditional style of game where an animation is expected but still needs physics interaction in order to deform around certain elements this is due to its much higher stability in comparison with the unity ragdoll. Either of the ragdolls could be a suitable choice for implementing into a game that requires the player to swim under water although work may need to be done in order to make the over all movement of the character more predictable due to the forces in the water having a great effect on both of the ragdolls movements.

Conclusion

Within in this paper two different solutions for active ragdolls have been evaluated in order to define which ragdoll would perform best under three test conditions; a rough terrain, rope bridge and water test have been conducted in order to simulate real-world worst-case scenarios for the active ragdoll systems. These tests where conducted in order to determine which active ragdoll system would be most appropriate for game developers looking to potentially implement them into a game. Throughout the testing interesting aspects of each of the active ragdoll system have been highlighted these include the unity based active ragdoll system being less performant within the test when compared to the puppet master active ragdoll; although on the surface this may seem like a clear case of which ragdoll to implement the lesser performance of the unity based active ragdoll may lend itself better to specific types of gameplay that would not be appropriate for the other puppet master ragdoll system. These insights on the performance of the ragdoll systems can benefit developers in deciding on which type of ragdoll system would be most appropriate to the specific needs of the game being developed.

Recommendations

In the future further research can be conducted into this subject in order to cover some of the short comings of this report. With in this report statistical rigour had not been applied in future instances statistical analysis could be used in order to verify the effects presented by each active ragdoll system however the intended purpose of this study was to provide initial insights into active ragdoll system through a qualitative pilot study due to this being a first study with no prior coverage. Future work within this field would require a larger sample size of active ragdoll methods such as implementing a pre trained AI model; along with conducting more scenarios for each ragdoll with more data being gathered from each test condition in order to conduct a solid statistical analysis. More work could also be conducted into the visualisation of the active ragdoll systems though conducting a user study in order to determine which active ragdolls visually perform best to an average user.

Bibliography

Ahmad, F. N., 2013. An Overview Study of Game Engines. *Int. Journal of Engineering Research and Applications,* 3(5), pp. 1673-1693.

Ariel Kwiatkowski, E. A. V. K. C. K. L. J. P. M. v. d. P. M.-P. C., 2022. A Survey on Reinforcement Learning Methods in Character Animation. *Computer Graphics Forum,* 41(2), pp. 613-639.

Binh Huy Le, Z. D., 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics,* 33(4), pp. 1-10.

Chris Lewin, M. T. T. W. C. W. P. W., 2013. *Rod Constraints for Simplified Ragdolls.* Anaheim, Association for Computing Machinery, pp. 79-84.

French, V., 2012. *Rag doll physics ,* Bournemouth: s.n.

GDC, 2018. *Physics Driven Ragdolls and Animation.* [Online]
Available at: https://www.gdcvault.com/play/1025210/Physics-Driven-Ragdolls-and-Animation

GDC, 2020. *GDC Vault.* [Online]
Available at: https://www.gdcvault.com/play/1026712/Machine-Learning-Summit-Ragdoll-Motion

Glimberg Stefan, M. E., 2007. *Comparison of ragdoll methods,* Copenhagen: s.n.

Joe Booth, V. I., 2020. *Realistic Physics Based Character Controller,* s.l.: Arxic.

Kenwright, B., n.d. *Joint-Torque Control of Character Motions,* s.l.: s.n.

Pathiral, 2020. *Water Float.* [Online]
Available at: https://assetstore.unity.com/packages/tools/animation/water-float-164693
[Accessed 10 May 2023].

Pawel Wrotek, O. C. J. M. M., 2006. *Dynamo: dynamic, data-driven character control with adjustable balance.* Boston, Association for Computing Machinery, pp. 61-70.

Phelps, A. M., n.d. *Simulating Realistic Water in Low Performance Game Engine,* Rochester: Rochester Institute of Technology.

RootMotion, 2022. *PuppetMaster.* [Online]
Available at: https://assetstore.unity.com/packages/tools/physics/puppetmaster-48977
[Accessed 10 May 2023].

Sujeong Kim, S. J. G. D. M., 2013. *Velocity-based modeling of physical interactions in multi-agent simulations.* Anaheim, Association for Computing Machinery, pp. 125-133.

Unity Technologies, 2021. *Hinge Joint component reference.* [Online]
Available at: https://docs.unity3d.com/Manual/class-HingeJoint.html
[Accessed 20 03 2023].

Unity, 2023. *Create a Ragdoll.* [Online]
Available at: https://docs.unity3d.com/Manual/wizard-RagdollWizard.html

Unreal, 2023. *How-To Apply a Physical Animation Profile.* [Online]
Available at: https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/PhysicsAssetEditor/HowTo/ApplyPhysicalAnimationProfile/
[Accessed 28 01 2023].

Victor B. Zordan, A. M. B. C. M. F., 2005. Dynamic Response for Motion Capture Animation. *ACM Transactions on Graphics,* pp. 697-701.