Rapid 3D Prototyping & Rigging For Animation

Challenge Problems and Resources

Developed by:
The teachers, students, and mentors in the
Gaming Research Integration for Learning Laboratory® (GRILL®) Summer 2015

TABLE OF CONTENTS

1. THE CHALLENGE

Creating a complex virtual model, like a human, can be very time consuming and often requires a level of skill and training not readily available. New technologies are creating pathways that could be utilized to reduce the time and necessary training for interested individuals to create these complex models. During this Challenge, students will utilize commercial off-the-shelf (COTS) software and hardware in order to scan, import and efficiently rig a full-bodied figure ready for animation in a virtual environment.

1.1. KEY QUESTIONS

- *What COTS game engine is most appropriate for this task?*
- *What level of fidelity is necessary for implementation of the complex human model?*
- *What is the current technology available?*
- *What is the budget available, if any?*
- *What features of the human model are most important?*
- *How are current/popular game engines modeling and animating their game characters?*
- *What is rigging?*
- *What level of detail do I need to rig to?*
- *How do I assign the rigging to be compatible?*
- *What software will be necessary to include a texture for my virtual character?*

1.2. BACKGROUND AND APPLICATION

The Gaming Research Integration for Learning Laboratory® (GRILL®), under the Air Force Research Laboratory 711th Human Performance Wing, Human Effectiveness Directorate Warfighter Readiness division, leverages COTS game-based technologies to increase the operational efficiency and effectiveness of the United State Air Force Airmen. Continuing to utilize new technologies to provide more efficient and more effective training is vital to the success of the laboratory and airmen training. We are seeking to leverage the complex behaviors of human virtual character that the COTS game-based technologies are capable of generating. The ultimate goal would be to reach a level of animation and character fidelity that would allow for effective training for a number of training audiences.

1.3. POTENTIAL TOOLS

Suggested tools include COTS game engine (Unity 5), Skanect, MeshLab, Blender, along with computer scripting or coding.

2. ASPECTS OF A SOLUTION

A solution was created during the summer of 2015 by participants in the Wright Scholar program in collaboration with the Gaming Research Integration Learning Laboratory® (GRILL®). The solution that follows was arrived at by the program participants during their nine week experience, so this is not the only way, nor is it intended to be the best way to solve the problem. The solution is given to provide guidance for future use. This section details the technology used in the solution and some of the issues encountered on the way to a solution, accompanied by documentation on how each issue was resolved.

2.1. TECHNOLOGY RESOURCES

Your students will likely choose different hardware and software to complete this Challenge Problem. Better hardware at a cheaper price, a newer version of the software, or free open-source technology may be available when your students tackle this problem. Detailed below are the tools used in the construction of this solution at the GRILL® during the summer of 2015. Do not limit your students to the specific technology mentioned in this document. Encourage students to search for a resource that allows them to continue making progress toward their solution.

2.1.1. HARDWARE

- 3D scanning device
  - This solution used Microsoft Kinect 1.0.
- Windows Machine
  - Computer running Windows 7, Windows 8, Windows 8.1, or Windows Embedded Standard 7.

2.1.2. SOFTWARE

- Scanning Software
  - This solution used Skanect Pro version 1.8.3.
- Scanning Software
  - This solution used ReconstructMe version 2.3.958.
- Scanning Software
  - This solution used 123D Catch.
- Editing Software
  - This solution used Autodesk Meshmixer version 2.9.1.
- Editing Software
  - This solution used Meshlab version 1.3.3.
- Editing Software
  - This solution used Blender version 2.74.
- Game Engine
  - This solution used Unreal version 4.0.

- Game Engine
  - This solution used Unity 5.1.1.
- Kinect Runtime Software
  - This solution used Kinect for Windows developer toolkit version 1.8.
- Kinect Runtime Software
  - This solution used Kinect for Windows runtime version 1.8.
- Kinect Runtime Software
  - This solution used Kinect for windows version 1.8 SDK.

2.2. POTENTIAL ISSUES

While working on the solution for this Challenge Problem at the GRILL® summer 2015 program, participating students encountered a number of different issues along the way. Included in this document are those issues that teachers or students who tackle this problem in the future would also likely encounter. If your students approached the Challenge Problem using different tools, they may not experience these exact issues. Issues are ordered based on when they arose in the process. Participating students did not document every single issue they encountered; this document includes issues that could be potential hurdles others might need to be able to resolve.

2.2.1. SCANNING SOFTWARE

When conducting research on potential 3D scanning software for the project, the initial issue was the cost of the software. Free or inexpensive software was researched, and it concluded the free scanning software programs presented many potential problems. Skanect, ReconstructMe, and 123D Catch were used for initial experiments. These options were found by determining which software programs were used on YouTube tutorials. The costly options provided better quality scans, but with the restraints in software and hardware price, we were restricted to inexpensive options. The issue with the free version of Skanect was the quality and export size, while the issue with ReconstructMe and 123D Catch was the scan fidelity. Skanect was chosen for the final scans although it condenses a 3D mesh to 5000 faces before exporting (see Figure 1: Skanect free version scan), and most of the scans were over 200,000 faces. Researching the pro version of Skanect and comparing it to other free software resolved the problem. Skanect Pro enabled unlimited faces, higher fidelity scans, and unlimited exporting capabilities (see Figure 2: Skanect Pro version scan). The pro version costs $129; compared to other online software, Skanect Pro is affordable, fast, and high quality.

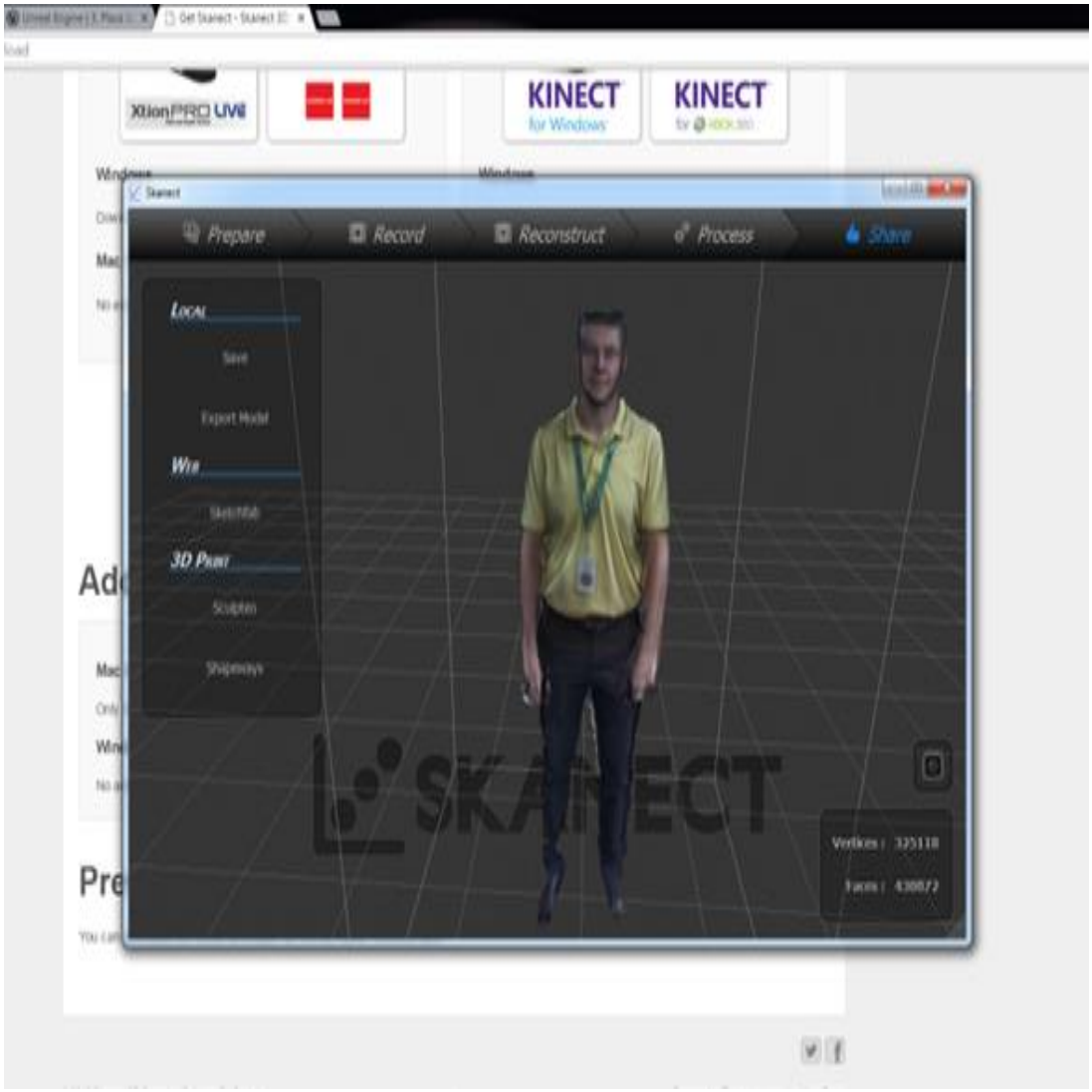


Figure 1: Skanect free version scan

After the pro version, the scans were precise and can be exported efficiently. The scan below, for example, was exported into Meshlab.

Figure 2: Skanect Pro version scan

2.2.1.1. RESEARCH CONDUCTED

To solve this issue, the main research focused on different scanning software: *Which software produces the highest fidelity? Which software is free or inexpensive? How difficult is the software program to use and how long does it take? How easily accessible is this software?*

Searching “accurate 3D scanning software” was used to gain knowledge on high fidelity models. Multiple websites contained biased information, so the best way to figure out the most accurate scanning software was to physically scan objects and compare them. Determining the information was biased after experimenting with all the software was necessary to narrow down the choices. After scanning with all the software, Skanect produced higher fidelity models than the others.

Searching “Free 3D scanning software” provided numerous software programs to experiment with. A pros and cons list was used to organize the results because it was efficient and useful (see Figure 3: Pros and Cons list).

Kinect Software

-Skanect Free

- Has a lot of scanning options-body, room, full room, or object
- Low-high Fidelity
- fusion
- can fill holes (watertight)
- made colors accurate
- very blurry and inacurrate
- range- up to 12x12x12m

-Scenect

-Brekel

- Have to pay to use

-Shapify

- Used to scan then print 3D models

-123D Catch (used on tablet)

- Used nexus tablet and downloaded app
- Took around 40 pictures of a chair
- **Took a long time to process the photos**
- Tablet took really good pictures
- Need to find a software to upload it to a software to edit the picture
- https://www.youtube.com/watch?v=NsBg-m2hrIM
- He used Autodesk 123D Catch, Autodesk Mudbox, Autodesk 3DS MAX, 3D Stitching, Iterative Prototyping, Realistic Visualization
- Have to move camera around stationary object
- If capturing a building or large object, focus on a single feature
- 40-50 pictures
- even lighting: ideal lighting is cloudy and outside

-Rendor

- App, doesn’t produce accurate models

Figure 3: Pros and Cons list

After watching YouTube tutorials on each program to determine which software is the easiest to use, Skanect was chosen because it had the most online tutorials and reviews. ReconstructMe’s online tutorials were limited and brief. Also, 123D Catch had very little help or tutorials because it is relatively new; the process for scanning for 123D Catch was confusing and time consuming. The scan below took a total of 10 minutes on Skanect (see Figure 4: Skanect scan). 123D Catch, however, took over an hour to complete an entire scan; the process

includes taking 40-50 pictures and waiting for the app to construct the mesh. ReconstructMe took 30 minutes to complete because the program froze and starting over appeared to be the only option. After experimenting with the software, Skanect was the easiest to use and the most efficient.

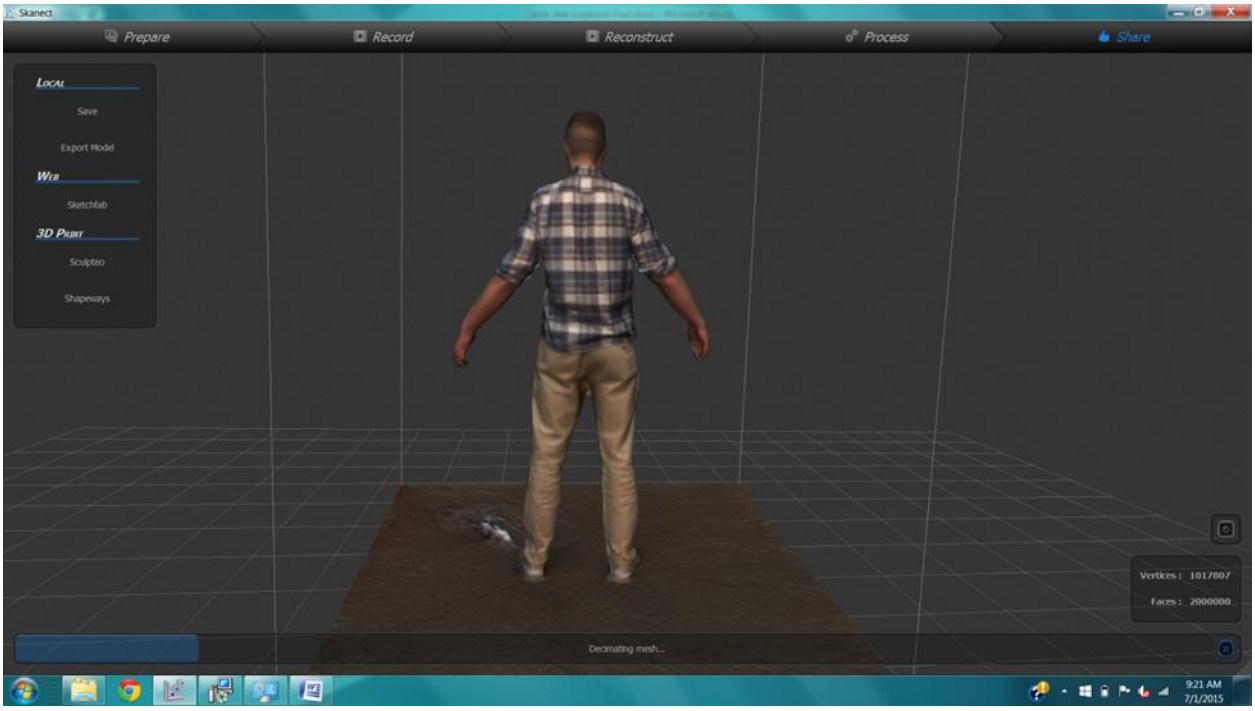


Figure 4: Skanect scan

During initial research, numerous scanning programs were available; however, they were all expensive. The price of Skanect, ReconstructMe, and 123D Catch was relatively the same. The only discrepancy was in the pro version of Skanect. It costed $129; the team from last summer used the same program, so consulting a mentor was the best way to obtain a license.

The next question that arose was the accessibility; the downloading specifications for Skanect, ReconstructMe, and 123D Catch were researched. Because it is a free app, 123D Catch is the most accessible. It can be downloaded on Apple or Android devices in a matter of minutes. ReconstructMe and Skanect, however, require Kinect runtime software. Before using Skanect, downloading the Kinect for Windows SDK and the OpenNI 2 64 bit is necessary. For ReconstructMe, downloading Kinect for Windows SDK and Kinect for Windows runtime version 1.8 is necessary for scanning. Although Skanect and ReconstructMe require Kinect Runtime Software, the online instructions provided easy steps to follow.

2.2.1.2. RESOURCES

- The Skanect website provided basic descriptions, pros and cons, and the price of each version.
  - http://skanect.occipital.com/download/

- This YouTube tutorial demonstrates the software and how to use it.
  - https://www.youtube.com/watch?v=TThFxH9U1iU

- This YouTube tutorial compares Kinect Fusion, ReconstructMe, and Skanect. Before the pro version, the final mesh had poor quality. The colors were off, the mesh had holes in it, and it did not do a complete scan. It was helpful in deciding which software to use.
  - https://www.youtube.com/watch?v=ZZrmPLas66E

- This website's instructions are used to determine the accessibility of the program. If the software was too complicated, we did not want to use it.
  - http://www.art.illinois.edu/content/resources/computer-labs/digital-output-lab/SkanectGuide.pdf

2.2.1.3. IMPORTANT CODE

N/A

2.2.1.4. RESOLUTION

After conducting research on fidelity, price, the process for scanning, and the accessibility, we concluded that Skanect is the best software for rapid 3D modeling with the parameters given. After purchasing the Pro version, the software produced the best quality models. They had a level of exactness the other software was incapable of producing. Skanect constructed fast, accurate models with little work. The other software tested required more time and effort. Even though it requires Kinect Runtime Software to be downloaded, the process for downloading is easy and clearly explained on the Skanect website. Although Skanect Pro required purchasing, it was relatively inexpensive compared to other professional 3D scanning software. In conclusion, Skanect manufactured precise 3D models most efficiently.

2.2.1.5. STUDENT REFLECTION

The process of resolving this issue required research and experimenting. At the beginning of our research, our group hit a large obstacle; we didn't have the capabilities of expensive software in the free programs available. We tried different methods of scanning in different environments to do the best we could with the software. After talking with a mentor about the

issue, we acquired the Pro version of Skanect for $129 for a single user license. With this upgrade, the scan quality and experimenting process accelerated. We developed an efficient process for scanning with Skanect Pro and achieved the goal of high fidelity models. Our group felt successful when we were able to create an efficient process for scanning that was high fidelity and low cost. The beginning of the project had many obstacles with software, but after the help of a mentor and perseverance in research, we completed our goal.

2.2.2. APPLYING PREMADE ANIMATION FILES TO BLENDER FILES

While trying to animate 3D modeled images in Blender, an issue occurred while applying imported BVH files to the character mesh from the scan. When the premade armature from the BVH file was attached to the mesh, the model was deformed in an unexpected way. The rest position looked perfectly fine, but when the armature was in pose position the mesh lost its figure (see Figure 5).

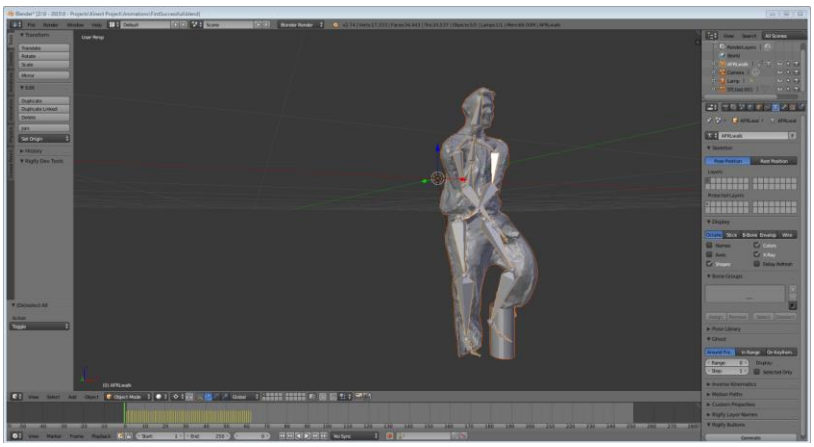


Figure 5: The deformed mesh after the armature is applied

2.2.2.1. RESEARCH CONDUCTED

The issues involving applying animations files to polygon character meshes were resolved by using a process of researching and trial and error to achieve a completed animation. The first question was: *Why is the mesh model different than the animation skeleton model?* Searches were conducted about the file types used in importing and exporting such as STL, OBJ, PLY, and FBX file types. The STL, or Stereolithography, file type describes the three dimensional geometry of a Computer Aided Design (CAD) object without any reference to texture, coloring, shadows, or other CAD model attributes. The OBJ, or Wavefront, file type describes a three dimensional object by listing all of the vertices and edges with their coordinates. The PLY, or Standford Triangle Format, stores the spatial data of the flat polygons that form a mesh. The

FBX, or FilmBox, file type describes stored two dimensional, three dimensional, audio, or visual data primarily used in AutoDesk products. The next path was to look into moving the armature, or joint structure, in the first frame of animation. It was assumed that the rest of the frames would result in a normal walking motion.

The method of changing bone roll angles and moving individual bones was tested; it involved adjusting the individual bones to change the way the animation moved. This method however, was ineffective in creating the desired walking animation and the resulting animation was unrecognizable. The next approach was to move the polygon mesh to match the armature structure. This idea was inspired by an article found in a Google search, researching how to retarget BVH files in Blender. A better looking animation was provided by this new approach, but some problems remained. The question became: *How can a model be edited to form a T-Pose?* Blender modifiers were determined to be ineffective after experimentation as they provided a well deformed model, but were prone to create graphical glitches while running the animation. The mesh around the arms and shoulders was cut which resulted in a normal looking model from a distance (see Figure 6).

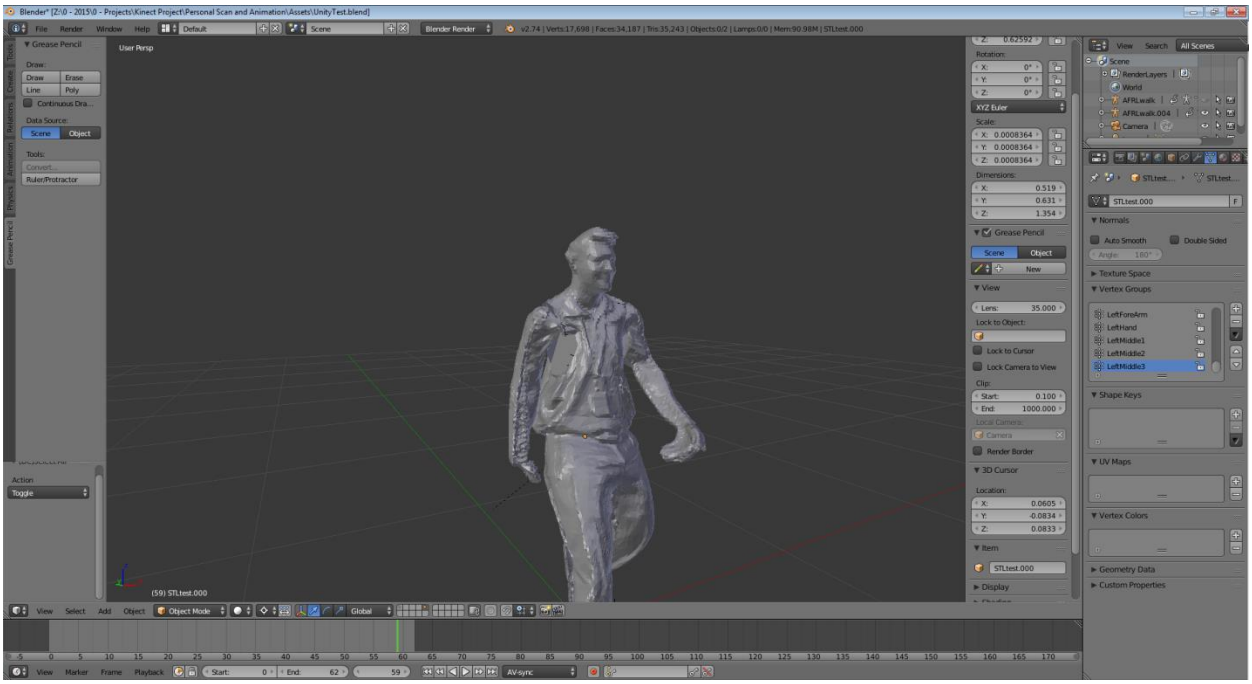


Figure 6: A model with broken shoulders

The final approach was to create a scan of someone who was already in a T-Pose. This resulted in a model that was properly rigged (see Figure 7). The skeleton mesh was then imported into a game engine to utilize the animation.

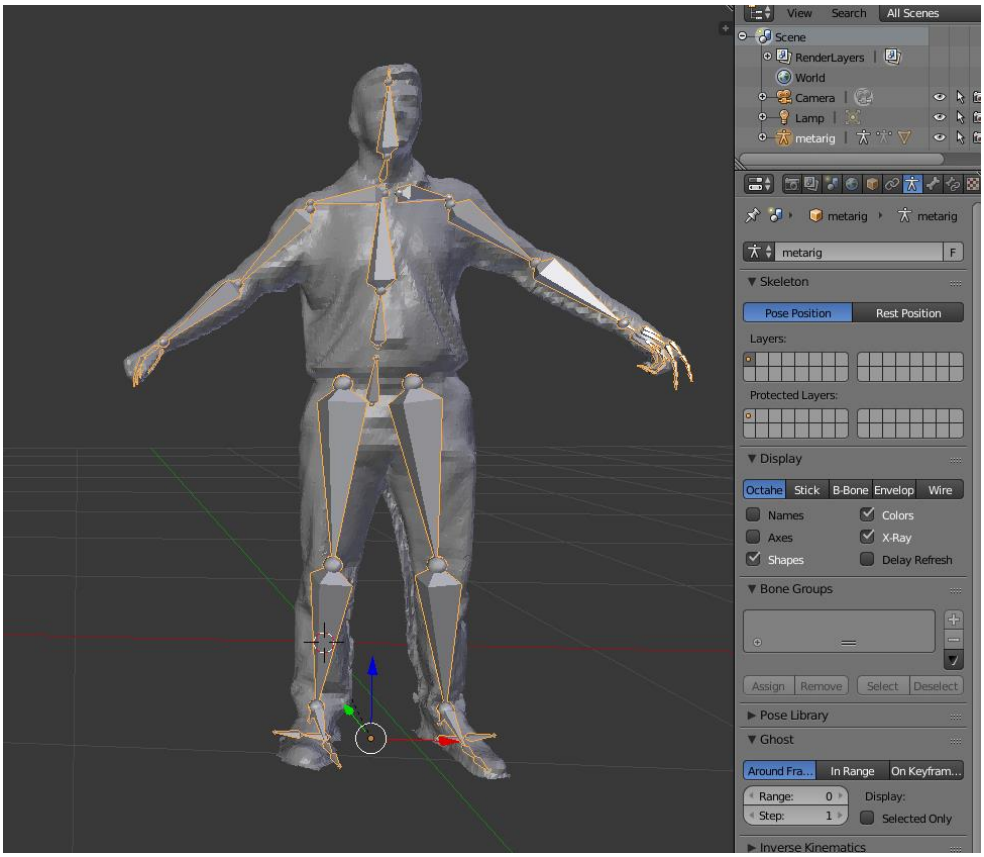


Figure 7: A model rigged correctly

2.2.2.2. RESOURCES

- Explanation of why exports from Unity are scaled incorrectly:
  - https://jibransyed.wordpress.com/2014/06/05/how-to-correct-scale-and-rotation-of-static-blender-models-for-unity/.
- A tutorial on how to take an existing BVH file animation and attach it to an existing character mesh:
  - http://www.3dartistonline.com/news/2011/03/free-blender-tutorial/.
- A description of Stereolithography file format:
  - http://bastech.com/sla/techtips/stlfiles.asp.
- An in-depth article on the Wavefront file format:
  - http://www.martinreddy.net/gfx/3d/OBJ.spec.
- A long description of the Stanford Polygon Format:
  - http://paulbourke.net/dataformats/ply/.
- Blender documentation on FilmBox file format:
  - http://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Import-Export/Autodesk_FBX.

2.2.2.3. IMPORTANT CODE

N/A

2.2.2.4. RESOLUTION

1. Determine the cause of the mesh deformation.
2. Find research about Blender importing and exporting through Google.
3. Test functionality in Blender to learn how things work.
4. Watch Youtube tutorials on how to use functionality in Blender.
5. Adjust bones manually to correct animation. This takes more time than creating a new animation.
6. Move mesh to accommodate the armature instead of moving the armature. Use modifiers and proportional editing to move mesh to the T-Pose. (We could not find a way to do this efficiently.)
7. Final solution, scan model in T-Pose (see Figure 8).

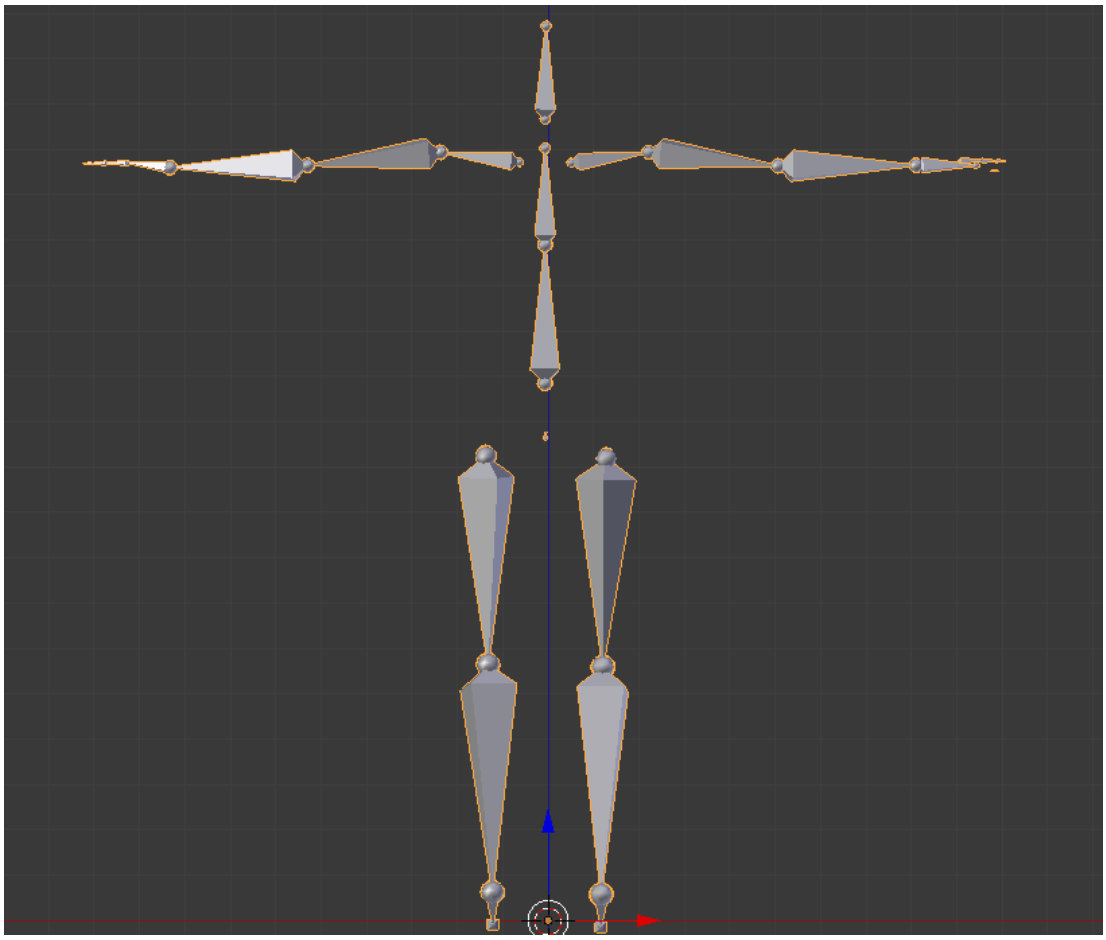


Figure 8: The ideal T-Pose

2.2.2.5. STUDENT REFLECTION

We started by focusing on the issues of retargeting an animation file to a scanned polygon mesh. To do this we had to learn more about Blender so a lot of time was spent watching tutorials and learning Blender. By immersing ourselves into the software, we learned a decent amount of the basic functionality. This helped us understand the meaning of the terminology used in the tutorials. We then tried to edit the mesh which allowed us to have our first success, making our first animated model. The issue remained, however, that parts of the mesh were broken or graphical glitches and we decided to adjust the way we scanned the model to give a character mesh that we could edit more easily. We realized that we were trying too hard to make edits using software, when we could much more easily change the modelled position by moving the model's arms into a T-Pose.

2.2.3. RIGGING SKELETAL MESH FOR ANIMATION

When using imported humanoid models for animation, a skeletal rig must be applied to the mesh in order to allow for the bending and twisting of muscles that occurs when an animation is applied to a model. When importing meshes, however, the rigging system built into Unity is extremely sensitive, and, more often than not, it incorrectly assigns the rigging to the humanoid so that the animation and movement appear abnormal.

2.2.3.1. RESEARCH CONDUCTED

Attempting to fix the incorrect rigging of the models proved fruitless; after numerous hours of failed experimentation and internet searches, manually rigging the skeletal mesh through a third-party program, such as Blender, proved to be the only work-around for this situation. Since Unity's importing system for humanoid figures relies on its internal algorithm for determining the mesh's skeletal configuration, Unity often incorrectly assigns bones within the body (see Figure 9).

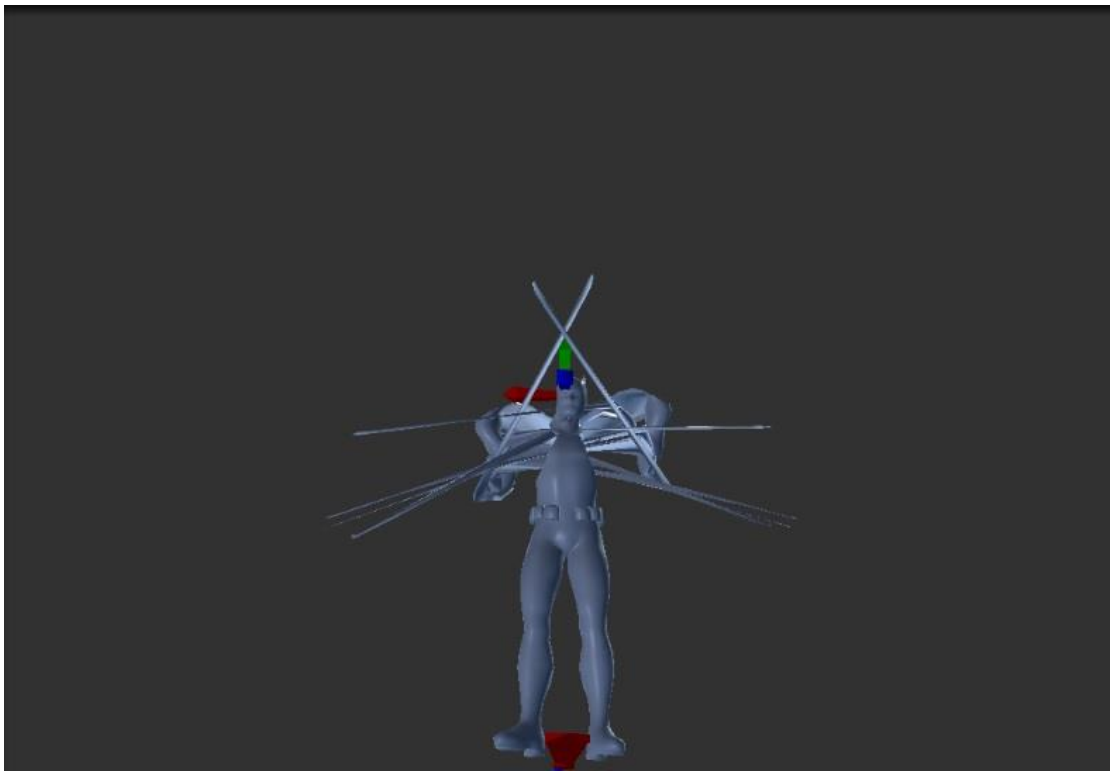


Figure 9: Batman Model with Incorrect Rigging

This incorrect assignment of the skeletal rigging leads to modular distortion and stretching of portions of the mesh, which looks completely different when compared to the correct rigging (see Figure 10).

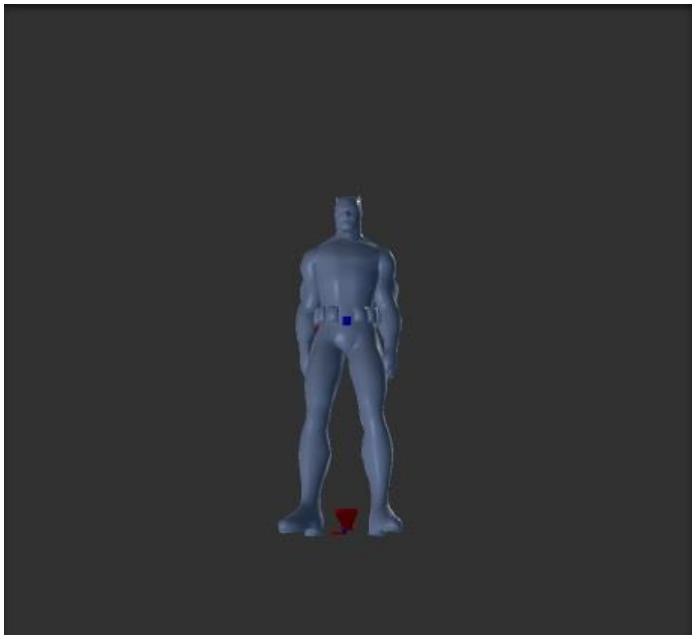


Figure 10: Batman Model with Correct Rigging

In the end, however, no method of manually fixing this issue in Unity arose. If a model was already correctly rigged and meshed in the third-party program, it was correctly exported to Unity's automatic rigging application. Thus, as a workaround, all models were correctly rigged in external applications, such as Blender, prior to exportation.

2.2.3.2. RESOURCES

- This document described the properties for configuring the rigging of an imported FBX file.
  - http://docs.unity3d.com/Manual/FBXImporter-Rig.html
- This document thoroughly described how to use the Avatar Configuration function.
  - http://docs.unity3d.com/Manual/ConfiguringtheAvatar.html

2.2.3.3. IMPORTANT CODE

N/A

2.2.3.4. RESOLUTION

In the end, since no obvious solution to the problem existed, the most efficient workaround was to pre-rig the model in an external application such as Blender or 3DS Max, and Unity's automatic rigging application was abandoned. While Unity's Avatar Configuration application allows for manual reconfiguration of certain bones, the bones that are manually configured often do not match the location that it belongs to on the body, and the engine also requires precise movements to specify where the muscles and bones are in the body. Additionally, Unity will only apply the rigging if it believes that the bone is in a valid location, which makes the engine extremely difficult to use. Overall, the models must be pre-rigged before imported for the best results.

2.2.3.5. STUDENT REFLECTION

We tackled the issue of skeletal rigging compatibility in Unity through collaboration with another team working on a similar issue. While changing configurations for the bone rigging in Unity, I became increasingly frustrated at how difficult it was for the bones to be perfectly aligned with the body. Additionally, the locations for the bones that Unity accepted were often incorrect, and as such, I determined that configuration of the humanoid rig through Unity was both impractical and time-consuming. After hours of fruitless experimentation into moving around joints in Unity proved ineffective, I eventually settled for simply importing the mesh

with the skeleton if it was compatible with Unity's humanoid rigging in the first place. If it was not compatible, I found no effective way to modify the mesh itself within Unity. There may be very few details on how the original model was created through an external program. However, I was simply importing models created by somebody else, and as such, I had no background knowledge on which modeling programs were most suited to this situation. Since working within Unity's rigging application was extremely difficult, I simply concluded that the model had to be rigged by somebody else in another application before I imported it into Unity.

2.2.4. CREATING ANIMATIONS

Once a model is created and rigged, it is ready for animation. Animation can be done in most 3D modeling software and isn't too difficult. However in order to create a satisfactory game character, numerous animations are required. This process of animating is time-consuming and meticulous.

2.2.4.1. RESEARCH CONDUCTED

Google was used to search for any premade animations or motion captures that may exist on the internet. After searching, ready-to-use motion capture files were found through Carnegie Mellon. These files can be imported into a 3D modeling software and rigged to fit the model. Since fitting all the animation files would still be pretty tedious, an alternative, quicker way to animate was researched. Searching through Google for "online animation software" yielded a fairly decent solution to the problem. Mixamo is online animation software where a model, even without a skeleton, can be uploaded, joints can be assigned to the model by the user, and the software will automatically rig the model, plus there is a large cache of premade animations to select from. However, Mixamo does have its drawbacks. Creating an account is free, but only the first 10 auto-rigs and 20 animations are free.

2.2.4.2. RESOURCES

- Carnegie Mellon's motion capture lab provides numerous files to be downloaded and used in character creation.
  - http://mocap.cs.cmu.edu/search.php?subjectnumber=%&motion=%
- Mixamo is a useful tool that can be used for multiple purposes in the character creation process.
  - https://www.mixamo.com/

2.2.4.3. IMPORTANT CODE

N/A

2.2.4.4. RESOLUTION

Both the Carnegie Mellon motion capture site and Mixamo found through research were viable solutions to the issue and each was tested. Carnegie Mellon's motion capture files proved to be more difficult to use than previously thought. The model needs to be rigged identically to the animation file in order to have acceptable results. In addition to this dilemma, this solution takes longer to implement than Mixamo. Mixamo was the final solution chosen for animation and creating a character to be used in a game engine.

2.2.4.5. STUDENT REFLECTION

We first realized an alternative was needed to hand-animating the model when it took hours to make one good animation and one of the main criterion we focused on was time required to make the animations. Upon initially finding the Carnegie Mellon motion capture files, we thought our problem was solved entirely. Yet, once again we had found that implementing this solution turned out to be problematic. Frustrated, we searched for another route to animation and found Mixamo which we thought was great software. Within minutes we had a viable model and could add hundreds of animations to it from Maximo's collection. We saw that there is a limit on free auto-rigs and animations so we were conservative when it came to what we used Maximo for. In the end, we saw Maximo as the easiest solution to rapid animation and the most efficient way to quickly and fully animate a model for use in game engines.

2.2.5. ANIMATION IMPLEMENTATION

When attempting to add animation to various characters within Unreal Engine, difficulties were found within the game engine implementation process. Unreal Engine's animation system contains a wide variety of tools and utilities that can aid in the animating process, but unless a complete understanding is attained beforehand, the task may seem overwhelming. Figure 11 displays a fully animated character.

Figure 11: A fully animated character within Unreal Engine

2.2.5.1. RESEARCH CONDUCTED

After animations were attained with Mixamo's online animation software, the next step in the process was to add the animations into the game. Though the idea sounds simple in theory, the process itself created many unexpected roadblocks that had to be tackled.

An initial look at the numerous online tutorials for Unreal Engine, led to the knowledge of its animation blueprints and their necessity in the process. The animation blueprint utilizes the basic blueprint format that Unreal is famous for to create a series of true-false parameters to control the character. By doing so, the characters can respond in real time and act independently of the main player. With the aid of Unreal Engine's online tutorials, the creation process became relatively smooth. Within the blueprint itself, a "State Machine" was created to control these true-false variables, saying that if one condition was true, play one animation, and if another was true, play a different animation. Overall, even though the layout of the interface took some time to learn, the implementation process was relatively smooth and simple.

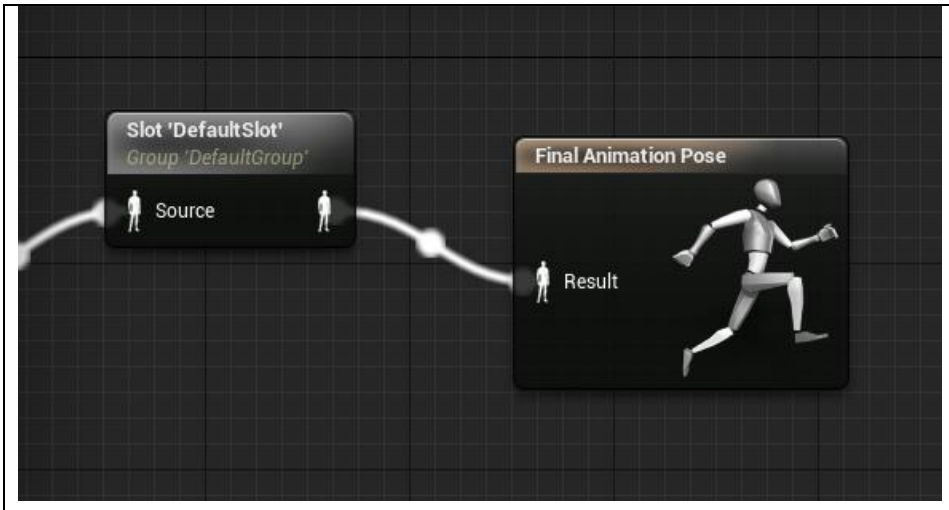
2.2.5.2. RESOURCES

- Unreal Engine's YouTube tutorials:
  - https://www.youtube.com/playlist?list=PLZlv_N0_O1gaCL2XjKluO7N2Pmmw9pvhE

2.2.5.3. IMPORTANT CODE

The true-false variables are located within the “DefaultSlot” State Machine located on the left, and they determine the end animation pose.

Table 1: Blueprint for final pose



2.2.5.4. RESOLUTION

Because previous work had been done with Unreal’s blueprint system, the process of animation itself is relatively quick. A simple import into the engine will provide the animations, and the animation blueprint itself will determine the actions of the character. Though the animations don’t always have smooth transitions from one to another, it is the simplest way to provide a fully functioning character within Unreal Engine 4.8.2.

2.2.5.5. STUDENT REFLECTION

When the idea of adding animation into the game first came to mind, it seemed like it would be quite the tedious task. We figured we’d have to fully animate our characters ourselves, and we thought we’d have to do it in a frame by frame process. We were more than relieved to discover Mixamo, as it completely eliminated the need for time consuming animation, and it allowed us to go straight to the implementation. From there the only issue we had was time, as the true-false statements, the importing process, and the multiple cross-blueprint references can all be lengthy tasks. Yet in the end, through the use of online tutorials and mentor assistance, we were able to figure it out.

2.3. PROBLEM SOLVING TIMELINE

Scanning and Animating Timeline

The Challenge

In this Challenge Problem students will be expected to use a variety of technology skills in order to create a realistic 3D game that will be used to examine go-kart performance in a specific track design.

Finding Effective Scanning Software

In this issue we struggled finding viable scanning software to utilize the functionality of the Xbox Kinect. Additionally, the only software that we found effective were the paid versions of some of the software.

Animating a Character Mesh Using Blender

We had difficulty using Blender to animate the scanned 3D meshes using armature. First, it took a long time to assign the character a full rig and create animations. Second, we found it difficult to assign an armature to the mesh. Lastly, we encountered difficulty attempting to deform the mesh to the character.

Difficulty Creating Good Animation

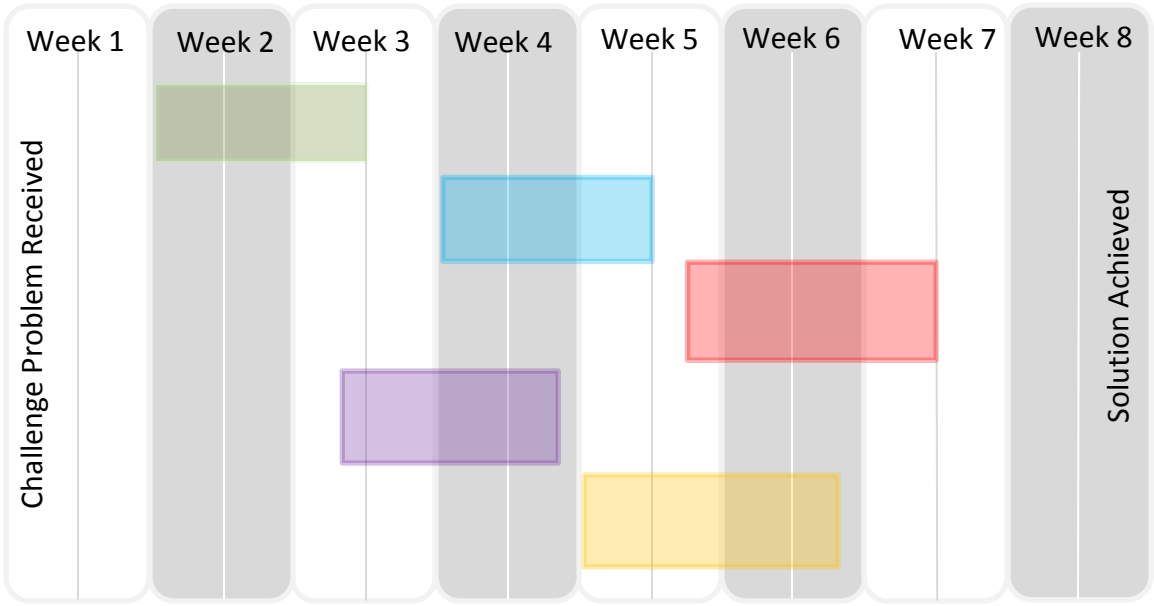
We realized the time requirement involved in creating animations would be a great detriment to a rapid 3D modelling process.

Trouble Opening Doors in the Unity Engine

The first main issue encountered while using the Unity Game Engine, was the difficulty of opening the doors on the Tec^Edge Building. We had difficulty developing an animation that would smoothly open a door without artifacts or other graphical effects.

Creating and Using Animation in the Unreal Engine

We found animation much more difficult in Unreal than in Unity, due to different mechanics. The animation requires a great deal of research into UE4 (Unreal Engine 4) mechanics.



Week 1
- Learned basic modeling skills on Sketchup
- Received Challenge Problem assignment
- Group discussions about roles

Week 2
- Started experimenting with Blender and Game Engines
- Looked for viable scanning software

Week 3
- Began work on basic animation in Blender
- Solidified choice of scanning software
- Discovered Unity Engine's built-in compatibility with armatures

Week 4
- Began work on writing code to develop more applications of the Kinect sensors
- Attempted to attach motion capture files to scanned models

Week 5
- Began work on final presentation
- Utilized more effective mesh cleanup techniques
- Researched more alternatives to hand animating in 3D modeling software

Week 6
- Began work on further applications of Kinect sensor
- Began using MIxamo to animate models

Week 7
- Determined viability of more Kinect functions
- Implemented MIxamo animated models into game engines

Week 8
- Wrapped up of Challenge problems
- Final tweaks on games developed

This timeline does not depict all of the issues students encountered as they developed the solution during the GRILL™ summer program 2015. Other issues arose during the problem solving process and were resolved within the given timeline.