
INSIGHT:
CREATING ENEMY SOLDIER UI v1.0

TABLE OF CONTENTS

3 INTRODUCTION

4 Scenario
5 Basic Workflow
6 Last Preparations

7 MODELING

8 Getting Started
9 Creating Torso And Arms
10 Attaching The Limbs
11 Creating The Boots
12 Creating the Hands
13 Finalizing The First Pass
14 Adding Detail
15 Refining The Head
17 Finalizing The Second Pass
18 Testing Deformation
19 Adding Asymetric Detail
20 Creating Accessories
22 Wrapping Up The Model

23 CHARACTER RIGGING

24 Getting Started
25 Creating The Leg Controls
26 Creating The Back Controls
27 Creating the Arm Controls
28 Adding The User Interface
29 Skinning The Character
30 Create Weapon Controls
31 Finalizing The Character Rig

32 TEXTURING

33 Applying A Default Texture
34 Creating The UV Layout
35 Creating The Camouflage
36 Adding Detail
37 Creating Variations
38 Creating The Boots
38 Creating The Faces
39 Creating The Rifle

40 WRAPPING UP

41 Creating Additional Heads
41 Modifying The UI
42 Troubleshooting The Rig

43 APPENDIX A: TEXTURES

51 APPENDIX B: USER INTERFACE

PART I

INTRODUCTION

I've been asked a lot about how I create my models and character rigs.
So instead of repeating how and what I do over and over again, I decided to start a new project and document my workflow step by step.
This is not to be seen as a deep tutorial of any kind rather than as an insight as to how I approach a character modeling task.
This document assumes the reader already has a fundamental understanding of 3D graphics, in particular Maya, as well as an understanding of 2D Image editing software, namely Adobe Photoshop.
Steps I'm taking in creating this character will at times be summarized or skipped if seen as unnecessary to show the progress of the project.

SCENARIO

I'm going to create a soldier character that can be used in a modern type war game as an enemy combatant. I'm setting the limit at 4000 polygons. This includes both the main mesh as well as several accessories like guns, helmets and grenades.
Additionally, I want to create a set of three different heads to choose from, to further individuality in the characters if used multiple times in the same environment.
Also, the camouflage uniform will come in different colors. One of the variations will be kept simple grayscale, to allow vertex coloring to put the uniform into environmental context (e.g. using a green vertex color for jungle camo, or using a blue vertex color for night-ops camo, etc.).
Same approaches will be taken for the different faces to mimic a variety of ethnicities.
This character will consist of three textures:
One 512x512 texture for each of the uniforms, one 256 x 256 texture for each of the heads and one 256 x 256 texture for places like the shoes that should not be affected by texture changes.
Additionally, each interchangeable item will have its own texture with a resolution depending on its relative size. All this is done to create a vast arsenal of characters with a minimum amount of textures and polygons.
For the textures I will use a combination of photographic resources and hand painted material modified to fit this project.
The character rig should be made up of 35 Joints or less. This will include the character as well as any equipment he will be carrying. I'm setting about two joints leeway, if absolutely necessary.

Let's summarize:

Human Character
4000 Polygons for Body and Accessories
One 512x512 texture for each uniform variation
One 256 x 256 for each face
One 256 x 256 texture for all other areas of the body
Five textures, one for each non-skin item ranging from 128x128 to 64x64
35 Joints or less

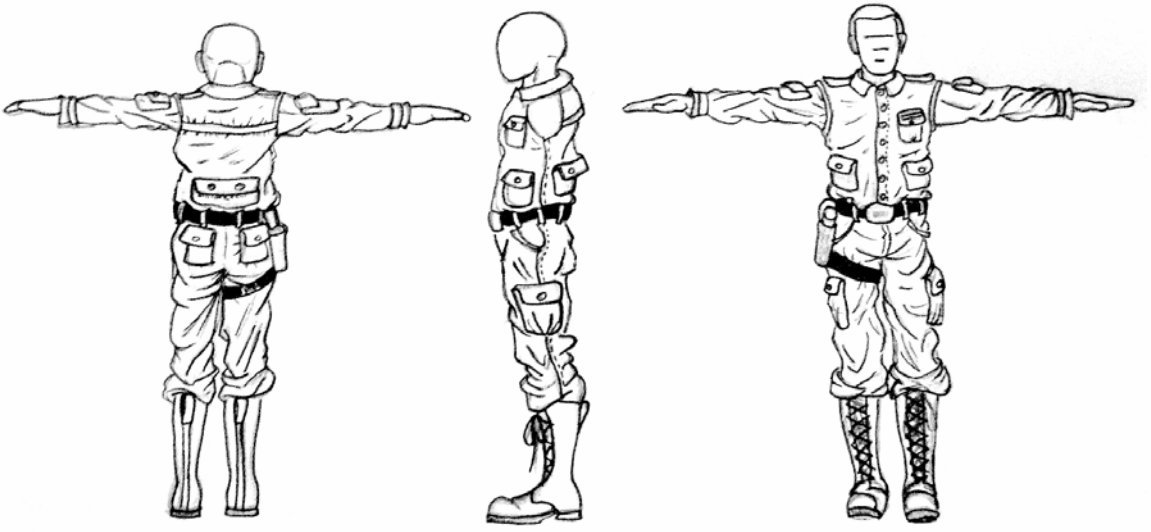
The time frame I'm setting for this project is 50 hours from model to finished character rig.
Broke down, this means 15 hours modeling, 15 hours texturing and 12 hours for the character rig. This leaves another eight hours to spend where needed.

BASIC WORKFLOW

I'm generally working in several different layers on projects of this nature.
First, I create the rough shape of the character.
Next I will go over the different parts one by one several times, adding little detail during every step.
This will prevent me from adding too much geometry in one area that would bust my budget.
Around halfway through the modeling part, I will create a simple skeleton to help me see the areas that will not deform correctly.
I can then correct these problem spots at a relatively low level of detail.
Generally, I found it to be a more efficient approach to create the character rig before I tend to the textures. One reason for this is that I can still modify the geometry at this point without destroying or awkwardly stretching the textures.
Basic game pipelines might differ from my approach, but if implemented correctly, this way will allow animators to use the character a lot earlier in the timeline.

LAST PREPARATIONS

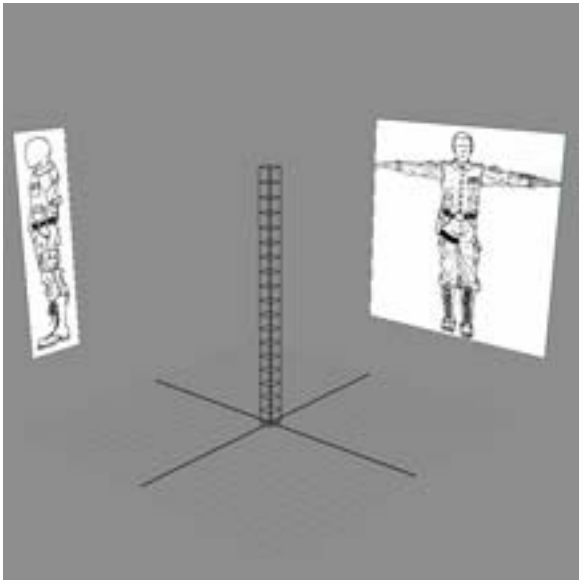
Let's take a look at the character sheet. The character has no distinguishable face due to his purpose as a NPC.



For the faces I will use photographs as a guide.
But for now, I will start with the modeling of the body.
The clock is running, let's go...

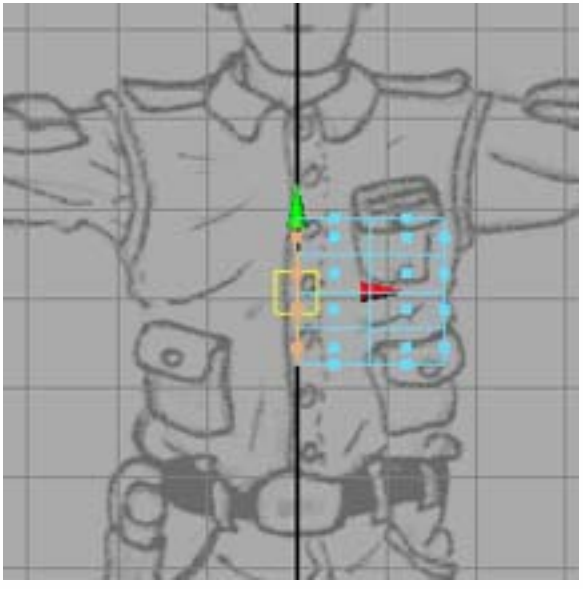
PART II

MODELING

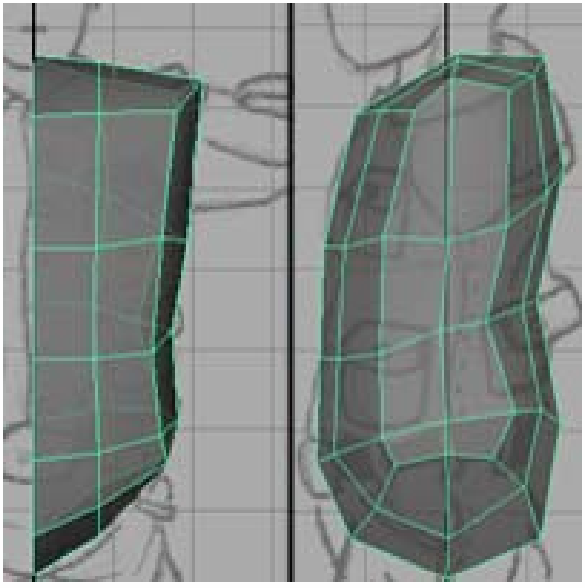


First, I place image planes as visual reference in my workspace. The character will be around 180 cm, so I create a poly box measuring 18 units in height as a primitive size chart.

During the next steps of modeling, I will make heavy use of the SPIN FACES Mel Script, the MJ Poly Tools 1.3 script as well as the standard Maya tools split polygon, merge vertices and extrude face.

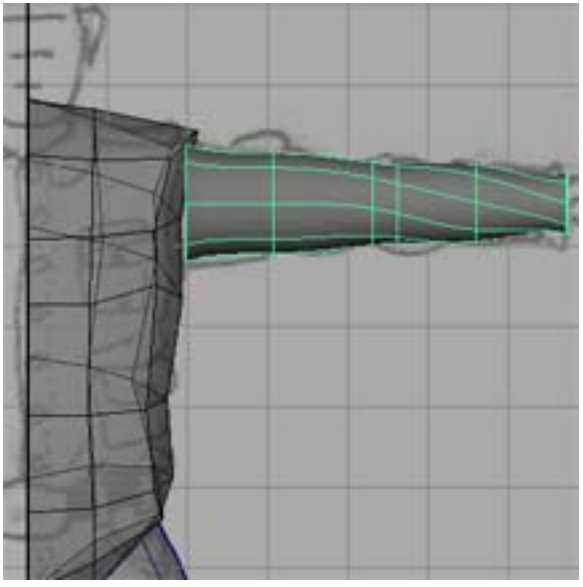


I start the character by creating a simple box with four subdivisions in height and two in both width and depth. I make sure the innermost vertices are snapped to the center grid, and delete the adjacent faces.
I will now begin to roughly block in the left side of the torso area.

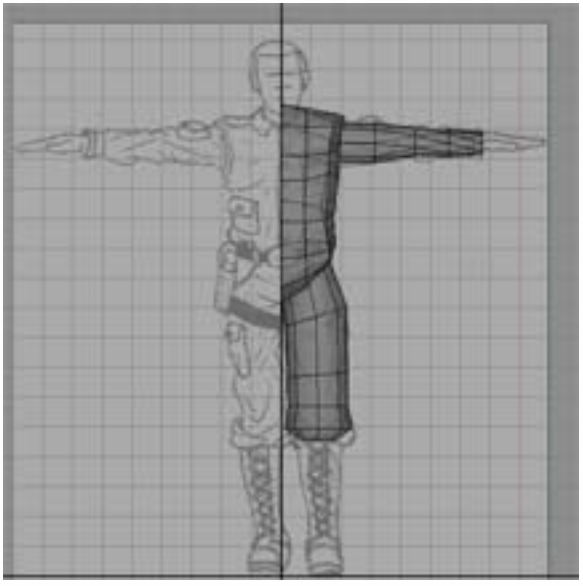


Using the front and side views I start to define the form. I try to keep the polygons as square as possible at this point. Having used all the geometry available to define the rough shape I will now begin to add extra detail.

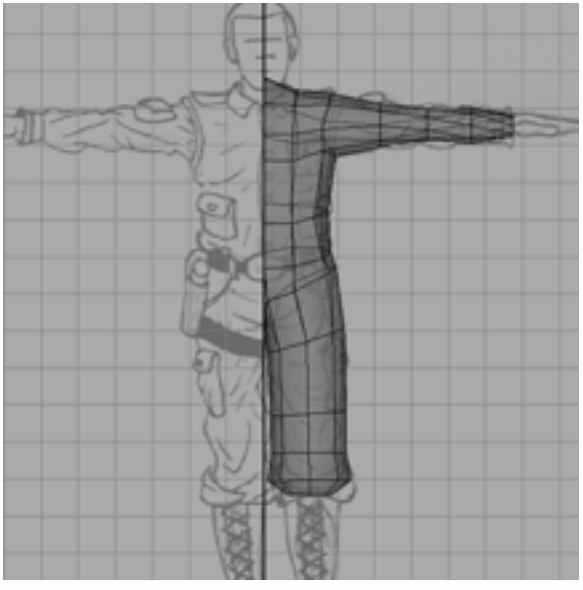
My tool of choice for the next steps is the Edge Loop Split feature in MJ Poly Tools, which will give me a clean new row of vertices to work with. (I prefer this approach over the results I would get by manually splitting the polygons.) The arms and legs will both consist of eight vertices in the circumference once I'm ready to attach them to the torso. The hip area already contains enough detail for a seamless attachment, but I will have to add another row around the arms.



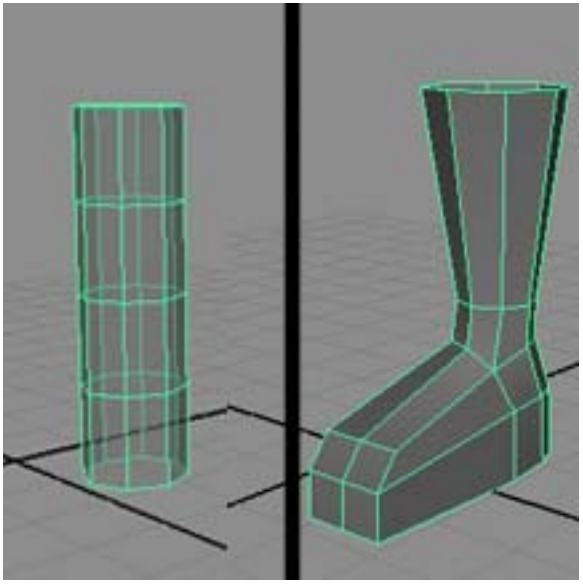
For this project, I will block in the arms and legs with simple NURBS cylinders using eight spans, adding isoprams where I need them.
For the arms, I like to add a little rotation to the geometry below the elbow.
In undressed characters, this will mimic the flow of the extensor muscles of the lower arm, but even for the long sleeves used in this project it will give me more desirable deformations when animated.



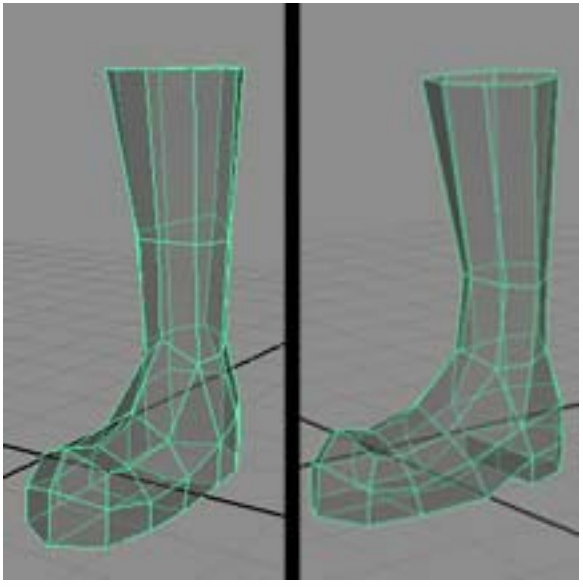
Once I get close to the desired shape I convert the NURBS objects to Polygons.
After I adjust the resulting geometry, I can continue attaching the extremities to the torso.
(I find it easier at times to use this approach rather than strict box modeling to achieve an organic form with the cleanest topology possible. Due to the square nature of NURBS it also provides a clean UV layout for the arms and legs that might be easier and faster to tweak than laying out the UVs manually later on.)



I delete the faces of the torso where the arms and legs will be attached.
Next I snap the vertices of the arm to their corresponding spot on the torso and do the same for the leg. Then I combine the objects and merge the vertices to get one single mesh.

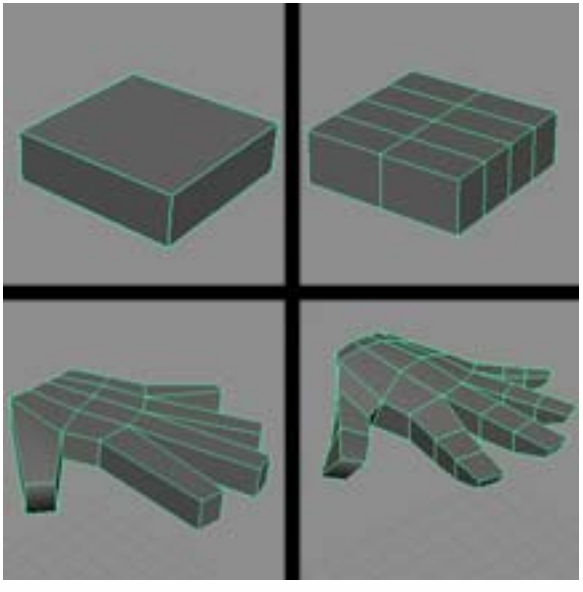


The next task is to create the boot.
I start with simple poly cylinder with four subdivisions in height and eight subdivisions in circumference.
I extrude the four foremost faces to create the tip of the shoe.

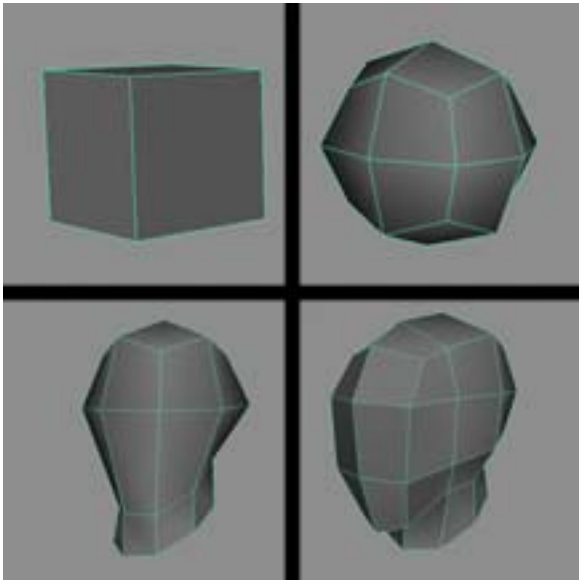


I continue adding rows of vertices to define the shape of the boot and extrude the edges at the back to create the heel. After closing the hole at the bottom of the heel the first stage of the boots is finished

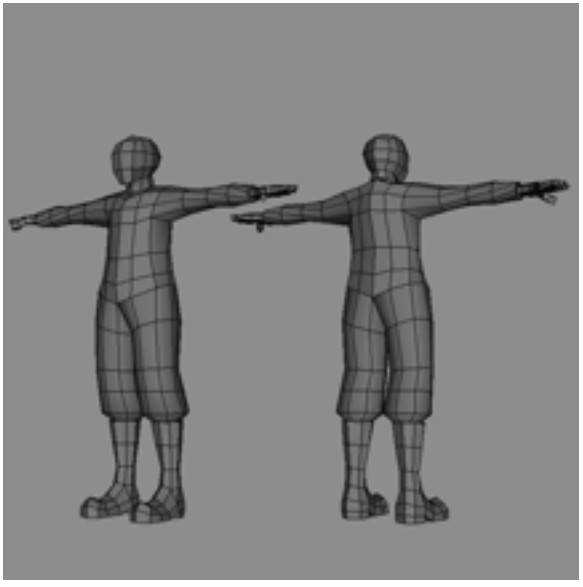
Next up are the hands.
Due to the joint limitations of the project, it is highly unlikely that the fingers will be animated, so keeping them as simple as possible is a great way to save both time and polygons.
Since the hands are a pretty straight forward part, they were not included in the character concept art.
I will use a photograph as reference to make sure the proportions are correct.



I start by creating a polygon cube.
I modify the cube to be four subdivisions in depth, two in width and one in height.
Next I extrude the fingers using an extra row of vertices only at the joints. After some basic refining, this stage of the hand is done.
I will leave the fingers in this pose for now to allow simpler UV mapping later on.
For the finished character, the hands will be put in a pose that allows grabbing things without the need to animate the hands.



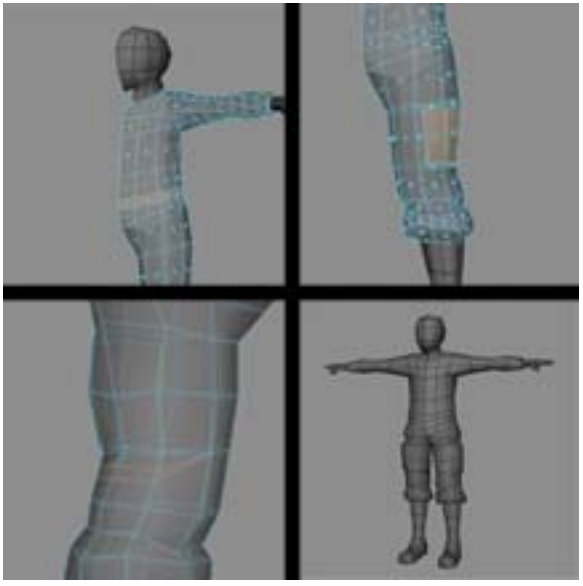
Next up is the head.
Since this project requires me to model three different heads, I will start by creating one simplified head at roughly the same level of detail as the rest of the body, and use this mesh to create the individual ones in the next stage of modeling.
For the first step, I create a simple poly cube and smooth it one level.
I extrude the front faces to create the face area and start refining the shapes.
I add extra geometry where needed.



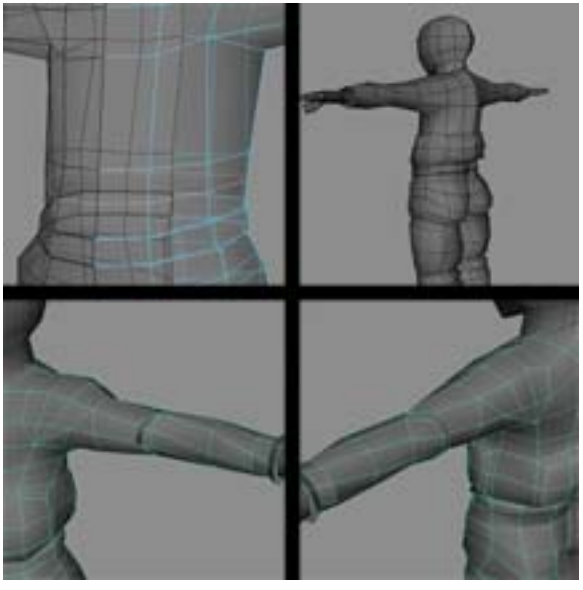
This concludes the first stage of modeling, the blocking of the shape. The next step will be to add detail, like wrinkles and pockets as well as creating a natural flow of geometry to assure correct deformation during animation.

Let's look at the budget:

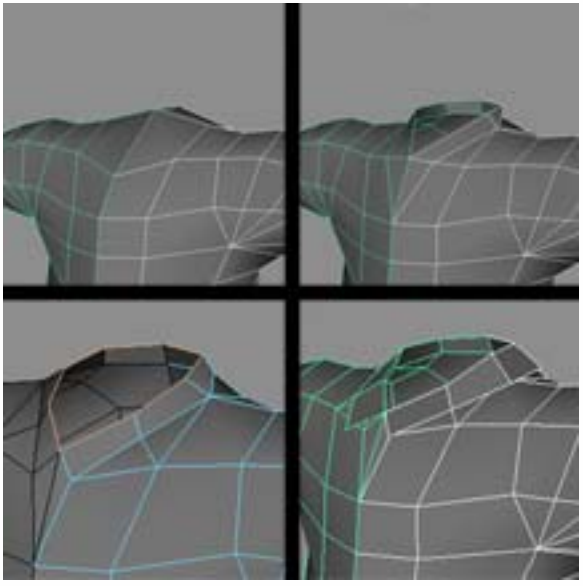
So far, I've used 1152 triangles, and the model up to this point took just about an hour.



After I define the belt line, I start adding detail to the pants. I extrude faces to form the pockets on the side. I add extra rows of vertices around the crotch and knee areas. This will create wrinkles to the mesh as well as provide means for more accurate deformation of these problem areas.
As a last step at this point, I extrude the faces around the belly to create the belt.



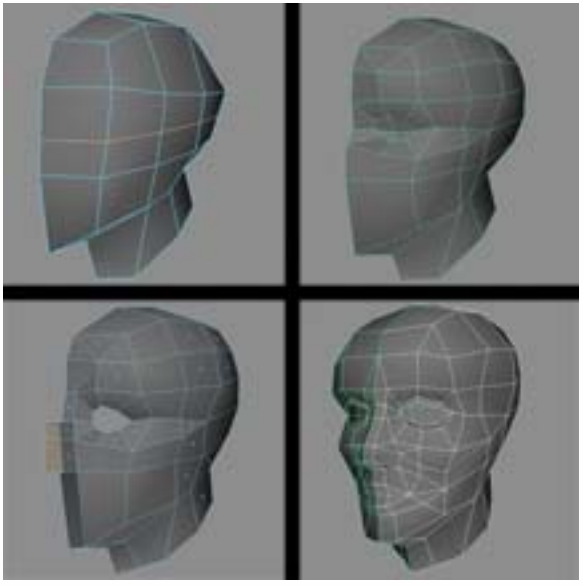
For the upper body, I start by adding a new row of vertices to create the bulge where the shirt is tucked in. Extra geometry is added to create the pockets at the front and back of the character.
I define the area around the shoulders and elbows to allow better deformation of the mesh.



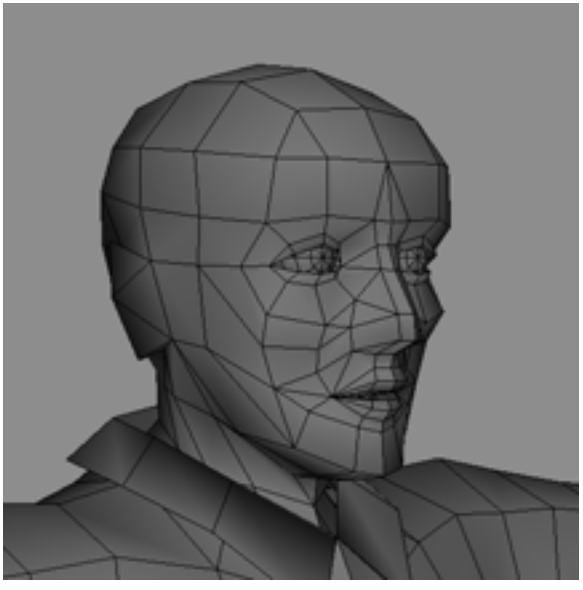
I reshape the area around the neck to define the collar. Since in this scenario, the head will be a separate mesh, I don't have to worry too much about continuity between the shirt and the neck.
I can therefore add extra geometry to achieve the desired shape.
Note that I haven't added the pocket on the left chest yet. Since I'm only working on one side of the model at this point, I will delay adding asymmetric detail until I combine both sides to form one single mesh.

At this point, both the shoes and the hands still provide a sufficient level of detail to fit the requirements. Thus I will concentrate on adding facial features to the head.

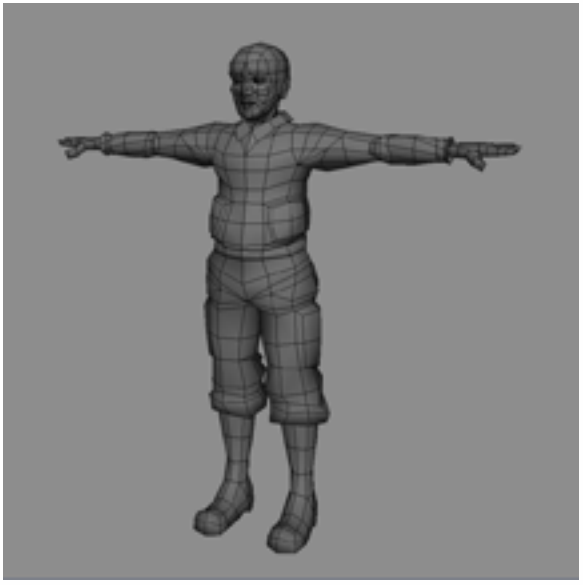
The face will still stay fairly generic. I'm using anatomic drawings as reference to place the eyes, nose mouth and ears in relative proportion.
Individual faces will be derived from the basic mesh created in this step.



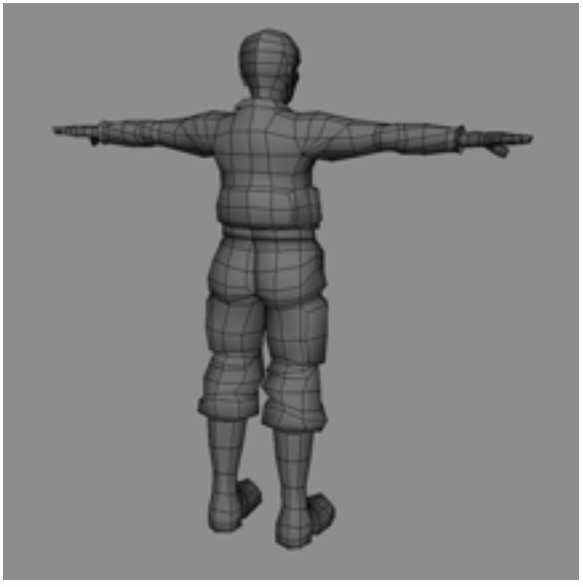
I add an extra row of vertices around the eyes and extrude the newly created faces to create the sockets of the eye.
I modify the geometry around the center providing means to extrude the nose.
In a similar fashion, I extrude the faces around the mouth to form the lips.
I add, delete and flip edges in the face area to follow the flow of the facial and neck muscles.



I add definition to the jaw line and extrude the ears.
I continue to refine the mesh until I'm close to the reference image. I add eyeballs and refine the shape of the lids around them. I add very little geometry to create the inside of the mouth. This will prevent a clear view inside the head when the mouth is opened.
The limited amount of joints in the budget of this project will only allow for minimal facial animation.
But since the face is the most distinct area of the character, I will try to add animation controls for the jaw and the eyelids. I can neglect most of the muscles I would need for lip sync and facial expressions.

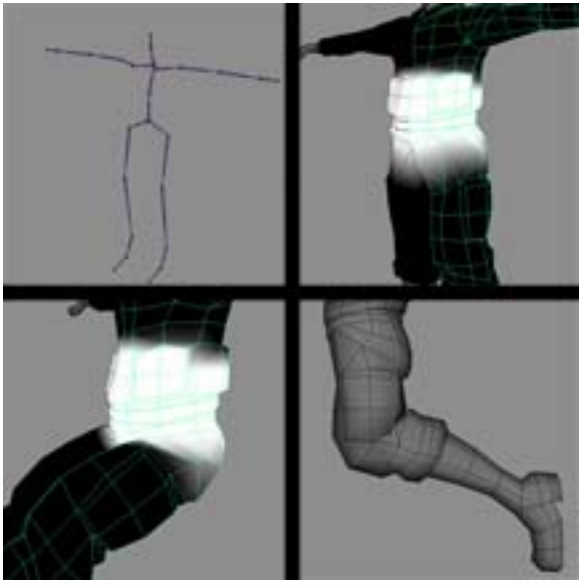


The second part of the modeling is complete.
I added more detail to the mesh, fixed some problem areas early on and generally got a lot closer to the concept drawing.
In the next steps I will create some basic rigs to test how the mesh deforms and fix the problems accordingly.
Also, I will combine both halves of the mesh and start adding asymmetric detail, like the holster and the chest pocket. The next time I'm working on the face, I will also create the different faces and hairstyles.
At this point I can take a look at areas that don't hold up against the concept.
For example, the chest appears a little flat and will need refinement in the following stages.

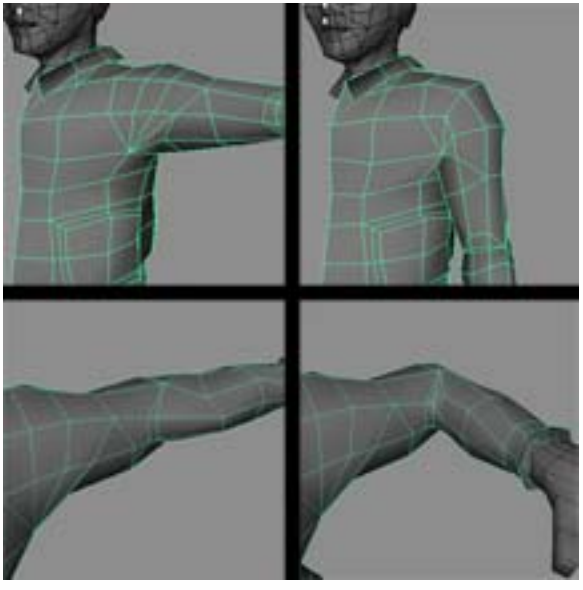


Wrapping up the second round of modeling, let's take another look at the budget:

At this stage, I've used 2308 Triangles, and worked another two hours on the character.
In total, the project up to this point took just about three hours.

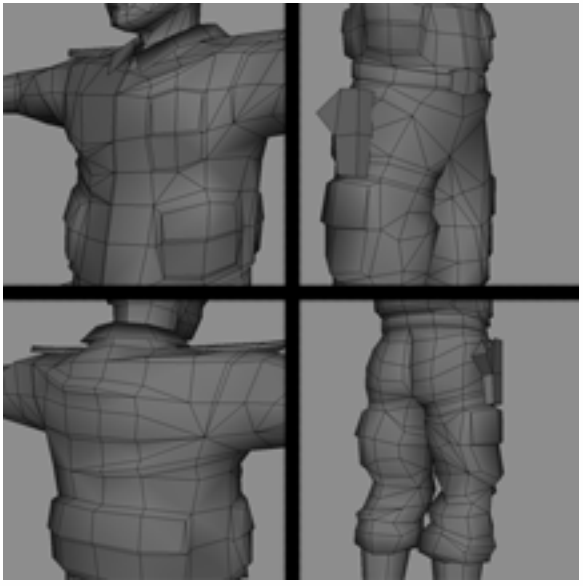


I will now start creating a basic character rig to test the deformation on various areas of the mesh.
Since the proportions of the character are already correct, I can use this skeleton as a starting point for the final rig later on.
I will be working on one side of the character only, fixing the areas of the lower body.
For most game environments, it is highly unlikely to achieve a perfect deformation of the mesh and some intersections are bound to occur.
I try to limit make these problem areas to appear as wrinkles in the clothing.



Another problematic area is at the shoulders. The clavicle I placed in the skeleton will fix a lot of pinching when the arm is raised above the head, but there is still some fixing to do.
As with the legs, I add, delete and move geometry until I achieve a satisfactory bend in the shoulders and elbows. In this case, I ended up with some intersections only in the arm pit area.

With these problems fixed, the budget stands at 2692 Triangles.



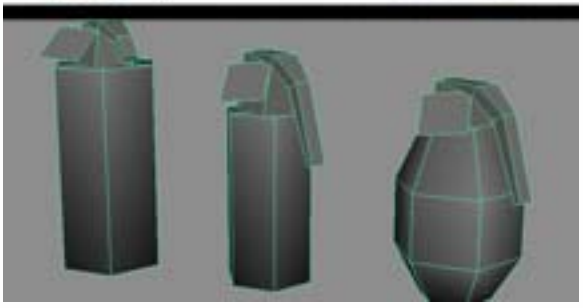
Having addressed most of the problematic areas, it is now time to combine both sides of the mesh and add asymmetric detail to the character.
In this part, I will work my way down from the neck towards the boots.
I will be creating distinct wrinkles in the clothing and assuring continuity where the former two halves meet.
Also, I will be adding separate objects for the belt buckle, the shoulder stripes and the gun holster on the right side of the characters body.
As I add the new detail, I continue to test all the changes on my default skeleton.

Next I add the individual details to the different heads I'm going to use in this project.
Since I'm using the same skeleton for all the heads, I have to make sure that the eyes and the general area of the jaw stays in the same position. This will cause some minor deviations from the source images, but will assure proper deformation.
The basic head mesh created in the previous steps provides a fast working base for these modifications, clocking in at just about an hour for all three heads.

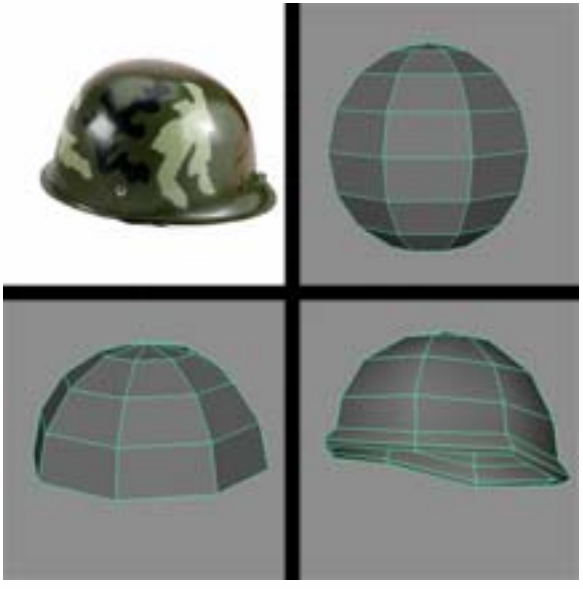
(NOTE: The textures seen are merely stand-ins. Neither the UVs nor the image map are anywhere near finished. Left and center image taken from www.3d.sk)

Another important task that was left untouched is creating the weapons. In these next steps I will model a M-9 Semiautomatic Pistol, a M-4 Carbine, a standard helmet and various grenades.
The M-9 measures 21.69 cm in length, 3.81 cm in width and 14.05 in height. I will use the size chart created at the beginning of the project as a guide.
I start with a simple cube and add small detail like the trigger as separate objects.
Fairly recognizable, the M-9 stands at 188 Triangles.

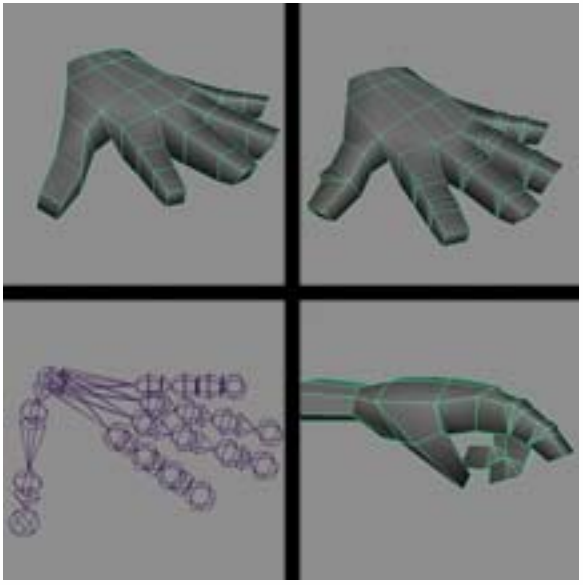
Next up is the M-9 Carbine.
Starting with a poly cube, I try to follow the most dominant features of the weapon.
As with the M-9 I use the size chart and a reference image to achieve the correct proportions.
At the finished stage, the weapon stands at 433 Triangles.



For the thrown weapons, I will create one egg shaped model for the frag grenade and cylindrical meshes for both the smoke and the flash bang grenade.
Combined, the grenades make up 204 Triangles.



The helmet is essentially a modified sphere. I delete the faces at the bottom and modify the geometry to resemble the source image.
Finished, the helmet is composed of 88 Triangles.
For all the weapons, I took just about another hour of the modeling budget.



With the weapons in place, I can now start finalizing the hands. I add extra edges around the joints and adjust the resulting geometry.
Next I put the fingers into their final pose.
To do this, I create a simple skeleton and rotate the joints into a relatively neutral pose. I want to be able to use this one hand to mimic actions ranging from holding a gun over saluting to pointing fingers.
I use the newly created weapons to determine how much I need to bend each individual finger.

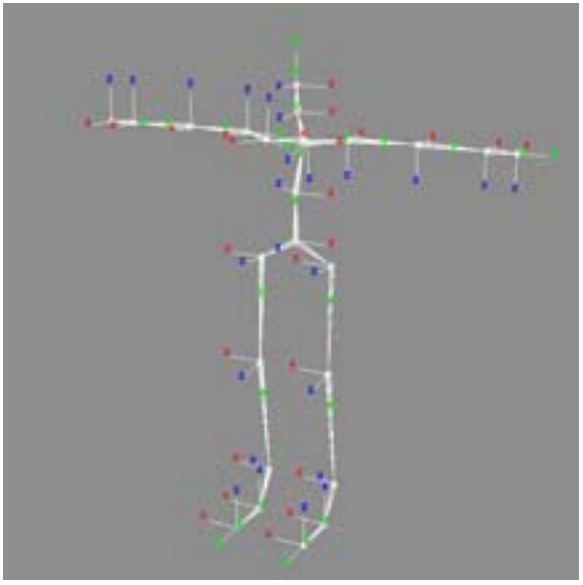
The lion share of modeling the character is completed. The only modifications the model will undergo from this point forward are merely fixes for problems that occur during the process of rigging.

Another look at the budget:

At the end of the modeling phase, I used 3948 Triangles for the character and the accessories, taking about 9 hours of the clock.

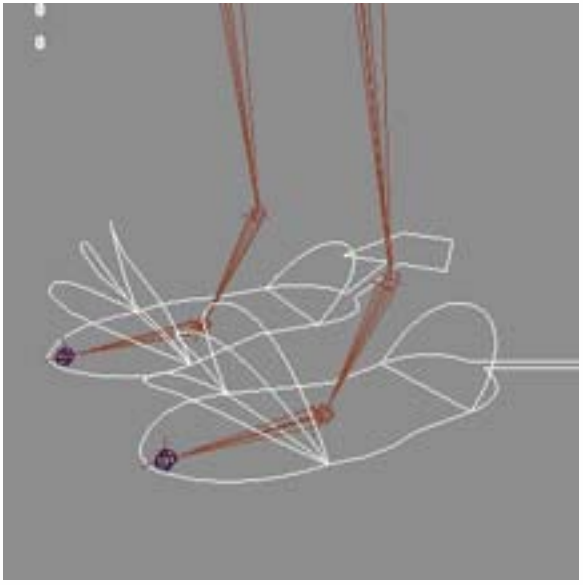
PART III

CHARACTER RIGGING

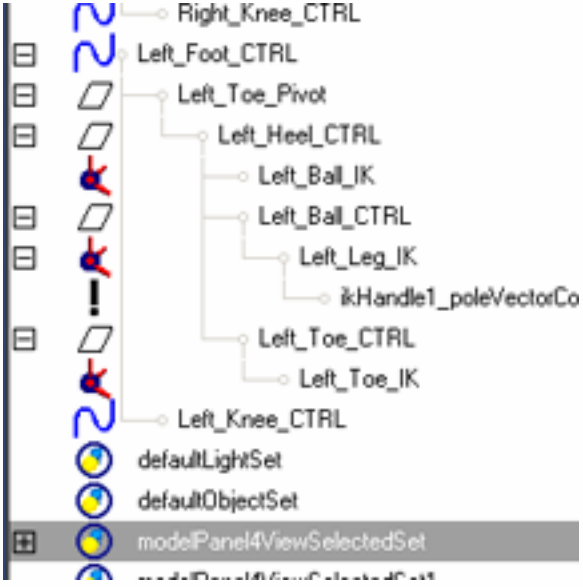


It's time to tackle the next big task of this project: The character rig.
As I mentioned before, I prefer to skin the character before I create the textures. This is done to avoid having to redo large portions of the texture maps due to excessive fixing of the mesh that might be necessary during the skinning process.
I use the rough skeleton created during the deformation tests in previous steps as a starting point. I trace the old skeleton and orient the joints locally.
In this case, the character consists of 26 joints (four for each leg, four for each arm, four for the back and neck and six for the head; I used that many joints in the head to allow jaw, eye and eyelid movement). This leaves four (plus/minus two)joints for the accessories.

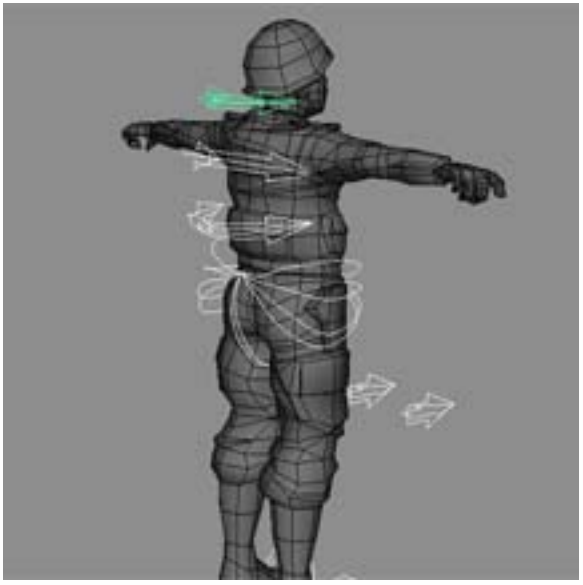
In general, I prefer the use of control objects over the direct manipulation of joints and IK handles.
Over time I assembled a small library of splines that I will be using during the creation of the character rig.



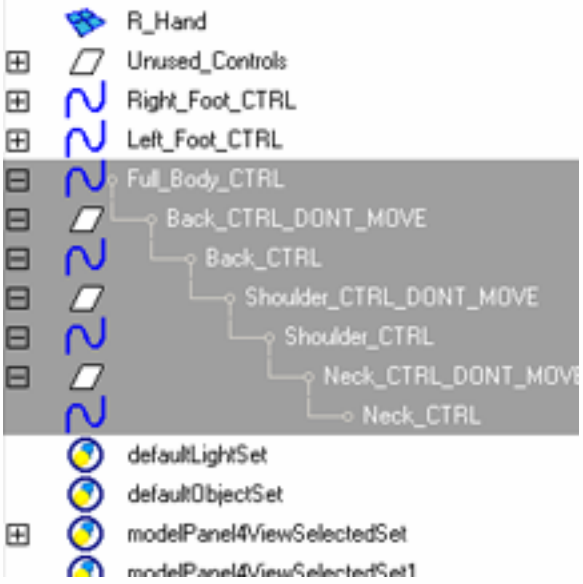
I start with the legs. For the rig, I create one IK skeleton for each leg, including a toe joint, and use it to drive the final skinning skeleton. Doing this will allow for more animation control of the foot without the need for a toe joint in the final skeleton.
I add an IK from the hip to the ankle, one from the ankle to the ball and one from the ball to the toe joint.
I cerate the pole vector for each leg, using one of my control objects. I position the foot control object around the boot and snap the pivot point to the ankle joint.
I duplicate the foot control four times and delete their shapes, leaving only an invisible group.



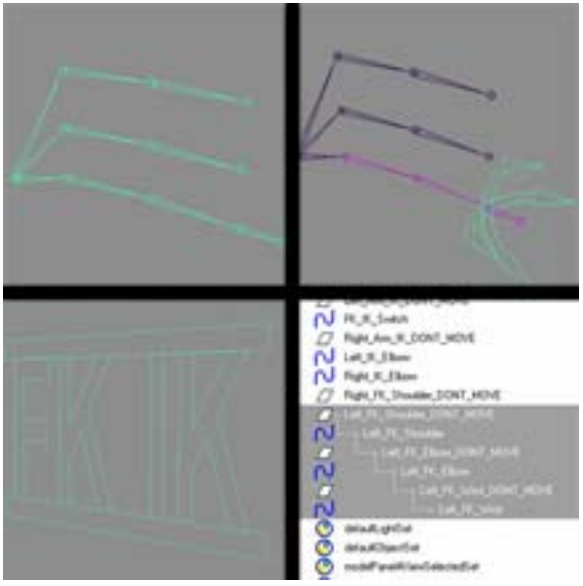
I rename one these groups to **Heel CTRL** and move its pivot around where the heel is located.
I rename the next group to **Ball CTRL** and move the pivot to the ball joint. Another one I rename to **Toe Pivot** and snap its pivot to the toe joint. The last group is renamed to **Toe CTRL** and its pivot is also moved onto the ball joint. As with all controls I will be using, I make sure that their values are set to zero. I arrange the groups and IK handles as seen in the image on the left. The visible foot control already contains some custom attributes (e.g. heel roll, ball roll, etc.) that I will use to drive the rotation of the groups created earlier. This will gather everything necessary to animate a leg into one single shape. As a last step on the legs, I connect the rotation of the IK skeleton to drive the rotation of the corresponding skin skeleton.



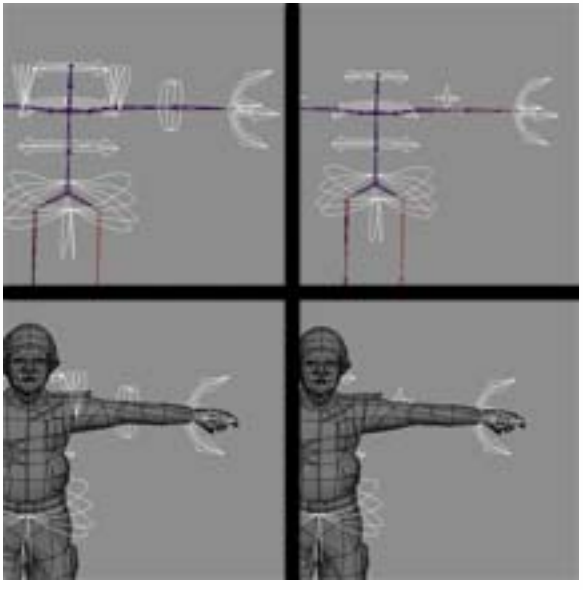
The setup for the back will be pretty straight forward. There are only three joints in the back, so I will drive each joint with a control shape. For the root, I simply position the Pelvis control at the hip area, snap the pivot to the root joint and connect the rotation and translation values. To achieve a clean setup for the rest of the back, I first position the three remaining controls exactly at the world origin and freeze all transformations. Next I parent the back CTRL to the back joint, the shoulder CTRL to the shoulder joint and the neck CTRL to the neck joint. Having done this, I set each controls translation and rotation values back to zero. Doing this (in MAYA) will position the control objects (and more importantly their local rotation axis) at and aligned with their counterpart joints.



As a next step, I unparent the objects from their joints. The objects will still be aligned with the joints, but now have values other than zero in their translation and rotation channels. Simply freezing transformations at this point would undo the alignment to the joint. To work around this issue, I duplicate each control object and delete their shape nodes. I parent the visible control to the newly created empty group. This little trick transfers the values of the control object to its parent. This might sound complicated, but it will save a lot of trouble later on if all the animation controls can simply be set to zero to revert to the bind pose. Next, I connect the rotation of the control objects to drive the joints of the back. To finish up the back, I parent all the controls as seen on the left side.



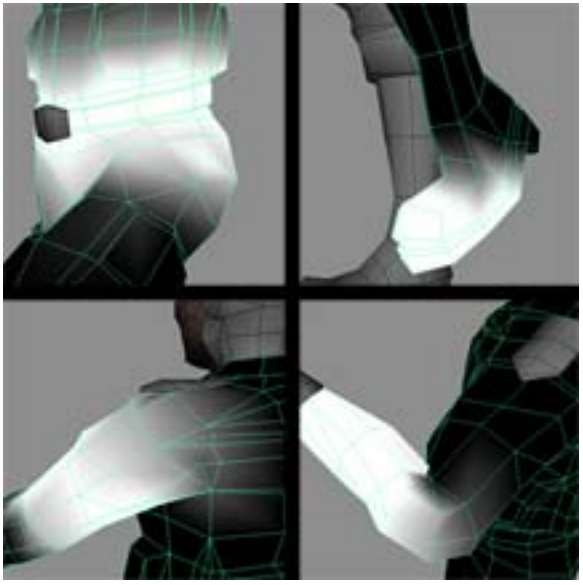
The arms will be slightly more complicated than the legs. To provide better animation controls, I will be creating three skeletal systems for each arm. (NOTE: Only one of these skeletons will burden the budget. The other two are merely for animation purposes.)
For a better overview, I spread out the three arms as seen on the left side. The lower (longer) skeleton is used for IK, the middle skeleton will be used for skinning and the top shows the FK skeleton. Even though IK/FK blends are available within the IK solvers in MAYA, I find this approach to cause fewer problems later on.
I create an IK chain from the shoulder to the wrist and one from the wrist to the hand of the IK skeleton. I rename them IK_Arm and IK_Hand respectively. I use the same approach as with the legs to align my wrist control object with the IK skeleton.



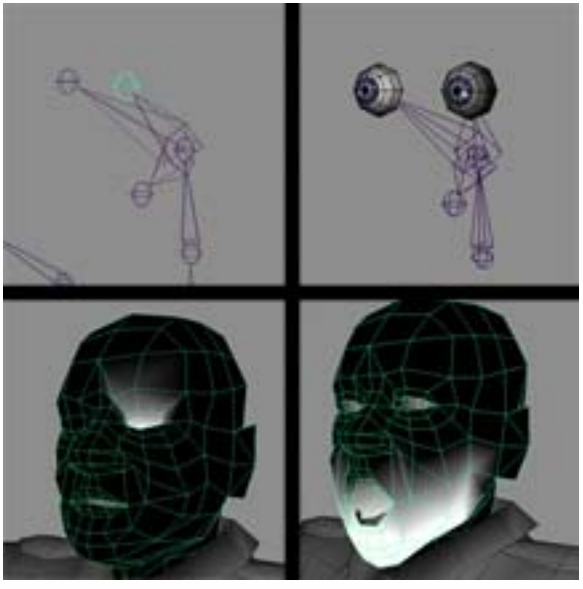
I snap my elbow control to the elbow joint, move it some units back and use it as a pole vector. Since only the translation values of the control object are of any importance, I can simply freeze its transformation. I parent both IK chains to the control object. To drive the skinning skeleton, I constrain the joints to both the FK and the IK arms. I can then animate the weight of these constraints to blend between them.
I created a FK/IK control object that holds these blend attributes, as well as channels to adjust the shoulders horizontally and vertically. To avoid confusion, I will also set keys for the visibility of the control objects so that at any given time only FK or only IK controls are onscreen.
As a last step, I add driven keys to drive the collar joints.

I parent and arrange all the controls under the Move All control (the big arrow at the feet). Above the character are the control objects for the FK/IK blend, an extra control that will drive the constrains of the different weapons, a face control for the eye and jaw movements, and the UI button, which will be a central part of the overall look of the character rig.

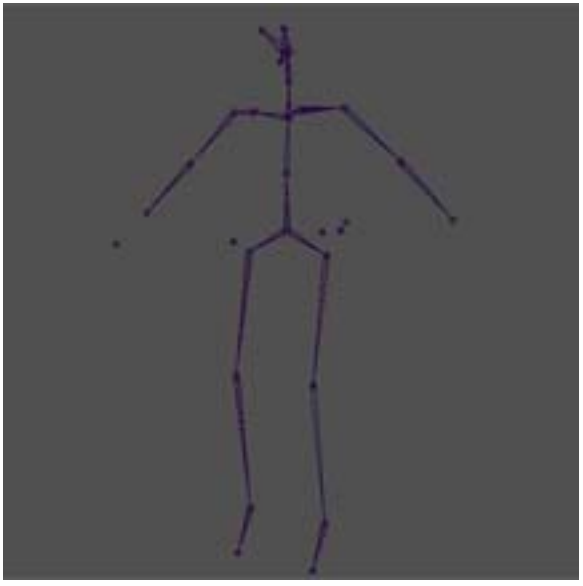
One feature that I like to include in my characters is a user interface. Over time I assembled a small library of MEL scripts that I either wrote or re-wrote. The scripts are set up in a way that allows me to quickly change the names of their attributes and variables. This way I can save a lot of time by recycling almost the entire script. Included will be the FK IK blend attributes, IK FK Match controls, switches for the weapons, a selection window to choose the color of the camouflage uniform, a selection window for the different heads, a large "Picker", that allows for an easy selection of any relevant animation control and a section for creating poses and animation clips.
(NOTE: I will not get into any further detail on the UI, but for the interested reader I added the entire script at the end of this document.)



For the process of skinning, I first create a couple of selection sets. I select all the joints that will deform the torso and create a quick select set. I do the same for all the other parts of the character. This will speed up the workflow if I have to detach and reattach any part of the model. Using these sets, I bind each piece of geometry to the skeleton. I paint the skin weights and fix any awkward deformations that still occur while working my way up from the boots to the head.

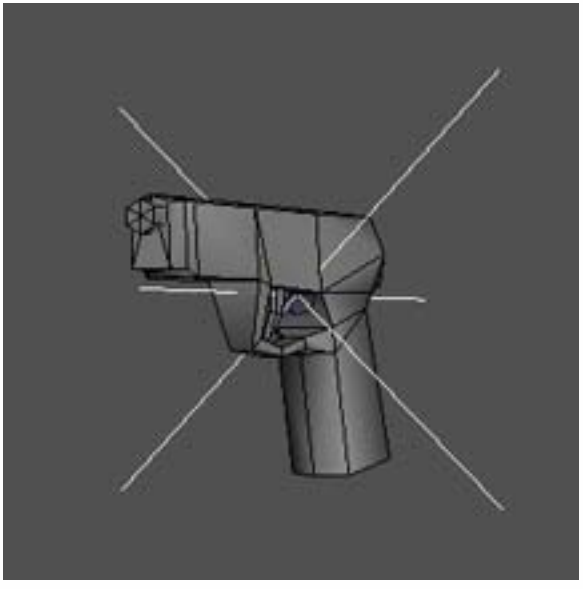


Before I bind the different heads, I have to make sure that the eyes and lips in all three meshes are in the same position. Next I bind all three heads to the skeleton. Working on the first mesh, I weight the jaw to allow a natural opening of the mouth. The eye joints seen on the left are actually two joints exactly on top of each other and both parented to the head joint. One of these joints on each side will animate the eyes the other one will drive the upper eyelids and a little bit of the eyebrows. This way the character can show a small range of emotion with his eyes.
After finalizing the weights on I can simply transfer all the skinning information over to the other head meshes.



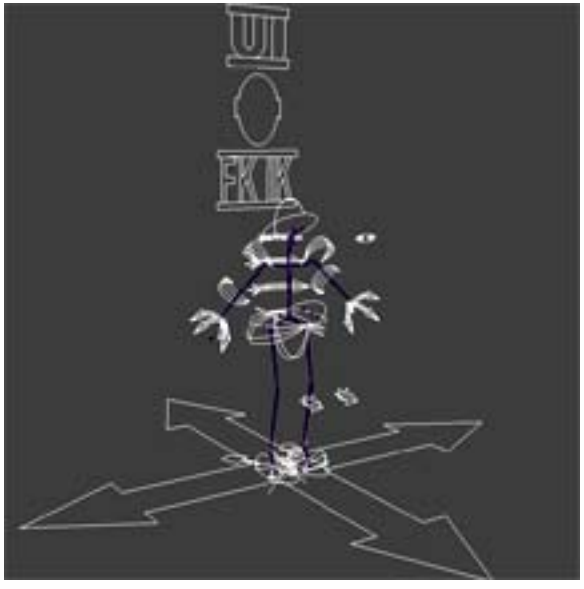
So far, the skeleton contains 26 joints. Six more joints will be added for the accessories.
This is just at the maximum limit set for this project, but the extra joints I decided to add for the facial setup will provide a much more lifelike character.

To finish the character rig I place one joint at every weapon and item (handgun, rifle, grenades and helmet) and bind the entire mesh to the corresponding joint.
I place each joint in its own group. This will provide me with a clean node with zero transformation values that I can use to place each item into a variety of positions.
To show the process of creating various target points for each item, I will take the handgun as an example.



The pistol will be constrained to either the holster, the right hand or to a free target which can be moved without restrictions. First I pose the handgun at the world origin. I create two empty groups and name them Holster_Target and Right_Hand_Target respectively. I cerate one curve to use for the free transform (in this case a cross) and name it M9_CTRL. I parent constrain the weapon group to all three of the targets. Setting the weight to only the hand target, I position the Right_Hand_Target group so that the weapon is polaced naturally in the hand. To lock the Right_Hand_Target group into place I simply parent constrain it to the wrist joint. I repeat this process for the other targets and then for the other items. By animating the constrain weights I can snap the items to all the different targets.

At this point, all the controls and attributes are already in the final stage. As a last step, I create character sets to store all the relevant date. I create one master set called ES_CHAR_SET and under it two sub-character sets named ES_Body_Set and ES_Face_Set. I lock all attributes not relevant for animation (e.g. scale attributes in most cases, visibility, and translation values on controls that are only allowed to rotate).
I place the remainder of the attributes under its respective character set (all but the eyes and face attributes will be placed under the body character set).
Having done this, the character is ready top be animated. Even though some minor changes might still occur on the mesh during the creation of the textures, any animation created can easily be transferred over to the revised model using the clips and poses.

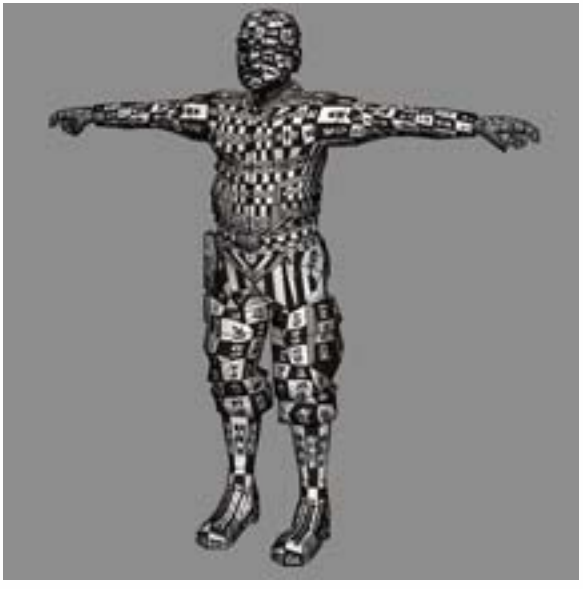


Time to look at the budget again.
Creating the skeleton took about four hours, rewriting the script for the UI and adding new features took about another two hours and painting the weights on the meshes took about another two hours.
This makes up eight hours spend on the rig so far.
At this point I estimate about another two hours to fine tune the rig and fix any problems that might occur.

PART IV

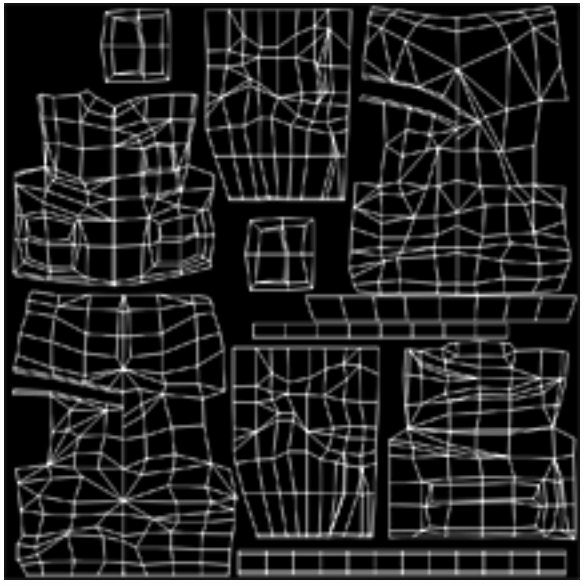
TEXTURING

Having dealt with two major aspects in creating this character, I am now ready to create the textures.
First up, I will have to lay out the UVs. As mentioned before, there are three textures that make up one character. Split up, these are one 256x256 texture for the camouflage part of the body, one 128x128 texture for each head (three, all together) and one 128x128 texture for everything not affected by the vertex coloring (this includes the boots, the belt buckle and the gun holster). Each additional item, like the weapons, will get their own texture as well, its resolution depending on the relative size to the character.
I'm using a default texture to lay out the UVs with as little stretching as possible.



Applied as is, the stretching and warping of the texture is more than apparent.
As a next step, I create the following quick selection sets that will aid me in projecting the textures:

One set for the front and one for the back of the upper body, one for each arm, one for the belt, one each for the front and the back of the pants. The rest of the geometry is still separated from the body mesh, so I don't need to create new sets for those objects.



I use a variety of different texture mapping techniques to create a clean UV layout. Using the checker texture seen above as a reference, I manipulate the UVs to minimize and/or hide any stretching that occurs. Although I will spend a couple of hours laying out the texture coordinates, I will not go into any further detail on this subject matter.
Once I'm satisfied with the ways the default texture behaves, I arrange the pieces to fit inside the grid.
Every part of the clothing will be using one texture, the shoes, hands, belt and holster will be using another one together, each head will receive it's own texture and last but not least, each item and weapon will have it's own as well. You can see the UVs of the torso on the left.

As for the dimension of the textures, I usually use twice the final size to create the textures and scale them down in the end. Going any smaller will only make it harder to show delicate detail while going any bigger will cause too much blurring of the image once it is scaled down.

I will start creating the texture map for the torso. On the left you can see a camouflage pattern that I will use as a reference. Since this image is in an unacceptable resolution for this project, I re-create the pattern and make it tile able using Photoshop's offset filter and clone stamp tool.
Also, while keeping the contrast equal to the reference file, I use only grey scale values.

First up, I place the newly created camouflage pattern on the bottom layer of the image file. To make the uniform look real, I cannot use the pattern as is. I want to create the illusion of different patches of cloth stitched together. I create patches for each of the pockets on the uniform and make sure the pattern shows a noticeable difference to the layer below. I do the same for each area of the clothing that will show a seam on the surface. From this point on, I will often view the texture on the actual model to check my progress.

Next I create place the seams. Using a new layer, I create stitches to outline the pockets and other areas where the clothing is stitched together.
I pay close attention to the size of the seams. Keeping in mind that the texture will be scaled down I make sure that every piece of detail I add measures at least two pixels, so that the detail will not be lost later on.

To add finer detail, I will use a combination of photographs and hand painted layers. First, I spend some time taking pictures of white cloth with a variety of different detail. The left hand side shows a picture of some wrinkles. Using white for the cloth will allow me to multiply the camouflage pattern with this image to create shadows and wrinkles that look natural.

I lighten and darken areas and use two more layers to hand paint shadows and highlights respectively. Although I'm adding these details I make sure that neither highlights nor shadows are too noticeable. I want them to only suggest wrinkles, but not a distinct light source.
Switching back and forth between Maya and Photoshop, I adjust the wrinkles until I am satisfied with their look on the character model.

Next, I create a different pattern for the belt and weapon holster strap.
Once I'm happy with the look on the model, I add one more layer with grain to overlay the texture to simulate the uneven surface of the cloth.

This texture, as is, will be used to show the snow camouflage uniform as well as serve as a base for the vertex coloring.

For all the other camouflage variations, I simply color each grey scale value of the base layer with a new color, derived from various reference images.
The left side shows the finished jungle camouflage uniform.
I repeat this step for the other two variations.

The torso textures from start to finish took about five hours.

The next texture I'm going to create covers all the other body parts other than the faces.
Again, I use a combination of photographic reference and hand painted layers to create the color map seen on the left.

This texture took roughly two hours to create.

Now it is time to create the textures for the different heads. Before I can use the reference images as base for the textures, I have to spend some time to tone done all the obvious highlights and shadows on the photographs. I want to keep some differences in the skin tone, but obvious points like on the forehead have to be adjusted before they can be used.
Without going into any further detail, the left side shows the steps from the reference file to the UV layout over to the finished texture all the way to the look on the model. The hair has been changed to show a more military like haircut.
The textures for each head took about an hour each to create.

Using a variety of reference files, I next create the textures for all the other accessories.
The left side shows the finished texture for the M-4 rifle.

The textures for each of the guns took about an hour each, the helmet and the grenades clocked in at about half an hour each.

Wrapping up the texturing, I spend about 14 hours on this part of the character creation.

U to this point I spent a total of 31 hours on the character.

(NOTE: Although the textures are at their final size, I have not yet optimized the color depth to fit any specifications. Also, I keep a file for each texture containing all the layers, so I have the ability to later go back and modify any area that does not work well with if put in a game environment. Since this is a fictitious project, I will have to skip these steps)

PART V

WRAPPING UP

At this point, I'm almost 19 hours under my projected timeline.
With this much time to spare, I decided to create two additional heads, one resembling an Asian character and one with a full face mask
I spend about an hour on each head. The mesh with the mask has a slightly lower poly count since I can get rid of any features in and around the mouth.
Once I'm at the stage seen on the left side, I copy the skin weights from one of the original heads over to the new ones.

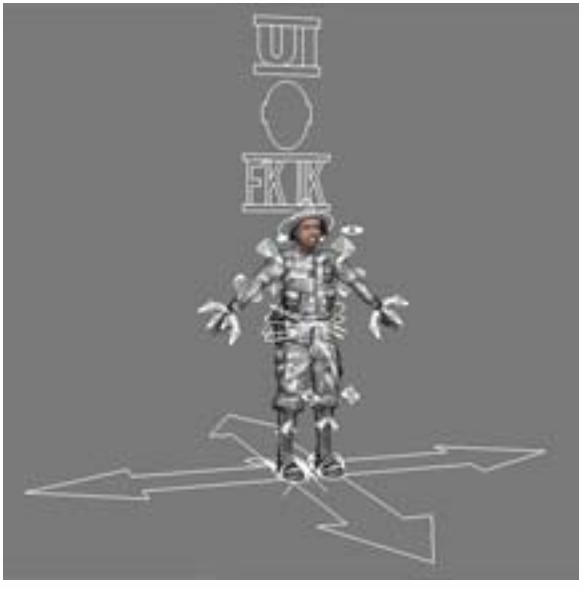
(NOTE: Since I am also the art director on this ficticious project, I got green light on this idea right away)

I add the last features to the user interface. This includes a section that allows choosing between all the available heads with the simple push of a button (1).
Also included will be controls to change the color of both the uniform and the eyes (2a and 2b).
The last new feature is a tab that houses all the weapons and items constrains (3).

This step took about another two hours.

With still a lot of leeway, I spend about three hours testing the rig. I check for problem areas, fine tune some of the skin weights and make sure nothing breaks.
I create a couple of poses to see how the textures behave in situations that might bee seen in the game.

Now, I'm just about at the point where I can't believe that I'm done.
But I am.



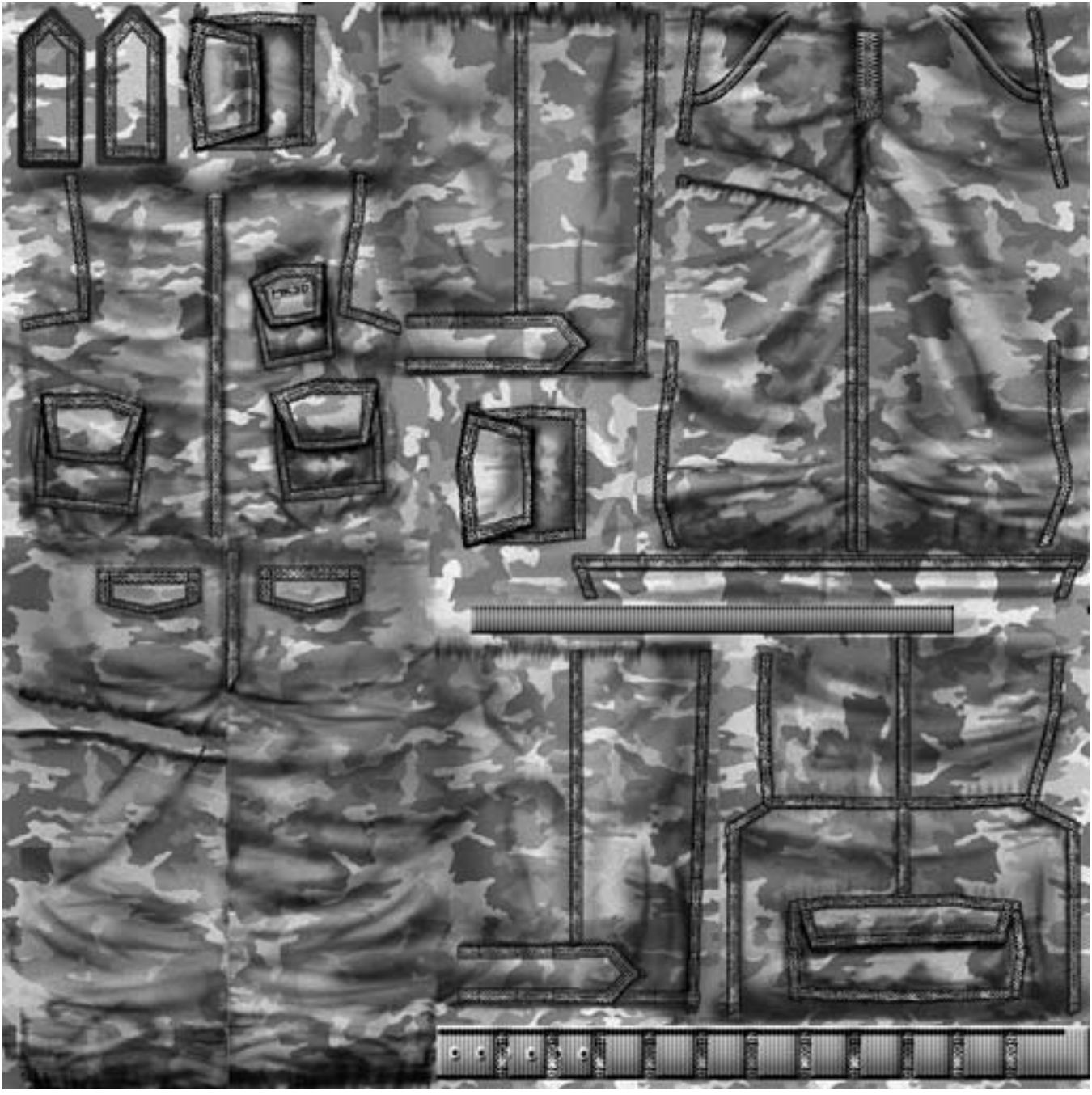
After reading (or browsing through) this document, you have witnessed the creation of the Enemy Soldier character from start to finish.
The character is just below the maximum polygon count of 4000 set as a limit at the beginning of the project. The joint count is 32, including six non-skin joints for all the accessories.
And looking at the time line for this project, I'm still about 12 hours below the projected 50 hours for this scenario.
That's it.

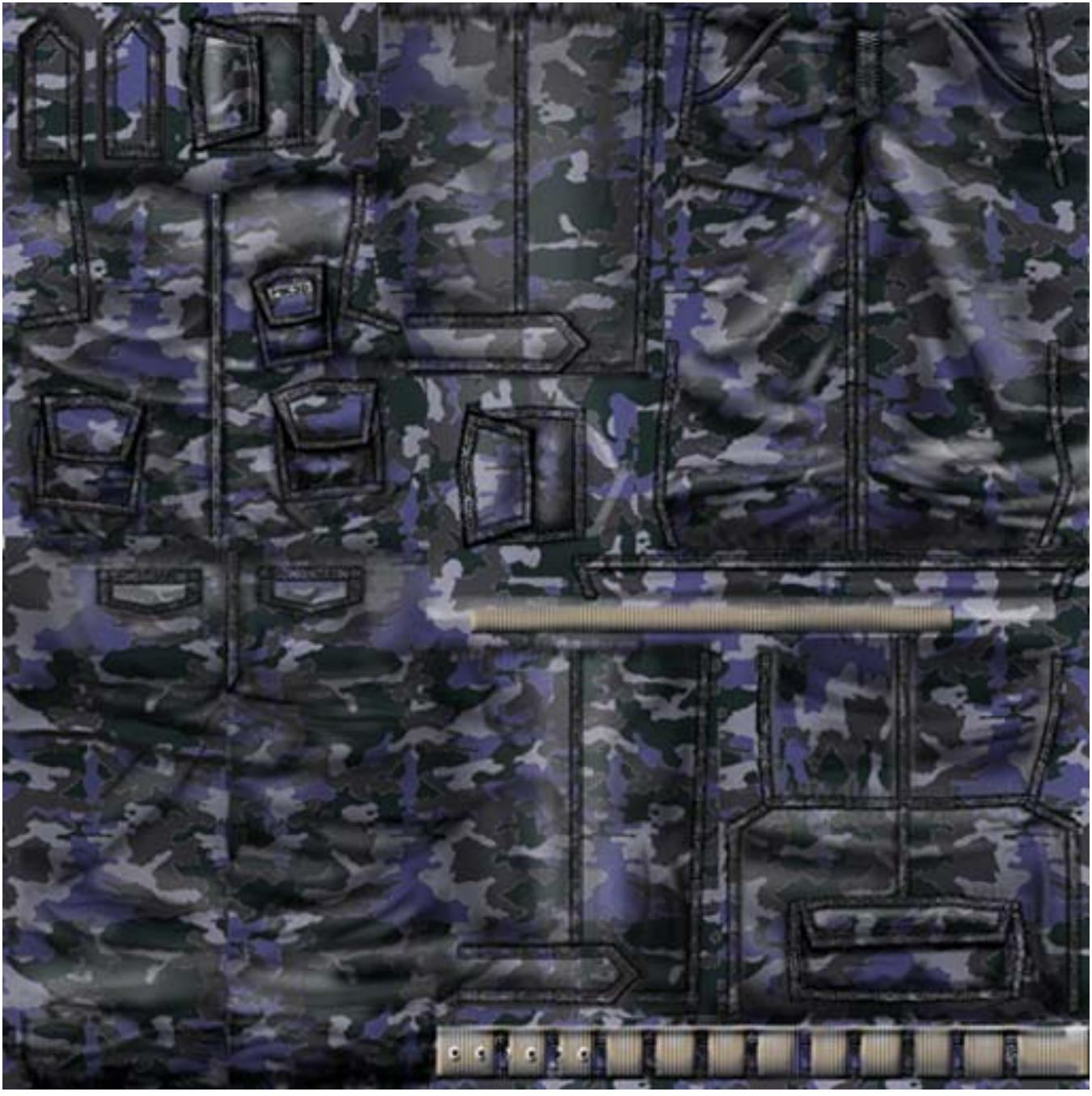
I need a drink.

Mike

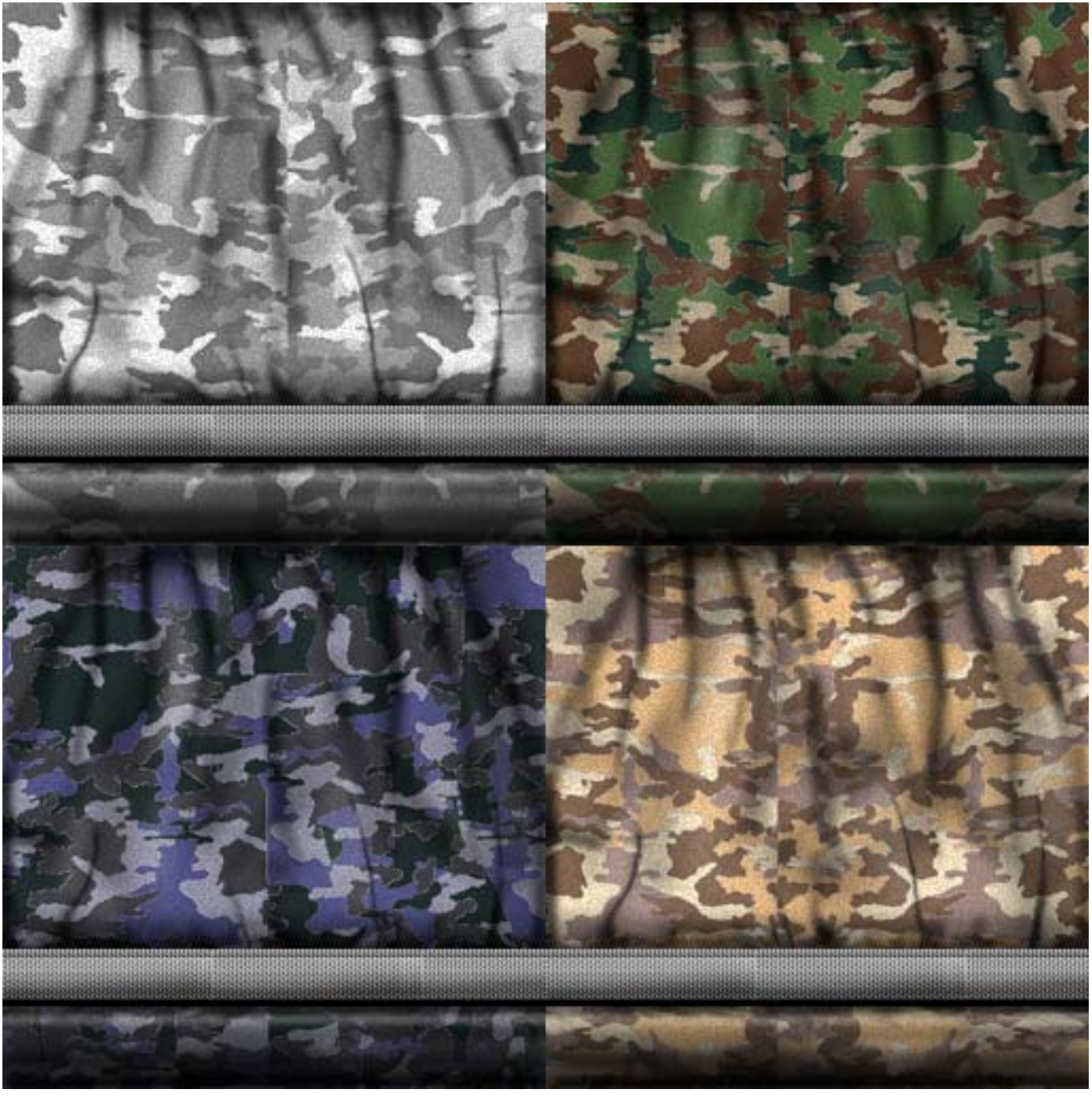

APPENDIX A

TEXTURES




MK-3D
Flash & Bang
Type 3 Grenade
DO NOT EAT!



APPENDIX B

USER INTERFACE SCRIPT

The following pages contain the entire script used in the creation of Enemy Soldier UI v1.0.
Each part of the script is accompanied by a headline briefly explaining its function.
The UI is written in Maya's MEL scripting language.
I will not go into any further detail, because explaining all the different procedures would cover a book by itself.
Feel free to write me with any questions you might have about the following script.

```
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
//                                                    //
//             This Script adds a UI Window for       //
//             the enemy soldier's Controls           //
//                                                    //
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
//                                                    //
//      Model, texture and Rig created April 2005     //
//                     by M.S.R Kiessling             //
//                        www.mk3d.com                //
//                        mike@mk3d.com               //
//                                                    //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
//                                                    //
//      This rig is free to use for non-commercial    //
//      purposes. All I ask is to see any final       //
//      Animations created with this rig              //
//      Have fun!!!                                   //
//                                                    //
////////////////////////////////////////////////////////


//      Check if the file pointing to the Folder      //
//      Containing all the neccessary files exists    //
//                                                    //

global string $ESCharName;

$ESExists = ( `internalVar -usd ` + "ESLoc.txt");
$ESLoc = `filetest -r $ESExists`;

 if ($ESLoc == 0)
      {

        //      Creates a popuop to specify location  //
```

```
        if (`window -q -ex GetESPath`==true) {deleteUI GetESPath;}
        windowPref -ra;
        window -widthHeight 260 200 -t "ES Path Finder" -s off GetESPath;
            columnLayout;
                text -l "" -h 6;
                text  -l (" Navigate to the <ESUI> Folder and select ES.Loc") ;

                text -l "" -h 6;
                textField -w 250 -ip 6 -en off  -tx "   <<< Please Specify Location Of ES.Loc >>>"
ESPath;
                text -l "" -h 6;
                button -w 250 -h 50 -bgc 0.6 0.6 .8 -l "FILE BROWSER" -c "ESBrowse";
                text -l "" -h 12;
                button -w 250 -h 50 -bgc 0.6 0.6 .8 -l "INITIALIZE" -c "ESWrite";

        showWindow GetESPath;}



   else
        ESUI;

//                                                              //
//      Global procedure ES Browse                              //
//                                                              //

global proc ESBrowse()
        {
        $ESPath = `fileDialog -dm "ES.loc"`;
        textField -e -tx $ESPath ESPath;
         }

//                                                              //
//      Global procedure ES Write                               //
//                                                              //

global proc ESWrite()
        {

        string $ESPath;
        string $ESStorage[];
        string $ESNamePath = `textField -q -tx  ESPath`;
        $ESToken = `tokenize $ESNamePath "/" $ESStorage`;
        $ESPathMel = ( `internalVar -usd `+ "ESLoc.txt");
        $ESLess = ($ESToken - 1);
        for ($K = 0; $K<=$ESLess; $K++)
        {
                if ($K ==0)
                        {
                        $File =`fopen $ESPathMel "w"`;
                        fprint $File ($ESStorage[0] + "/");
```

```
                        fclose $File;
                        }
                else if ($K >0 && $K < $ESLess)
                        {
                        $File =`fopen $ESPathMel "a"`;
                        fprint $File ($ESStorage[$K] + "/");
                        fclose $File;
                        }
                }

                ESUI;
                deleteUI GetESPath;

        }

//              Call A SCript job to open the UI              //
//              whenever the UI Control is selected   //

scriptJob  -event "SelectionChanged" ESUIOpen;

global proc ESUIOpen()
        {
        string $ESSelected[] = `ls -sl`;
        if ($ESSelected[0] == "ES_UI_Button")
                {
                select -cl  ;
                print "hello";
                ESUI;
                }
        }

//      End of procedure                                        //

//      Creation of the actual UI                               //

global proc ESUI()
        {
        $ESFile = (`internalVar -usd ` + "ESLoc.txt");
        $GKL = (`fopen $ESFile "r" `);
        global string $ESLocation;
        $ESLocation = `fgetline $GKL`;


        global string $ESUIWin = "ESUIWin";
        if (`window -exists $ESUIWin`)
        deleteUI $ESUIWin;
        window  -tlb on -mb true -rtf true
                        -w 400 -h 500
                        -t "ES UI V.1.0"
                        $ESUIWin;
```

```
        menu    -label "File" -tearOff false;
        menuItem        -label "Exit"
                        -ann "Closes the UI"
                        -command ("deleteUI "+$ESUIWin);
        menu    -label "?" -tearOff false;

        menuItem        -label "Help"
                        -ann "Read this first"
                        -c ESHelp;
        menuItem        -label "About"
                        -ann "General Information"
                        -command ESUIAbout;
        menuItem        -label "More"
                        -ann "Check out my website..."
                        -c MySite;
        columnLayout;



setParent..;
setParent..;
 string $ESForm = `formLayout -w 390 -h 270`;
    string $ESTabs = `tabLayout  -w 380 -h 270`;
    formLayout -edit
       -attachForm $ESTabs "top"    0
              -attachForm $ESTabs "left"   0
              -attachForm $ESTabs "bottom" 0
              -attachForm $ESTabs "right"  0
              $ESForm;

              string $child1 = `paneLayout -w 390 -h 270`;
                             string $browser;
                      $browser = `webBrowser -width 800 -height 600 `;
                      webBrowser -edit -url ("file:///"+$ESLocation + "/html/Select.htm") $browser;
                      setParent..;

              string $child2 = `paneLayout -ps 1 70 90 -ps 2 30 90  -w 390 -h 270 -cn quad`;
                      string $ESFaceCam = `modelPanel -mbv false`;
                      modelEditor -edit      -cam "ES_ESUI_FaceCam"
                                             -da "smoothShaded" -displayTextures on
                                             -gr 0 -alo 0  -pm 1 $ESFaceCam;
                      setParent..;
                      frameLayout "The Channel Box";
                      channelBox -fw 25 ;
                      setParent ..;

              columnLayout;
                      button  -h 15 -w 105 -bgc 1 .8 .6
                                     -ann "Select the Face control"
                                     -l "Select Face"
```

```
                        -c "select -r ES_Face_CTRL";
        setParent..;

rowColumnLayout -nc 3 -cw 1 90 -cw 2 90 -cw 3 90 -h 15;
        button  -h 12 -w 88 -bgc .8 .8 1
                        -ann "Reset the Face Cam to its default position"
                        -l "Reset Face Cam" -c ESResetFaceCam;
        button  -h 12 -w 88 -bgc .8 .8 1
                        -ann "Store the Current face as a pose"
                        -l "Create Pose"
                        -c "CreateFacePose";
        button  -h 12 -w 88 -bgc .8 .8 1
                        -ann "Store the Current face animation as a clip"
                        -l "Create Clip"
                        -c CreateFaceClip;

        setParent..;

setParent ..;
string $child3 = `paneLayout -w 390 -h 270 -ps 1 78 100 -ps 2 22 100 -cn vertical2`;
        string $ESPoseCam =`modelPanel -mbv false`;
        modelEditor     -cam "ES_ESUI_PoseCam"
                        -edit -da "smoothShaded"
                        -displayTextures on -gr 0
                        -alo 0  -pm 1
                        $ESPoseCam;
        setParent ..;
        columnLayout;

        button  -l "Select None" -h 50 -w 75
                        -bgc 1 .8 .6 -c ESNone
                        -ann "Clears the Character Set selection";
        button  -l "Select Body" -h 50 -w 75
                        -bgc 1 .8 .6 -c ESBody
                        -ann "Selects the whole character, except the Face";

        separator -h 10 -w 75 -style "singleDash";
        button  -l "Create Pose" -h 50 -w 75
                        -bgc .8 .8 1 -c ESBodyPose
                        -ann "Creates a new Pose";
        button  -l "Create Clip" -h 50 -w 75
                        -bgc .8 .8 1 -c CreateClipOptions
                        -ann "Creates a new Clip";
        button  -l "Reset Cam" -h 50 -w 75
                        -bgc .8 .8 1 -c ESResetPoseCam
                        -ann "Resets the Camera";

        setParent..;
        setParent..;
```

```
                tabLayout -edit
                        -tabLabel $child1 "Selection Handles"
                        -tabLabel $child2 "Face Controls"
                        -tabLabel $child3 "Pose Maker"

                        $ESTabs;
setParent -top;



setParent..;
setParent..;

        frameLayout             -collapsable true -l "FK IK Controls" -w 400 -h 220 ESFKFrame;
        frameLayout -edit -cl 1 ESFKFrame;
        columnLayout ;
        rowColumnLayout -nc 2 -cw 1 325 -cw 2 65;
        attrFieldSliderGrp      -l " Left FK To IK" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                                -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                                -at ES_FK_IK_Switch.L_FKIK_Switch -ann "Blend between the left FK and IK
Skeletons";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.L_FKIK_Switch")  -ann
"Set key for left FK Blender";
        attrFieldSliderGrp      -l " Right FK To IK" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                                -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                                -at ES_FK_IK_Switch.R_FKIK_Switch -ann "Blend between the right FK and IK
Skeletons";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.R_FKIK_Switch") -ann
"Set key for right FK Blender";

        attrFieldSliderGrp      -l " L Shld Up" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                                -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                                -at ES_FK_IK_Switch.LShoulderUP -ann "Moves the Left Clavicle Up and
Down";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.LShoulderUP")  -ann
"Set key for Left Shoulder";

        attrFieldSliderGrp      -l " L Shld FWD" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                                -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                                -at ES_FK_IK_Switch.LShoulderFWD -ann "Moves the Left Clavicle Back and
Forth";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.LShoulderFWD")  -ann
"Set key for Left Shoulder";

        attrFieldSliderGrp      -l " R Shld Up" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                                -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                                -at ES_FK_IK_Switch.RShoulderUP -ann "Moves the Right Clavicle Up and
Down";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.RShoulderUP")  -ann
"Set key for Right Shoulder";
```

```
        attrFieldSliderGrp     -l " R Shld FWD" -cal 1  "left" -cal 2 "left" -cal 3 "left"
                               -cw 1 80 -cw 2 75 -cw 3 135 - w 300 -hmb 1 -pre 1
                               -at ES_FK_IK_Switch.RShoulderFWD -ann "Moves the Right Clavicle Back and
Forth";
        button -l "Key" -w 50 -bgc 1 .775 .775 -c ("setKeyframe"+" ES_FK_IK_Switch.RShoulderFWD")  -ann
"Set key for Right Shoulder";


        setParent..;
        rowColumnLayout -nc 2 -cw 1 195 -cw 2 195;
        button  -l "Match Left IK To Left FK" -bgc .8 .8 1 -ann "Moves the Left IK Arm to the Position of
the Left FK Arm" -c ESLIKtoLFK;
        button  -l "Match Left FK To Left IK" -bgc .8 .8 1  -ann "Moves the Left FK Arm to the Position
of the Left IK Arm" -c ESLFKtoLIK;;
        button  -l "Match Right IK To Right FK" -bgc 1 .8 .6 -ann "Moves the Right IK Arm to the Position
of the Right FK Arm" -c ESRIKtoRFK;;
        button  -l "Match Right FK To Right IK" -bgc 1 .8 .6 -ann "Moves the Right IK Arm to the Position
of the Right FK Arm" -c ESRFKtoRIK;;
        button  -l "Reset Left FK"     -bgc .775 1 .775 -ann "Resets all of the Left FK Controls to Zero"
-c ESResetLFK;
        button  -l "Reset Right FK"    -bgc .775 1 .775 -ann "Resets all of the Right FK Controls to Zero"
-c ESResetRFK;

        setParent..;
        setParent..;




setParent..;
setParent..;

        frameLayout            -collapsable true -l "Customize" -w 400 -h 220 ESCFrame;
        frameLayout -edit -cl 1 ESCFrame;
        columnLayout;

        frameLayout            -collapsable true -l "Face Picker" -w 400 -h 120 ESFSFrame;
        frameLayout -edit -cl 1 ESFSFrame;

        columnLayout;
                rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                text -l "        Face 1";
                text -l "        Face 2";
                text -l "        Face 3";
                text -l "        Face 4";
                text -l "        Face 5";
                setParent..;

        rowLayout -h 80 -w 400;
```

```
                image -i ( $ESLocation + "images/FaceButton.jpg") -h 80 -w 400;
                setParent..;

        rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                button - l "Apply" -bgc .775 1 .775 -c "setAttr ES_Face_CTRL.Select 0";
                button - l "Apply" -bgc .775 1 .775 -c "setAttr ES_Face_CTRL.Select 1";
                button - l "Apply" -bgc .775 1 .775 -c "setAttr ES_Face_CTRL.Select 2";
                button - l "Apply" -bgc .775 1 .775 -c "setAttr ES_Face_CTRL.Select 3";
                button - l "Apply" -bgc .775 1 .775 -c "setAttr ES_Face_CTRL.Select 4";
                setParent..;

                rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                text -l "        Amber";
                text -l "        Blue";
                text -l "        Green";
                text -l "        Brown";
                text -l "        Custom";
                setParent..;

        rowLayout -h 80 -w 400;
                image -i ( $ESLocation + "images/EyesButton.jpg") -h 80 -w 400;
                setParent..;
        rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                button - l "Apply" -bgc .775 1 .775 -c ES_AmberEyes;
                button - l "Apply" -bgc .775 1 .775 -c ES_BlueEyes;
                button - l "Apply" -bgc .775 1 .775 -c ES_GreenEyes;
                button - l "Apply" -bgc .775 1 .775 -c ES_BrownEyes;
                button - l "Apply" -bgc .775 1 .775 -c ES_CustomEyes;
                setParent..;
setParent..;
        setParent..;

        frameLayout            -collapsable true -l "Camouflage Color" -w 400 -h 120 ESCCFrame;
        frameLayout -edit -cl 1 ESCCFrame;

        columnLayout;
        rowColumnLayout -nc 2 -cw 1 200 ;
        text -l " Choose Coloring Method";
        attrEnumOptionMenu     -h 40 -w 80
                       -attribute Attribute_Storage.Method;
        setParent..;
                rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                text -l "        Snow";
                text -l "       Jungle";
                text -l "        Urban";
                text -l "       Desert";
                text -l "       Custom";
                setParent..;

        rowLayout -h 80 -w 400;
                image -i ( $ESLocation + "images/CamoButton.jpg") -h 80 -w 400;
```

```
                setParent..;

        rowColumnLayout -nc 5 -cw 1 80 -cw 2 80 -cw 3 80 -cw 4 80 -cw 5 80;
                button - l "Apply" -bgc .775 1 .775 -c ES_SnowCamo;
                button - l "Apply" -bgc .775 1 .775 -c ES_JungleCamo;
                button - l "Apply" -bgc .775 1 .775 -c ES_UrbanCamo;
                button - l "Apply" -bgc .775 1 .775 -c ES_DesertCamo;
                button - l "Apply" -bgc .775 1 .775 -c ES_CustomCamo;
                setParent..;
setParent..;
        setParent..;
        frameLayout             -collapsable true -l "Weapons and Accessories" -w 400 -h 220 ESWPFrame;
        frameLayout -edit -cl 1 ESWPFrame;

rowColumnLayout -nc 5 -cw 1 100 -cw 2 75 -cw 3 75 -cw 4 70 -cw 5 70;
        text -l "  Accessory";
        text -l "Visibilty";
        text -l "Match To";
        text -l "Select";
        text -l "Set Key";
        text -l "  Hand Gun";
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_M9_CTRL.Visible;
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_M9_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_M9_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_M9_CTRL.Match_to";
        text -l "  Rifle";
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_M4_CTRL.Visible;
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_M4_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_M4_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_M4_CTRL.Match_to";
        text -l "  Helmet";
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_Helmet_CTRL.Visible;
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_Helmet_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_Helmet_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_Helmet_CTRL.Match_to";
        text -l "  Frag Grenade";
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_Frag_CTRL.Visible;
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_Frag_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_Frag_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_Frag_CTRL.Match_to";
        text -l "  Smoke Grenade";
        attrEnumOptionMenu      -h 40 -w 80
                                -attribute ES_Smoke_CTRL.Visible;
```

```
        attrEnumOptionMenu     -h 40 -w 80
                               -attribute ES_Smoke_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_Smoke_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_Smoke_CTRL.Match_to";
        text -l "  Flash Grenade";
        attrEnumOptionMenu     -h 40 -w 80
                               -attribute ES_Flash_CTRL.Visible;
        attrEnumOptionMenu     -h 40 -w 80
                               -attribute ES_Flash_CTRL.Match_to;
        button -l "Select" -bgc .8 .8 1 -c "select -r ES_Flash_CTRL" ;
        button -l "Key" -bgc 1 .775 .775 -c"setKeyframe ES_Flash_CTRL.Match_to";
        setParent..;
        setParent..;
        setParent..;

setParent..;
setParent..;

        frameLayout -collapsable true -l "Modules" -w 400 -h 50 ;
        rowColumnLayout        -nc 4
                               -columnWidth 1 99
                               -columnWidth 2 99
                               -columnWidth 3 99
                               -columnWidth 4 99;
        button  -l "Graph Editor"      -c GraphEditor
                                       -ann "Hmm...what could that possibly refer to..?";
        button -l "Trax Editor"        -c CharacterAnimationEditor
                                       -ann "Nope, not a DJ Tool";
        button -l "Dope Sheet"         -c DopeSheetEditor
                                       -ann "Not what you might think...";
        button -l "Visor"              -c VisorWindow
                                       -ann "Whatever";


setParent..;
setParent..;


        frameLayout    -collapsable false  -l " " -w 400 -h 50;
        columnLayout;
        helpLine -w 380;

setParent..;
setParent..;
        showWindow;
                }


//      End of UI Creation                                      //
```

```
//      Procedure for Menu Item "MORE"                          //

global proc MySite()
        {
        string $MK3Dlink = "start explorer \"http://www.mk3d.com\"";
        system($MK3Dlink);
        }

//      End of procedure                                        //


//      proc for the help file                          //

global proc ESHelp()
        {
        global string $ESLocation;
        string $HelpLoc = ($ESLocation + "/html/Help.htm");
        system("load" + $HelpLoc );
        }

//      Procedure for the About menu                    //

global proc ESUIAbout ()
        {
        global string $KUIAboutWin = "KUIAboutWin";
        if (`window -exists $KUIAboutWin`)
                deleteUI  $KUIAboutWin;
        window  -t "ES UI About"
                        -w 175 -h 150
                        -rtf true
                        $KUIAboutWin;
        columnLayout;
        text -l "                   ES UI v1.0" ;
        text -l "==============================" ;
        text -l " " -al "center";
        text -l "  ES UI, including all textures, models"    ;
        text -l " and scripts created April/May 2005 by"            ;
        text -l "               M.S.R.Kiessling"            ;
        text -l "            WWW.MK3D.COM"                  ;
        text -l " ";
        button -l "OK" -w 175 -h 25 -c ("deleteUI  $KUIAboutWin");
        showWindow;
        }

//                                                      //
//            Match Left IK To Left FK                  //
//                                                      //

global proc ESLIKtoLFK ()
```

```
        {

        //      Set The Pole Vector                     //

        $KLFKPVT        = `getAttr ES_Left_FK_Elbow_Dummy.t`;
        setAttr ("ES_Left_Elbow_IK_CTRL." + "t")            $KLFKPVT[0] $KLFKPVT[1] $KLFKPVT[2];

        //      Set The Wrist CTRL                      //

        $KLFKWDT        = `getAttr ES_Left_FK_Wrist_Dummy.t`;
        $KLFKWDR        = `getAttr ES_Left_FK_Wrist_Dummy.r`;

        setAttr ("ES_Left_IK_Wrist_CTRL." + "t")     $KLFKWDT[0] $KLFKWDT[1] $KLFKWDT[2];
        setAttr ("ES_Left_IK_Wrist_CTRL." + "r")     $KLFKWDR[0] $KLFKWDR[1] $KLFKWDR[2];

        }


//                                                      //
//              Match Left FK To Left IK                //
//                                                      //

global proc ESLFKtoLIK ()

        {
        $KLIKSR = `getAttr ES_Left_IK_Shoulder.r`;
        $KLIKER = `getAttr ES_Left_IK_Elbow.r`;
        $KLIKWR = `getAttr ES_Left_IK_Wrist.r`;

        setAttr ("ES_Left_FK_Shoulder_CTRL." + "r")  $KLIKSR[0] $KLIKSR[1] $KLIKSR[2];
        setAttr ("ES_Left_FK_Elbow_CTRL." + "rz")     $KLIKER[2];
        setAttr ("ES_Left_FK_Wrist_CTRL." + "r")     $KLIKWR[0] $KLIKWR[1] $KLIKWR[2];


        }

//                                                      //
//              Match Right IK To Right FK             //
//                                                      //

global proc ESRIKtoRFK ()

        {

        //      Set The Pole Vector                     //

        $KRFKPVT        = `getAttr ES_Right_FK_Elbow_Dummy.t`;
        setAttr ("ES_Right_Elbow_IK_CTRL." + "t")           $KRFKPVT[0] $KRFKPVT[1] $KRFKPVT[2];

        //      Set The Wrist CTRL                      //
```

```
        $KRFKWDT       = `getAttr ES_Right_FK_Wrist_Dummy.t`;
        $KRFKWDR       = `getAttr ES_Right_FK_Wrist_Dummy.r`;

        setAttr ("ES_Right_IK_Wrist_CTRL." + "t")    $KRFKWDT[0] $KRFKWDT[1] $KRFKWDT[2];
        setAttr ("ES_Right_IK_Wrist_CTRL." + "r")    $KRFKWDR[0] $KRFKWDR[1] $KRFKWDR[2];
        }



//                                                   //
//             Match Right FK To Right IK            //
//                                                   //

global proc ESRFKtoRIK ()

        {
        $KRIKSR = `getAttr ES_Right_IK_Shoulder.r`;
        $KRIKER = `getAttr ES_Right_IK_Elbow.r`;
        $KRIKWR = `getAttr ES_Right_IK_Wrist.r`;

        setAttr ("ES_Right_FK_Shoulder_CTRL." + "r")         $KRIKSR[0] $KRIKSR[1] $KRIKSR[2];
        setAttr ("ES_Right_FK_Elbow_CTRL." + "rz")    $KRIKER[2];
        setAttr ("ES_Right_FK_Wrist_CTRL." + "r")    $KRIKWR[0] $KRIKWR[1] $KRIKWR[2];

        }



global proc ESResetLFK ()
        {
        setAttr "ES_Left_FK_Shoulder_CTRL.rotateX" 0;
        setAttr "ES_Left_FK_Shoulder_CTRL.rotateY" 0;
        setAttr "ES_Left_FK_Shoulder_CTRL.rotateZ" 0;
        setAttr "ES_Left_FK_Elbow_CTRL.rotateZ" 0;
        setAttr "ES_Left_FK_Wrist_CTRL.rotateX" 0;
        setAttr "ES_Left_FK_Wrist_CTRL.rotateY" 0;
        setAttr "ES_Left_FK_Wrist_CTRL.rotateZ" 0;
        }

global proc ESResetRFK ()
        {
        setAttr "ES_Right_FK_Shoulder_CTRL.rotateX" 0;
        setAttr "ES_Right_FK_Shoulder_CTRL.rotateY" 0;
        setAttr "ES_Right_FK_Shoulder_CTRL.rotateZ" 0;
        setAttr "ES_Right_FK_Elbow_CTRL.rotateZ" 0;
        setAttr "ES_Right_FK_Wrist_CTRL.rotateX" 0;
        setAttr "ES_Right_FK_Wrist_CTRL.rotateY" 0;
        setAttr "ES_Right_FK_Wrist_CTRL.rotateZ" 0;
        }

//      End Of IK FK switcher Procedures             //
```

```
//             Reset the Face Cam                    //

global proc ESResetFaceCam ()

        {
        setAttr "ES_ESUI_FaceCam.translateX" 0;
        setAttr "ES_ESUI_FaceCam.translateY" 0;
        setAttr "ES_ESUI_FaceCam.translateZ" 0;
        setAttr "ES_ESUI_FaceCam.rotateX" 0;
        setAttr "ES_ESUI_FaceCam.rotateY" 0;
        setAttr "ES_ESUI_FaceCam.rotateZ" 0;
        }


global proc CreateFacePose ()
        {

global string $KFPWin = "KFPWin";
        if (`window -exists $KFPWin`)
        deleteUI $KFPWin;
window  -t "ES CREATE FACE POSE" -w 200 -h 100
                -rtf true -mbv false
                $KFPWin;

columnLayout -cw 200;
        text            -l "Please enter a name for your new pose" -al "center";

        textField -w 200 -tx "ENTER POSE NAME" KFPN;
        text            -l " ";
        button  -l "CREATE FACE POSE" -w 200 -h 50 -bgc .8 .8 1
                        -c ESFacePose;
        button  -l "EXIT" -bgc 1 .775 .775
                        -w 200 -h 25
                        -c ("deleteUI " + $KFPWin);
        showWindow;
        }

global proc CreateFaceClip ()
        {
        setCurrentCharacters( { "ES_FaceSet" } );
        CreateClipOptions;
        }

global proc ESFacePose ()
        {
        global string $KFPWin;
        global string $KFPN;
        global string $ESLocation;
        string $ESPoseNameFace;
        $ESPoseNameFace = `textField -q -tx KFPN`;
```

```
        print ($ESPoseNameFace);

        if ($ESPoseNameFace == "ENTER POSE NAME")
                {
                error "Please Specify Unique Name";
                }
        else    if ($ESPoseNameFace == "")
                {
                error "Please Specify  Name";
                }
        global string $ESCharName;
        $ESCharName = "ES_FaceSet";
        setCurrentCharacters( { "ES_FaceSet" } );
        pose -name ($ESPoseNameFace) $ESCharName;
        select -r ($ESPoseNameFace);
        file -op "v=0" -typ "mayaBinary" -es ($ESLocation +"ClipsAndPoses/" + $ESPoseNameFace +".mb");
        deleteUI $KFPWin;

        }


//      Procs to select the Character Sets           //

global proc ESNone ()
        {
setCurrentCharacters( {} );
        }
global proc ESBody ()
        {
setCurrentCharacters( { "ES_BodySet" } );
global string $ESCharName;
$ESCharName = "ES_BodySet";
        }

global proc ESBodyPose ()
        {
global string $KBPWin = "KBPWin";
        if (`window -exists $KBPWin`)
        deleteUI $KBPWin;
window  -t "ES CREATE BODY POSE" -w 200 -h 100
                -rtf true -mbv false
                $KBPWin;

columnLayout -cw 200;
        text            -l "Please enter a name for your new pose" -al "center";

        textField -w 200 -tx "ENTER POSE NAME" KBPN;
        text            -l " ";
        button  -l "CREATE BODY POSE" -w 200 -h 50 -bgc .8 .8 1
                        -c ESBodyPoser;
        button  -l "EXIT" -bgc 1 .775 .775
```

```
                        -w 200 -h 25
                        -c ("deleteUI " + $KBPWin);
        showWindow;
        }


global proc ESBodyPoser ()
        {
        global string $KBPWin;
        global string $KBPN;
        global string $ESLocation;
        string $ESPoseNameBody;
        $ESPoseNameBody = `textField -q -tx KBPN`;
        if ($ESPoseNameBody == "ENTER POSE NAME")
                {
                error "Please Specify Unique Name";
                }
        else    if ($ESPoseNameBody == "")
                {
                error "Please Specify  Name";
                }
        global string $ESCharName;
        setCurrentCharacters( { "ES_BodySet" } );
        pose -name ($ESPoseNameBody) $ESCharName;
        select -r ($ESPoseNameBody);
        file -op "v=0" -typ "mayaBinary" -es ($ESLocation +"ClipsAndPoses/" + $ESPoseNameBody +".mb");

        deleteUI $KBPWin;
        }

global proc CreateBodyClip ()
        {
        global string $ESCharName;
        setCurrentCharacters( { "$ESCharName" } );
        CreateClipOptions;
        }


global proc ESResetPoseCam ()
        {
setAttr "ES_ESUI_PoseCam.translateX" 0;
setAttr "ES_ESUI_PoseCam.translateY" 0;
setAttr "ES_ESUI_PoseCam.translateZ" 0;
setAttr "ES_ESUI_PoseCam.rotateX" 0;
setAttr "ES_ESUI_PoseCam.rotateY" 0;
setAttr "ES_ESUI_PoseCam.rotateZ" 0;
        }
```

```
//      The following procedures cover the vertex coloring of the eyes and the camouflage //

//      Procedures for the eyes       //

global proc ES_AmberEyes ()
        {
        select -r R_Eye_Pupil ;
        select -tgl L_Eye_Pupil ;
        polyColorPerVertex -rgb .757 .593 .382;
        select -cl  ;
        setAttr "Pupil_tex.colorGain" -type double3 .757 .593 .382;
        }
global proc ES_BlueEyes ()
        {
        select -r R_Eye_Pupil ;
        select -tgl L_Eye_Pupil ;
        polyColorPerVertex -rgb .761 .761 .887;
        select -cl  ;
        setAttr "Pupil_tex.colorGain" -type double3  .761 .761 .887;
        }
global proc ES_GreenEyes ()
        {
        select -r R_Eye_Pupil ;
        select -tgl L_Eye_Pupil ;
        polyColorPerVertex -rgb .508 .634 .508;
        select -cl  ;
        setAttr "Pupil_tex.colorGain" -type double3 .508 .634 .508;
        }
global proc ES_BrownEyes ()
        {
        select -r R_Eye_Pupil ;
        select -tgl L_Eye_Pupil ;
        polyColorPerVertex -rgb .489 .334 .257;
        select -cl  ;
        setAttr "Pupil_tex.colorGain" -type double3 .489 .334 .257;
        }
global proc ES_CustomEyes ()
        {colorEditor;
        if (`colorEditor -query -result`)
                {
                float $ES_Eyes_Color[];
                $ES_Eyes_Color = `colorEditor -query -rgb`;
                select -r R_Eye_Pupil ;
                select -tgl L_Eye_Pupil ;
                polyColorPerVertex -rgb $ES_Eyes_Color[0] $ES_Eyes_Color[1] $ES_Eyes_Color[2] ;
                select -cl  ;
                setAttr "Pupil_tex.colorGain" -type double3 $ES_Eyes_Color[0] $ES_Eyes_Color[1]
$ES_Eyes_Color[2] ;
                }
        }
//      End of eyes coloring procedures       //
```

```
// Procedures for applying vertex color to the uniform and helmet   //

global proc ES_SnowCamo ()
        {
        $ES_ShadingMethod = `getAttr Attribute_Storage.Method`;
        if ($ES_ShadingMethod == 1)
        {
        setAttr "Camo_tex.frameExtension" 1;
        setAttr "Helmet_tex.frameExtension" 1;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 1;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 1 1 ;
        setAttr "Helmet_tex.colorGain" -type double3 1 1 1 ;
        }
else if ($ES_ShadingMethod == 0)
        {
        setAttr "Camo_tex.frameExtension" 1;
        setAttr "Helmet_tex.frameExtension" 1;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 1;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 1 1 ;
        setAttr "Helmet_tex.colorGain" -type double3 1 1 1 ;
        }
        }
global proc ES_JungleCamo ()
        {
        $ES_ShadingMethod = `getAttr Attribute_Storage.Method`;
        if ($ES_ShadingMethod == 1)
        {
        setAttr "Camo_tex.frameExtension" 2;
        setAttr "Helmet_tex.frameExtension" 2;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 1;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 1 1 ;
        setAttr "Helmet_tex.colorGain" -type double3 1 1 1 ;
        }
else if ($ES_ShadingMethod == 0)
        {
        setAttr "Camo_tex.frameExtension" 1;
        setAttr "Helmet_tex.frameExtension" 1;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 0.613154 0.801 0.565506 ;
        select -cl  ;
```

```
        setAttr "Camo_tex.colorGain" -type double3 0.613154 0.801 0.565506 ;
        setAttr "Helmet_tex.colorGain" -type double3 0.613154 0.801 0.565506 ;
        }
        }
global proc ES_UrbanCamo ()
        {
        $ES_ShadingMethod = `getAttr Attribute_Storage.Method`;
        if ($ES_ShadingMethod == 1)
        {
        setAttr "Camo_tex.frameExtension" 3;
        setAttr "Helmet_tex.frameExtension" 3;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 1;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 1 1 ;
        setAttr "Helmet_tex.colorGain" -type double3 1 1 1 ;
        }
else if ($ES_ShadingMethod == 0)
        {
        setAttr "Camo_tex.frameExtension" 1;
        setAttr "Helmet_tex.frameExtension" 1;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 0.516477 0.516477 0.779;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 0.516477 0.516477 0.779;
        setAttr "Helmet_tex.colorGain" -type double3 0.516477 0.516477 0.779;
        }
        }
global proc ES_DesertCamo ()
        {
        $ES_ShadingMethod = `getAttr Attribute_Storage.Method`;
        if ($ES_ShadingMethod == 1)
        {
        setAttr "Camo_tex.frameExtension" 4;
        setAttr "Helmet_tex.frameExtension" 4;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 1;
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 1 1 ;
        setAttr "Helmet_tex.colorGain" -type double3 1 1 1 ;
        }
else if ($ES_ShadingMethod == 0)
        {
        setAttr "Camo_tex.frameExtension" 1;
        setAttr "Helmet_tex.frameExtension" 1;
        select -r ES_Accesories|ES_Helmet ;
        select -tgl ES_Geometry|ES_Body|ES_Body ;
        polyColorPerVertex -rgb 1 1 0.786;
```

```
        select -cl  ;
        setAttr "Camo_tex.colorGain" -type double3 1 .967 .88;
        setAttr "Helmet_tex.colorGain" -type double3 1 .967 .88;
        }
        }
global proc ES_CustomCamo ()
        {
        $ES_ShadingMethod = `getAttr Attribute_Storage.Method`;
        if ($ES_ShadingMethod == 0)
        {
        setAttr "Camo_tex.frameExtension" 1;
        colorEditor;
        if (`colorEditor -query -result`)
                {
                float $ES_Camo_Color[];
                $ES_Camo_Color = `colorEditor -query -rgb`;
                select -r ES_Accesories|ES_Helmet ;
                select -tgl ES_Geometry|ES_Body|ES_Body ;
                polyColorPerVertex -rgb $ES_Camo_Color[0] $ES_Camo_Color[1] $ES_Camo_Color[2];
                select -cl  ;
                setAttr "Camo_tex.colorGain" -type double3 $ES_Camo_Color[0] $ES_Camo_Color[1]
$ES_Camo_Color[2];
                setAttr "Helmet_tex.colorGain" -type double3 $ES_Camo_Color[0] $ES_Camo_Color[1]
$ES_Camo_Color[2];

                }
        }
        else if ($ES_ShadingMethod == 1)
                {
                string $ESSEWin = "ESSEWin";
                if (`window -exists $ESSEWin`)
                deleteUI $ESSEWin;
        window  -tlb on -mb true -rtf true
                        -w 150 -h 50
                        -t "ERROR!"
                        $ESSEWin;
                columnLayout;
                        text -l "    Custom only works with";
                        text -l "     Texture selected as";
                        text -l "        Shading Method";
                        button -l "OK" -bgc 1 .5 .5  -w 150 -c "deleteUI $ESSEWin";
                showWindow;
                }
        }

// End of vertex coloring for the uniform           //

// End of vertex coloring                           //

// End of UI script                                 //
```