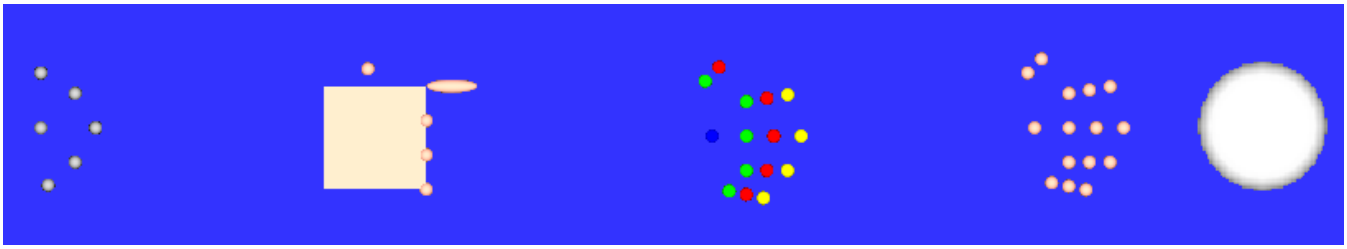
Kinematic and Dynamic Hand Grasping Animation

Ryan Todesco



Abstract

The objective of this system is to animate a hand picking up an object with accurate transferring of forces. The development of this system focused on the transferring of force, then the look of the hand, followed by the movement of the hand to get to the object. In related works, motion capture is commonly used, and typically for hand animation follows one of two paths: real time capturing, or pre-calculation. Hand animations are useful in the animation industry but can also be used virtual assembly and rehabilitation. This system, within the framework of other works, was designed to be attached to other motion capture systems, to add the force transferring applications to them. The basis for this system is a particle system that use forward Euler for its motion. The particles represent the object, the palm, and parts of the finger and have different behaviors depending on their type. Dynamic motion is used for the motion of the object and palm particle, while kinematic motion is used for the rotation of the hand and fingers. The hand is a kinematic chain made of the palm and finger particles. The particles also have different collision detection and resolution behaviors depending on their type. The evaluations of this system showcased that the force transferring, which was given the most time, worked as expected, but the rotation of the hand and fingers had their faults. In conclusion, parts of this system worked as intended, but more future effort would still be required.

CCS Concepts

• ***Computing methodologies*** → *Collision detection ; Collision resolution; Articulated character; Force transferring;*

1. Introduction

Picking up something in the real world causes forces to be transferred between the hand and whatever is being picked up. In animation this done in several different ways. A joint can be attached between the hand and the object, meaning wherever the hand moves the object moves accordingly, however this means that the object is attached to the hand, and does not simulate being actual carried by the hand. Models can be changed to include held items but this also has the same problem where the object is now attached to the hand and can not be dropped if the grasp is let go. The inspiration for this system was a video game in which an animated character was holding a gun, however the gun looked as if it was floating within the grasp and even oscillated slightly when the character moved their hand. This is what this system was sent out to fix.

The objective of this system is to as accurately as possible simulate the transferring of forces between the hand and the object, without the object being attached to the hand in any way, while also ensuring that the object stays within the hand when grasped. The difficulties of this system include what is the best way to create the hand for the best visual appeal, accurate collisions so the hand grasps the object correctly, and an accurate transferring of force.

Most time went into the development of the transferring of force, followed by the look of the hand, and lastly by the movement of the hand towards the object. This order was chosen based on perceived difficulty, but also be potential future development, mainly the decision to focus less on the movement of the hand before grasping the object. The idea being that the force transferring and hand system could be attached to other systems that control the movement of the hand, such as the commonly used in the area motion capture.

2. Related Work

The human hand is a complicated articulation of movement, but it is also an important tool for everyday life, so it makes sense the amount of research that has gone into simulating the hand for animation, but also what simulating the human hand can do for the real world.

2.1. Development

Some developments into animating hands focus on simulating the behaviour of human hands during common actions, such as playing the guitar. ElKoura and Singh [ES03] used inverse kinematics to animate a hand playing simple chords on a guitar. Sanso and Thalmann implemented automatic grasping systems that change how an object is grasped depending on the size of the object, and used both direct and inverse kinematics [ST94]. However, the most common development into hand and grasping animations implement motion capture data, including ElKoura and Singh, which allows actual behaviour to be used. This motion capture data is commonly used in two different ways; real time capturing of data, versus precalculation grasping configurations.

The advantage of using real time hand tracking data for animation means that the animation can be more efficient and does not require the use of prerecorded data. Nasim and Kim and Höll et al. both used virtual reality in order to showcase their physics-based interactive grasping interactions that resulted in the ability to "perform real-time grasping tasks in complicated virtual environments" [NK16] as well as "convincing simulations of many types of actions such as pushing, pulling, grasping, or even dexterous manipulations such as spinning objects between fingers without restrictions on the objects' shapes or hand poses" [HOAL18].

The other side of using motion capture use different methods for determining stable grasp configurations and realistic motions. Tian et al. used "machine learning and particle swarm optimization to automatically pre-compute stable grasp configurations for that object" [TWMZ19] in order to create plausible looking grasps. Zhao et al. used a data-driven synthesis algorithm and a physics-based motion control algorithm in order for a "user to act out the desired grasping motion in front of a single Kinect camera" [ZZMC13]. Lastly, Pollard and Zordan built a controller that could uphold compliant yet controllable motion, and "dds compensation for movement of the arm and for gravity to make the behavior of passive and active components less dependent on the dynamics of arm motion" [PZ05].

2.2. Uses

The obvious use hand animation is very displaying believable characters in animated films, television and games. But there are several other fields that would benefit from more accurate hand animated systems, such as virtual mechanical assembly and disassembly and rehabilitation.

Li, Xu, Ni and Wang researched a glove that could be used to "manipulate 3D objects for virtual assembly" to create something "far more intuitive, natural and immersive" [LXN16]. This would allow more viable testing and training protocols, as well as be used for the improvement of task learning and automation in assembly and disassembly, as discussed by Aleotti and Caselli [AC11].

Moreira et al. developed a more accurate motion tracking glove that could be used in animation industries, but hand a larger goal for hand function impairment rehabilitation [MQF*14]. The goal was to use inertial measurements units to more accurately capture human hand motion, giving them "the ability to monitor hand motion for extended periods of time in a more natural environment can help assess ADL–based hand function and ultimately improve the rehabilitation outcome".

2.3. Within Framework

Several of the papers mention using another system an attaching it to their motion capture system, hand model and virtual reality system in order to build a more complicated interactive system. This is the objective of this system, to create something to attach to the already existing motion capture or virtual assembly systems and create something bigger for more interactive purposes.. The idea being that the particle system explained in Section 3.1 could be replaced with the motion capture system, and the force transferring system would still apply. This would mean that motion capture or virtual reality systems would be able to pick up virtual objects using the system described in this paper.

3. Overview

This hand grasping animation is built using a particle system and implements the movement of those particles using kinematic and dynamic motion. The dynamic motion is used to control the major movements such as the movement of the hand as a whole and the movement of the object. Dynamics are also responsible for the transferring of forces between the hand and the object. Kinematics are used for the minor movements of the particle, that is the rotation of the fingers in the hand.

Each item see in the system is rendered using OpenGL and The OpenGL Utility Toolkit. All items a represented using spheres and are drawn using the glutSolidSphere function. The translation of the sphere is determined by the particle system, see subsection 3.1, the rotation of the sphere is determined by the dynamic motion, see subsection 3.2, and the scaling of the sphere is determined by the particles set size. TCL and TK were also used to display the system and give some interactive control such as camera control, as well as allow the parameterization of the evaluations.

3.1. Particle System

Each of the rendered items in this system are represented as particles with the same attributes. These attributes are as followed; Global X, Y and Z position, X, Y and Z velocity, X, Y and Z forces acting on that particle, mass, size, type and a determiner of if that particle has collided or not. The type of the particle determines what forces or motions apply to that particle and specific collision rules described in section 3.3. The different types of particles are object, the item being picked up, palm, the center of the hand, and the proximal, intermediate and distal phalanges, the different parts of the finger. Regardless of type, each of the particles

major dynamic motion is defined the same way using the Forward Euler approach, defined in Algorithm 1.

Algorithm 1 Particle Simulation Loop

for each particle i **do**
  $Fi = Force(i)$
  $Fi = Collision(i)$
  $ai = Fi/mi$
  $vi,new = vi + ai * 0.01$
  $pi,new = pi + vi * 0.01$
  $display(pi)$
end for
for each particle i **do**
  $pi = pi,new$
  $vi = vi,new$
end for

In this simulation loop, the Force function adds the dynamic forces or kinematic movements acting on that particle as seen in section 3.2, and the Collision function handles the collision detection and resolution, as defined in section 3.3, and adds the appropriate forces.

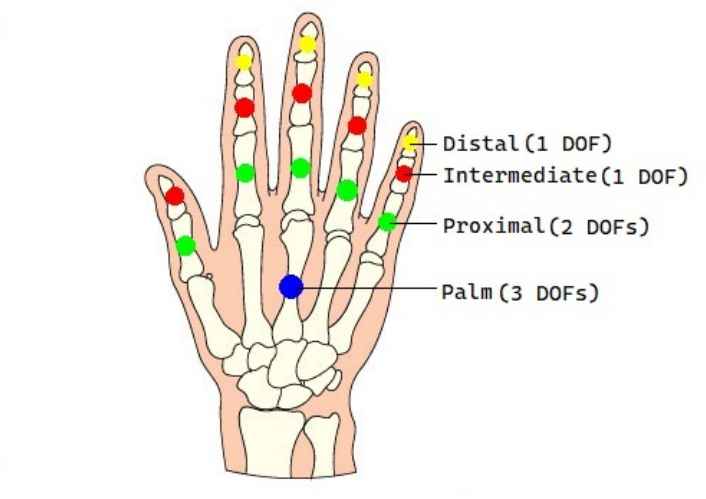
3.1.1. Hand Articulation

As stated before, the hand is made up of four different types of particles for a total of fifteen particles. One palm particle, five proximal phalanges, five intermediate phalanges, and four distal phalanges. As seen in Figure 1, the palm particle as three degrees of rotational freedom and three degrees of translation freedom, the proximals have two degrees of rotational freedom in their X and Y rotations, and the intermediate and distal phalanges have only one degree of rotational freedom in their X rotation. So the is a total of twenty five degrees of freedom. These degrees of freedom where chosen as a simplified version of the defined degrees of freedom of a hand, excluding the more complicated degrees of freedom that the thumb possesses [ES03].

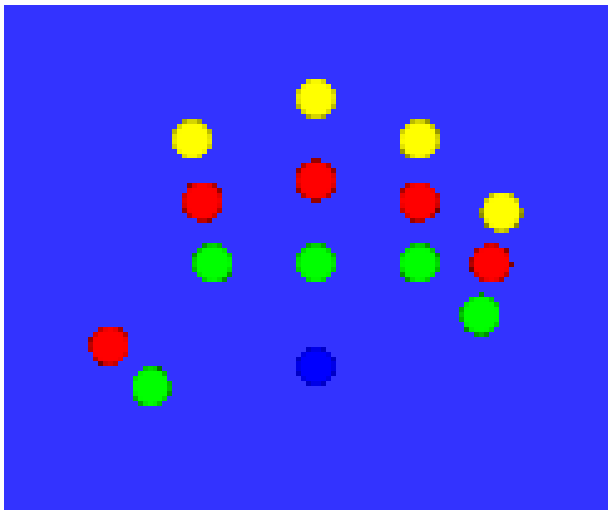
The hand is a kinematic chain created using OpenGl's built-in push and pop functions. The root of the chain is the palm particle, then each proximal phalanges are connected to the palm, the intermediates are connected to them, and the distals are connected to them.

Since each finger particle's global position is needed for the collision calculations, each finger particle is treated as an end effector and such a different calculation is required for determining where that particle is in space. This is to due to each particle having a different chain of rotations and translations from the kinematic chain. These differences in calculations can be see in Algorithm 2.

The reason a kinematic chain is used in conjunction with a particle system is to save computational time. During the dynamic motions of the hand, the end effectors do not need to be recalculated using Algorithm 2 and therefor can simply be a part of the kinematic chain for determining where they need to be rendered. It is only when the palm particle collides and the finger particles start rotating does the end effectors position matter for collision calculation. So by relying on the chain during this part less computations are needed to be done.



(a) Hand Drawing



(b) Hand Render

Figure 1: Types of Hand Particles

3.2. Motion

As stated before, the items in this system are moved by either dynamic or kinematic motion, depending on whether it is classified as major motion or minor motion, respectively. Major motion is the overall movements of the object and the hand as a whole, and minor motion is the movement of the individual finger parts, as well as the rotation of the hand. The dynamic motion is also used for the transferring of forces from the hand to the object, and vice versa.

3.2.1. Dynamic Motion

Independence is defined as when an item is not in collision with any other item. When the object is independent, it's motion is simply defined as gravity, a downward force of -9.8, acting on the mass of the object, creating the force that is used in Algorithm 1. When the hand is independent, a force is created that pushes the hand

Algorithm 2 Finger End Effector Calculation

for each finger particle i **do**
  if i is proximal **then**
    $p \leftarrow palmTranslation * palmRotation * xRotation * yRotation * fingerEndEffector$
  else if i is intermediate **then**
    $p \leftarrow palmTranslation * palmRotation * parentFingerRotation * parentFingerTranslation * xRotation * fingerEndEffector$
  else
    $p \leftarrow palmTranslation * palmRotation * parentParentFingerRotation * parentParentFingerTranslation * parentFingerRotation * parentFingerTranslation * xRotation * fingerEndEffector$
  end if
end for

towards the object. A vector is created by subtracting the point of the palm particle by the point of the object, it is then normalized to gain a pure direction vector, and lastly multiplied by a constant in the X,Y, and Z elements to insure the hand reaches the object in the X,Y, and Z position at the same time.

Once either item is no longer independent, the immediate collision resolution implementations occur. See section 3.3 for more detail. If the collision is between the object and the hand, the force transferring process begins, as seen in Algorithm 3. Fi/Fj represents the current loops force of that particle, while FI/FJ represents the stored force found within the particle. The *inst* element of the algorithm is to account for the loop delay caused by Algorithm 1. After the collision, there is one loop of the particle system where the stored forces is inaccurate. This *inst* element cancels out the particles current force to keep it in place so that the particle is ready for the next loop. The *massAdjust* is used to ensure that the hand and the object move at the same velocity, regardless of their different masses.

For the finger particles, their dynamic force used for the movement before the collision is simply equal to the force of the palm particle at that time.

3.2.2. Kinematic Motion

Kinematic motion is used for all of the rotations within the system; the rotation of the hand as a whole, and the rotation of the each finger joint. The hand needs to point towards the object, so the rotation towards the object not only needs to be calculated, but needs to be recalculated at each step due to the object having its own dynamic motion. To get the angle the hand needs to rotate to, the system uses the property of Dot Product

$$a \cdot b = ||a||||b|| \cos\theta \quad (1)$$

where the a vector is a vector pointing forwards from the hands starting position, and vector b is a vector created by subtracting the point of the palm from the point of the object. Since both vectors

Algorithm 3 Force Transferring

$inst \leftarrow true$
for each collided particle i,j **do**
  if i is object **then**
    $Fi \leftarrow Fi + gravity * Mi$
    store gravity * Mi in particle
  else if i is palm **then**
    if All hand particles collided **then**
      $Fi \leftarrow handF$ {Parameterized Variable}
      $massAdjust \leftarrow (Mj - 1) * (handF + FJ)$
      store handF + massAdjust in particle
    else
      $Fi = Fi - FJ$
      store -FJ in particle
    end if
  end if
  if inst **then**
    $Fi \leftarrow Fi - FI$
    $inst \leftarrow false$
  else
    $Fi \leftarrow Fi + FJ$
  end if
end for

can be normalized, solving for the angle is simple. Algorithm 4 describes how the hand uses that angle to rotate the hand into place.

Algorithm 4 Hand Rotation

if θ != X rotation **then**
  if Objects Y position < Hands Y Position **then**
    $Xrotation \leftarrow Xrotation - 1$
  else if Objects Y position > Hands Y Position **then**
    $Xrotation \leftarrow Xrotation + 1$
  end if
end if
if θ != Y rotation **then**
  if Objects X position <= Hands X Position **then**
    $Yrotation \leftarrow Yrotation + 1$
  else if Objects X position > Hands X Position **then**
    $Yrotation \leftarrow Yrotation - 1$
  end if
end if

The rotation of the fingers is much more simple, as they do not rely on rotating into position, but instead rotate until they have collided. Each finger particle has a unique rotation or rotations and those rotation increases or decreases until that particle collides. The rotation process for the finger particles does not start until the palm particle has collided.

3.3. Collision Detection and Resolution

There are two collision detection and resolution mechanics within this system. One that is unique to the object, and one that applies to every item. The objects unique collision is simply with the ground and uses a spring to keep it above the ground. If the object falls

below the established ground, a spring is attached with a spring force of -1000 and a dampening force of 20. This ground force is turned off once the hand has collided with the object. The reason for this is described in Section 4.

During the other collision detection, each particle is tested against specific other particles, while ignoring certain types of particles. All types of hand particles, that is the palm and the four finger types, do not attempt to detected collision with each other. This is due to their starting close proximity which is used for the collision detection. So, the only particle collisions that the second process detects are hand particles with the object, and the object with hand particles. Since each object is rendered as a sphere, the collision detection checks if the length between the current particle and every other viable particle added with the current particles size, subtracted the other particles size, and if that is smaller than an established epsilon, then collision occurs. The palm particle and finger particles have different epsilons because the palm particle is only moving transitionally, while the finger particles are moving rotationally meaning they have bigger jumps during their steps, resulting in the epsilon for the finger detection being smaller.

The second collision resolution is a cancellation of force. Once the particle has collided, an equal force is pushed in the opposite direction into the particle, causing it to stop. This simulates Newton's law that states that every actions has an equal and opposite reaction. The equal action gets pushed into the other particle, and the opposite reaction gets pushed back into the particle, forcing it to stop. However, the force pushed back into the particle is not simply the force built up during this loop, nor is it the currently stored force within the particle. If those were used, the force would become zero, but the velocity would take some time before it came to a stop. Since the objective of this collision is to fully stop the particle, a different method was used. A force calculated using the particles current velocity and its mass is used to create a larger force in the opposite direction, so when the velocity is calculated later, it results in zero. The hand particles each of a mass of one, so its mass is inconsequential in the calculation.

Algorithm 5 describes the second collision detection and resolution

Algorithm 5 Collision Detection and Resolution

```
for each particle i,j do
  if i is palm particle then
    if j is finger particle then
      continue
    end if
    if (ijLength + iSize) - jSize < epsilon then
      set particle collision to true
      Fi ← −Vi/0.01
    end if
  end if
  if i is object particle then
    if (ijLength + jSize) - iSize < epsilon then
      set particle collision to true
      Fi ← (−Vi/0.01) ∗ Mi
    end if
  end if
  if i is finger particle then
    if j is finger particle or palm particle then
      continue
    end if
    if (ijLength + iSize) - jSize < epsilon then
      set particle collision to true
      Fi ← −Vi/0.01
    end if
  end if
end for
```

4. Evaluation

The evaluation for this system were built on the created parameterizations. These parameters are:

- Location of the object
- Mass of the object
- Size of the object
- The force of the hand once it has the object in it's grasp
- A toggle for gravity

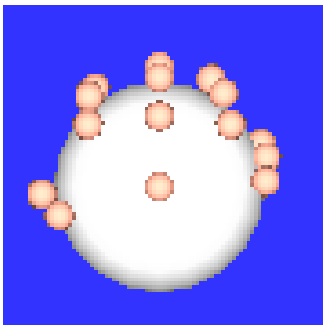
The gravity toggle was simply used test the rotation of the hand and fingers, without more complicated movements. These parameters were used to create a series of test in which the visual aspects of the system were evaluated. The qualitative aspects that were looked for in each test were

- Does the hand rotate correctly
- Does collision stop the particles appropriately
- Do the finger rotate correctly
- Does the object stay in the hand once the hand starts moving again
- Does the hand look like it is grasping the object

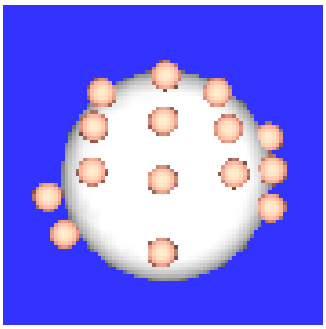
4.1. Results

In visual test to determine if the hand looks like it is grasping the object directly impacted what the final hand looked like. During the development process, the hand has only made up of six spheres, as seen in the first image as apart of the banner on Page 1. This was indented to just be used as a placeholder while the rest of the work was done. Once enough of the system was finished, more work went into the hand. Initial steps into making the hand more complicated can be seen in the second image of the banner, using shapes like cubes and disproportionately scaled spheres. But since this shapes do not bend when rotated, the hand did not reach the visual evaluation. So it was decided to stick with spheres for the creation of the hand. This also connects more with the idea of attaching this system to other systems. If a motion capture data system was attached to this, the more complicated shapes would not with the data points of the capture, a more simple version is more applicable here. Also, if this system were attached to move a hand model, the look of the underlying skeleton of that hand would not matter, so it is more appropriate for it to be as simple as possible.

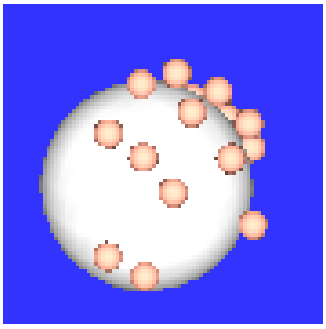
The next evaluation that met most of expectations was the object staying in the palm. This is were most of the effort went and depended entirely if the force was transferred correctly. Due to the



(a) Forward Angle



(b) Downward Angle



(c) Side Angle

Figure 2: Hand Grasping Object

added elements of the *inst* and *massAdust* components, the forces applied to the hand and the object were balanced in such a way that caused them to move at the same speed, meaning the object would never leave the hand by being to slow and falling out of the grip, or going to fast and entering the hand. Where stops working is when the force of the hand is not enough to carry the object. Above the ground this works as intended. The force of the object is greater than the hand so the object pushes the hand down. When it gets to the ground is where problems happen. In the final system the ground spring force is turned off once the object and the hand have collided. This spring forces fights with the *inst* component used to ensure a correct balancing of force. The *inst* component is used to account for the difference in time caused by the loop of the particle system, but when this is combined with the spring force, is causes a second version of the spring force to be added to the object, resulting in the object moving up before the hand was finished grasping the object. The compromise made was to turn off the ground force once the object and hand have collide was made so that the hand can still accurately pick up the object of the ground, but does not accurately showcase the scenario where the hand does not have enough force to lift it up.

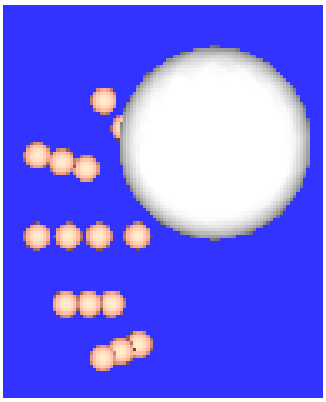
For the most part rotations of the fingers worked as intended, but had two inherent problems. First, if the object was too small, the finger particles would not collide as intended and continue rotating. If they did collide, it would not be where an actual finger would collide. Actual hands have limits on how much the fingers can rotate, either by colliding with the hand itself, or limitations imposed on those rotations by the structure of the hand. The other problem was caused by the rotation of the hand. As discussed later, the rotation of the hand did not always collide correctly, so if it was slightly of center, the fingers did not rotate at the correct speed to catch the collision of the object, resulting in the finger not rotating into the object, but missing it and continuing to rotate. But if the hand collided correctly and the object was of an appropriate size, the finger rotations behaved as expected.

The hand rotation is where the evaluations showcased faults in the system. In simple cases the hand rotates as intended, and this was where the testing has done when working on the force transferring elements of the system. But when it came time to test other angles for the hand to rotate to, the system fell apart. This was caused by how the angle to rotate to was determined. If the hand only needed to rotate in X or Y, the angle calculated was correct. But any combination of rotations resulted in the wanted angle being 90 degrees. This resulted in the hand always rotating 90 degrees in X rotation and Y rotation, regardless of where the object was. Figure **??** showcases this problem. The hand rotated to 90 degrees in both X and Y, past were it should have rotated, so when the finger start to wrap around the position of the palm particle causes the fingers not to collide were appropriate, or the miss the object altogether.

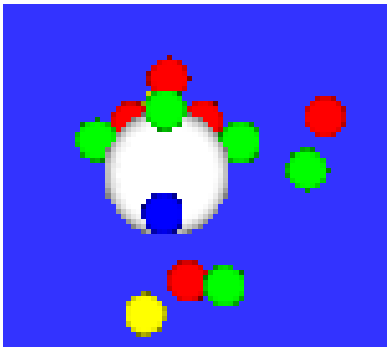
5. Conclusion

After the evaluations it is clear where most of the effort for this system went, and where more effort is needed. The inciting goal for this system was based around keeping the object in the hand, so for this system that was the force transferring mechanics. Though difficulties did occur during the force transferring development, this problems were given enough time to overcome. What was stored in the particle was revamped to include the forces acting on that particle was done to ensure forces were properly transferred, and the mass adjustment component was created so that even though the hand and the object had different quantity of force applied to them, their different masses would result in the same velocity, thus keeping the object within the hand.

That being said, this system still has a major problem, getting the hand to the object still needs more work. Not enough time was given to what ended up being a more complicated and time consuming area. The rotation approach was simple to develop and worked for simple cases, so more time could be given to the force transferring, but once the force transferring was finished, the evaluations indicated that the rotation approach was not sufficient. If



(a) Failed Hand Rotation



(b) Failed Finger Rotation

Figure 3: System Faults

future work would be done to fix this problem, a change in how the rotations are done would be needed. For example, quaternions or other rotation methods could be used as a potential fix.

However, the fault for this problem came from the initial development plan, as well as related work in this field. The transferring of forces was seen as more important, so it was given more time during the development. Less time was then given movement of the hand because it was seen as secondary, but also because of the potential of attaching this system to other system. As stated in Section 2, a majority of the related work in this area relies on motion capture for the movement of the hand, and a potential idea for this system was using it in conjunction with that motion capture to create a more accurate interactive system. Since the transferring of forces aspect works as intended, the idea of attaching this to motion capture is still well within reason, but that should not have been used as a crutch. More time should have been allocated towards the rotation of the hand so that this system could stand on its own, and not be dependant on any other system.

5.1. Future Work of this System

Independent of the fixes that needs to be done to this system, and if it was not decided to attach this to a motion capture system, there are other additions that could be made to this system. The main one would be implementing dynamic movement for the rotational aspects. Currently, the rotations use kinematic motion, so they are independent of any forces. Future development would allow the rotations to be caused by forces, resulting in an entirely dynamic based system. This would require the system to move away from the particle system, and change it into a rigid body system that could use torques to allow the rotation of the hand and fingers. The other change would be used to fix the rotation of the fingers when the object is to small. A more complex control would be used to adjust the finger rotation, or even the rotation of the entire hand, to better position itself to grasp the object not matter what size. Machine learning could also be implemented for this part so that the system learns when and how to grasp objects of different sizes, or even learn what amount of force is needed to pick up objects of different mass.

References

[AC11] ALEOTTI J., CASELLI S.: Physics-based virtual reality for task learning and intelligent disassembly planning. vol. 15, pp. 41–54. doi:https://doi.org/10.1007/s10055-009-0145-y.

[ES03] ELKOURA G., SINGH K.: Handrix: Animating the human hand. SCA '03, Eurographics Association, p. 110–119.

[HOAL18] HÖLL M., OBERWEGER M., ARTH C., LEPETIT V.: Efficient physics-based implementation for realistic hand-object interaction in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)* (2018), pp. 175–182. doi:10.1109/VR.2018.8448284.

[LXN16] LI J., XU Y., NI JIANLONG W. Q.: Glove-based virtual hand grasping for virtual mechanical assembly. vol. 36, pp. 349–361. doi:https://doi.org/10.1108/AA-01-2016-002.

[MQF*14] MOREIRA A., QUEIRÓS S., FONSECA J., RODRIGUES P., RODRIGUES N., VILAÇA J.: Real-time hand tracking for rehabilitation and character animation. doi:10.1109/SeGAH.2014.7067086.

[NK16] NASIM K., KIM Y. J.: Physics-based interactive virtual grasping. In *Proceedings of HCI Korea* (Seoul, KOR, 2016), HCIK '16, Hanbit Media, Inc., p. 114–120. URL: https://doi.org/10.17210/hcik.2016.01.114, doi:10.17210/hcik.2016.01.114.

[PZ05] POLLARD N. S., ZORDAN V. B.: Physically based grasping control from example. SCA '05, Association for Computing Machinery, p. 311–318. URL: https://doi.org/10.1145/1073368.1073413, doi:10.1145/1073368.1073413.

[ST94] SANSO R. M., THALMANN D.: A hand control and automatic grasping system for synthetic actors. *Computer Graphics Forum 13*, 3 (1994), 167–177. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.1330167, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.1330167, doi:https://doi.org/10.1111/1467-8659.1330167.

[TWMZ19] TIAN H., WANG C., MANOCHA D., ZHANG X.: Realtime hand-object interaction using learned grasp space for virtual environments. *IEEE Transactions on Visualization and Computer Graphics 25*, 8 (2019), 2623–2635. doi:10.1109/TVCG.2018.2849381.

[ZZMC13] ZHAO W., ZHANG J., MIN J., CHAI J.: Robust realtime physics-based motion control for human grasping. *ACM Trans. Graph. 32*, 6 (nov 2013). URL: https://doi.org/10.1145/2508363.2508412, doi:10.1145/2508363.2508412.

Image Credit

Hand Clipart used for the creation of Figure 1 found at http://clipart-library.com/clipart/1598316.htm