

To the Graduate Faculty:

I am submitting herewith a project written by Nathan John Bolt entitled “Modular Asset Design and Creation.” I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Interactive Technology in Digital Game Development, with Specialization in Art Creation.

---

Chad Walker, Faculty Supervisor

We have read this Project  
and recommend its acceptance:

---

---

---

Accepted for the Faculty:

---

Executive Director, The Guildhall at SMU

MODULAR ASSET DESIGN AND CREATION

A Project Presented to the Graduate Faculty of  
The Guildhall at Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Master of Interactive Technology

in Digital Game Development

with

Specialization in Art Creation

by

Nathan John Bolt

B.A, Ball State University, 2005

April 14, 2008

Modular Asset Design and Creation

Advisor: Professor Chris Jagers

Master of Interactive Technology degree conferred

Project completed April 14, 2008

This project examines the growing demand for assets with the advent and subsequent popularity of Massively Multiplayer Online Games, specifically those of the RPG genre.

Modular assets solve many of the problems generated by this increased demand by increasing the productivity of artists along with the reusability of the assets those artists generate, allowing a small number of modules to be used to generate a very large number of final assets

The research explores the phenomenon of modular assets in MMOs in terms of visual styles represented by past and recent games. It also examines the methods that have been used to break assets into separate modules in addition to the specific application of modular design as it applies to player control of visual assets and customization thereof.

The development of a modular vehicle clearly demonstrates the utility and benefit of using a modular design approach to asset creation by producing a wide variety and large number of configurable final assets from a relatively small number of individual modules. Additionally, the process of designing a modular asset is examined in detail. Several key concepts of modular design are discussed as well as different approaches to both designing new modular assets and understanding the underlying structures of existing modular assets.

## TABLE OF CONTENTS

1. INTRODUCTION .....	2
1.1 Motivation.....	5
1.2 Field Review .....	6
1.2.1 Catalogue of Visual Styles.....	6
1.2.2 Parameters of Customization .....	12
1.2.3 Summary .....	16
1.3 Proposal.....	16
1.3.1 Overview .....	17
1.3.2 Art Style.....	17
1.3.3 Process .....	18
1.3.4 Display .....	19
1.3.5 Schedule.....	19
2. METHODOLOGY .....	21
2.1 Concept .....	21
2.2 Modeling.....	22
2.3 Unwrapping and Texturing.....	24
3. RESULTS AND ANALYSIS.....	28
3.1 Final Output .....	28
3.2 Analysis.....	29
4. DISCUSSION/CONCLUSIONS.....	31
4.1 Scope.....	31
4.2 Design Methodology.....	33
4.3 Structure and Modular Mapping.....	36
4.3.1 Points of Contact.....	36
4.3.2 Modular Flow.....	38
4.3.3Types of Modules .....	39
4.3.4 Level of Modularity .....	41
4.3.5 Modular Mapping .....	42
4.4 Granularity .....	45
5. REFERENCES .....	47

## **1. Introduction**

As video games become increasingly large and complex in scope the amount of money and time necessary to produce video games rises dramatically. One of the single largest contributing factors to both time requirements and budget concerns in video games is that of visual assets. Video games which are larger in scope by definition require a great number of art assets. Additionally, these are assets become more and more complex and difficult to produce every year and with every advance in technology. This creates a bottleneck in video game design as the limited resources of an art team and limited budget available to outsource art requirements attempt to complete the sheer volume of work set before them.

One of the potential solutions to this problem that is only beginning to be fully explored is the concept of a modular asset. While modular assets have been around for quite some time and used in a variety of applications, their full worth has only begun to be realized. Modular assets allow a fixed number of artists working within a fixed amount of time to produce a near-limitless volume of work.

Before we can begin to understand what a modular asset is (as it relates to video game development), we must first understand exactly what a module is. Dictionary.com defines a module as “a separable component, frequently one that is interchangeable with others, for assembly into units of differing size, complexity, or function.” As such, a modular asset would be one that is composed of modules. However, this alone is not what makes modular assets so useful. Any sufficiently complex model can be broken down into its component modules which can then be produced independently, yet this does not satisfy the ‘nearly limitless volume of work’ mentioned earlier. The key phrase

in the definition of a module that gives modular assets their utility and flexibility is ‘frequently one that is interchangeable with others’. By having alternate modules that may be substituted for an original we open up the possibility of producing a new asset at the cost of only making a new module. With a sufficient number of modules, this can lend itself to nearly limitless possibilities.

A modular asset then can be more accurately thought of as a large group of modules which can be configured to produce a final asset. The final asset itself does not use all of the constituent modules of the entire group, but rather only a small selection. Also, the total number of possible final assets (based on the number of possible configurations) is quite large when compared to the number of constituent modules. As such, a modular asset does not replace a single asset in video game development, but instead an entire group or collection of related assets.

As has already been discussed, one of the largest and most obvious benefits of using modular assets in place of more traditional non-modular assets is the sheer volume of final assets that may be produced from the relatively small amount of work necessary to produce the constituent modules of that modular asset. However, there are other benefits as well, and these benefits form the majority of the reasons modular assets have been used in the past.

Probably the single largest contributing factor to the use of modular assets is the concept of player customization. If a large number of modules are produced and then the player of the game is given access to and control over the configuration of these modules he or she then gains some control over the appearance of a visual asset within the video game, making it unique and customized to that player. While this has been used across a

variety of genres, it has been traditionally used by Role-Playing Games (RPGs) and is currently seeing its largest growth of usage in Massively Multiplayer Role-Playing Games (MMORPGs). These games in particular benefit greatly from the visual variety afforded to them by using modular assets for player-character avatars and their equipment, either through the sheer volume of combinations (as is the case in more traditional single-player RPGs), the customization afforded to the player, or the differentiation possible between final assets (as is the case with MMORPGs in which it is very helpful to differentiate different player-character avatars, allowing them to identify each other visually, rather than with names or text).

However, visual variation is not the sole reason modular assets have been used, or remain useful. While many games give players the opportunity to customize their characters or other assets on a purely visual level, often this customization runs much deeper, allowing the player to customize the gameplay to some degree as well. Oftentimes a single game may afford both types of customization. In the case of the popular MMORPG *World of Warcraft* players are given the opportunity to customize their avatar's face and hairstyle which have absolutely no impact on gameplay whatsoever. However, players may also customize their characters clothing and weaponry, each of which carry a significant impact on gameplay in addition to the appearance of the avatar. This application helps give players visual feedback to their gameplay choices in addition to their control over the appearance of customizable assets.

Despite their many benefits, modular assets are not equally suited to every application or video game. A certain set of criteria must be met for a modular asset to be advantageous over normal single asset production. The two player benefits mentioned

earlier (player control over appearance and gameplay customization feedback) are in and of themselves reason enough to use a modular asset, and up until recently have been the two primary reasons to use a modular asset over a traditional asset. However, the third and potentially most advantageous reason to use a modular asset, which is the benefit of producing a large volume of work at decreased cost in man-hours, must be carefully weighed. While modular assets can take far less time in actual production than a group of traditional assets, modular assets require much more time in planning than a traditional asset to make sure all of the modules are consistent in style and work and fit together properly and coherently. The point at which modular assets become viable is when this pre-production time becomes a relatively insignificant concern, i.e. when there are a very large number of assets that all share a significant number of modules in common. If there are a relatively small number of assets, or if this group of assets share little in common (i.e. few modules in common), the benefit of modular design is quickly outweighed by the time necessary to plan the production.

## **1.1 Motivation**

My motivation in undertaking this project has been largely twofold. My initial interest in modular asset stems from my experience as a player of video games which offer character customization. This has long been of particular interest to me, not only in exploring the wealth of options available, but in my interest in pushing the envelope and expanding the scope and application of to allow even greater player control over customization. My second interest in modular assets is to more fully understand the

process of design and execution of these assets and to explore what other applications modular assets can have along with the benefits and drawbacks they offer.

## **1.2 Field Review**

My field review contains an in-depth analysis of the phenomenon of player-character avatar customization within the MMORPG genre. As discussed earlier, this is one of the largest scale applications of a modular asset in video game developments and as such has been very informative of the design and execution of my own modular asset. Of particular interest is the analysis and commentary on the parameters of customization available within current MMORPGs and the impact this has on player perception, scope of the modular asset, and overall visual interest and appeal.

Since the earliest video games, players have commonly been represented within the game world as a character, sometimes called an avatar. With the advent of Massive Multiplayer Online games (MMOs), the emphasis and importance placed upon these avatars has greatly increased. Whereas early avatars were simple representations of a player, MMOs often offer a wide variety choice and customization to make each individual's avatar unique. Also, as technology has increased and allowed for more sophisticated graphics, the variety of visual styles possible for these avatars has likewise increased. In this paper, I will discuss the phenomenon of the player character avatar, both in terms of visual styles and parameters of customization.

### **1.2.1 Catalogue of Visual Styles**

With the large number of MMOs currently available in the market today, it is difficult to find a way to categorize even their content, much less the visual style of this large library of games. However, to simplify the process, I will discuss the visual styles of a few large categories of games, grouped roughly chronologically as they relate to a modern day setting. First, I will look at some of the MMOs set in future together as a sci-fi genre and discuss some of the differences and similarities between these game's visual styles. Next, games set in the modern time frame will be examined. Finally, games which share a pre-modern setting, from actual historical settings to more fantasy inspired ones will be discussed.

One of the more popular visual styles in video games in general is that of sci-fi, or a futuristic look. Several MMOs which take place in a sci-fi setting could better be described as 'earthy sci-fi'. This style is characterized by a very 'realistic' style, where everything has a lived-in appearance, caked in grime and mud, with very few assets appearing shiny or brand-new. This carries over to character styles, most notably in terms of character races. Races present in games of this visual style are widely varied and imaginative, often departing significantly from the base human model in both proportion and visual make-up. Color scheme is also an important aspect in this style, with colors tending towards warmer reds and browns in general, and drawing particular attention to colors which break with the general scheme, such as blues and purples. Star Wars Galaxies<sup>i</sup> is a particularly good example of this visual style, as is, to a lesser degree, Anarchy Online<sup>ii</sup>.

On the other end of the sci-fi spectrum in terms of visual style is that of Eve Online<sup>iii</sup>. Whereas Star Wars Galaxies has an earthy and varied visual style, what could

be termed a 'clean sci-fi' style is largely uniform and stereotypically futuristic. This visual style is characterized by a much more mechanical approach, with organic objects appearing rarely in any assets. The color scheme tends toward a cooler spectrum of dull purples and intense blues. Even when other colors are present, they are often cooled, giving everything somewhat of a sterilized look. This is particularly well presented by Eve Online as nearly all gameplay takes place in the vacuum of space, with other characters rarely being seen in person, save for small communiqué icons.

Although technically set in the future, The Matrix Online takes place in a facsimile of the modern world, making its visual style more modern than sci-fi. However, with that in mind, The Matrix Online's visual style has much in common with the clean sci-fi look, and can probably be best summed up in a single word: sleek. All character visual assets look brand-new and un-used, often too perfect or idealized to even exist in the real-world, even on a store shelf. Character clothing tends toward leather and vinyl, with the one major unifying factor being that nearly all assets are slick and shiny. Drawing from the visual style of the movies on which it is based, the color scheme is startlingly uniform green, further driving home the point that the world in which the game takes place is an idealized, 'too-perfect' version of our own world<sup>iv</sup>.

Like The Matrix Online, City of Heroes<sup>v</sup> and City of Villains<sup>vi</sup> are similarly set in a world loosely based on modern reality. However, while The Matrix Online draws on a specific series of movies for its visual vocabulary, the Cities games draw upon a set of stereotypes that have coalesced to form the comic book genre. At first glance this style seems largely chaotic and un-uniform, but upon closer inspection a few generalities can be gleaned which define it as a cohesive style. The most cohesive element is also that

which makes this style so tumultuous: exaggeration. This applies to nearly every aspect of character visuals, from proportions to color scheme. Any individual will be cohesive when considered by itself within this element of exaggeration. That is to say, if a character is meant to look strong or super heroic, his proportions will be exaggerated in accordance, with much more emphasis placed on the upper body musculature, broad shoulders, and commonly a large chin. However, when considered next to characters exaggerated in a different manner, dissimilarities quickly become apparent. Color scheme follows in the same pattern, in that any given character will likely have a cohesive color scheme individually, only to clash wildly with the color scheme of other characters. However, almost uniformly, colors are very bright and very highly saturated.

While not technically a game, Second Life<sup>vii</sup> deserves mention and examination along with other semi-modern set MMOs as it shares, and in some cases carries to an extreme, the importance of character customization and individuation. However, unlike nearly every other MMO currently commercially available, the majority of visual content present in Second Life is user made and contributed. As such, defining a cohesive visual style is even more challenging than with the Cities games, though one is still present. Default character avatars, being one of the few developer-made assets, tend to follow a set rule of simplification, tending toward a cartoon feel with over-large eyes, mouths, heads, and hands while arms and legs are often represented as nothing more than simple jointed tubes. User-contributed assets tend toward a general scheme of cartoon-like simplification, largely attributable to the fact that the modeling tools included with the program are best adapted to producing simple shapes. Beyond these generalities,

however, there is next to no unifying factor among visual assets, though near limitless flexibility and customization with respect to character avatars.

Beyond modern and sci-fi genres, a large percentage of MMOs tend towards pre-modern and fantasy visual motifs. Within this genre, a style rapidly growing in popularity is one easily recognizable in games such as *Guild Wars*<sup>viii</sup>, *Lineage 2*<sup>ix</sup>, and *Rappelz*<sup>x</sup>. Though semi-realistic, this style also has a healthy amount of idealization and differs from standard human proportions in some significant ways. In general, figures are slim and elongated, while features alternate sharply between smooth roundness and slightly subdued angularity. Of particular note in this visual style is the attention to detail. There are very few broad expanses of a flat color or texture. Instead, in an almost *horror vacui* manner, any available areas are filled with small details and decoration, giving this style a signature look of high contrast in small areas.

In nearly direct contrast to this style is that of one of the most popular MMOs of all time: *World of Warcraft*<sup>xi</sup>. Whereas the majority of other MMOs tend to stick pretty closely to real human proportions, *World of Warcraft* has its own unique set of proportions which, when they stray near realistic ones, do so completely by coincidence. In general, this visual style tends toward big and bulky, applying equally to figure proportions as well as costumes, weapons, and all visual assets. Most assets also tend towards a bottom-heavy hourglass shape, being narrower at the top than the bottom and thinnest in the middle. Like a cartoon or comic book, colors tend to be highly saturated, although not to the same degree as *City of Heroes*. Also, in direct contrast to games like *Guild Wars* or *Lineage 2*, *World of Warcraft* has few areas of high detail or extremely

high contrast. Details, like all other elements, tend to be big and bulky, and as such are spaced far apart, allowing for large areas of relatively flat texture.

As much as World of Warcraft approaches a cartoon-like visual style, there are other games which have given over entirely to a cartoon look. Notable among these are Maple Story<sup>xii</sup> and Ragnarok Online<sup>xiii</sup>. Despite the fact that both of these games are sprite-based 2-D games (which will be discussed in depth later), they follow a simplification of character to the most basic essentials. Heads become spheres or balls, fingers are an unnecessary detail, eyes are enlarged to take up half of the head, mouths are non-existent unless open. Colors and detail are likewise simplified, with most elements being composed of a few simply defined shapes of broad, flat color with little to no variation whatsoever.

In addition to these three broad categories of sci-fi, modern, and fantasy, a number of games do not fall neatly into a single profile, though they often share much in common with games of these genres. Puzzle Pirates<sup>xiv</sup>, for instance, is not fantasy in nature, but is every bit as much a cartoon style as Ragnarok Online or Maple Story, possibly even more so as characters are more reminiscent of Playmobil figurines than actual people. Toontown<sup>xv</sup> draws on a series of conventions in a manner similar to City of Heroes. However, instead of comic books, Toontown looks to color cartoons from the 50's and 60's, particularly those of Warner Bros. and Disney.

The visual style of MMOs does not exist in a vacuum and is heavily influenced and informed by a number of other factors. As technology plays a pivotal role in the look of a game, the target platform and overall rendering capabilities play an enormous role in defining the visual style of a game. Games which are 2-D, such as side-scrolling MMOs

like Maple Story, must by nature be much more simplified and cartoon-like. Isometric or 2½-D games like Ragnarok or Westward Journey<sup>xvi</sup> can allow for greater detail in characters or even blend 2-D and 3-D elements into a nearly seamless whole. Side-scrolling and isometric games, despite their limitations, allow for greater control of a unified visual style as they not only have complete control of what a player sees, but also of what viewing angles content will be seen from.

While 3-D games allow for much more viewing freedom from the player perspective, there is still a great deal of variation in what a render engine can support and what the target hardware is capable of. On one end of the spectrum are games like World of Warcraft, targeted at machines with very low render capabilities. These games by necessity tend toward blockier forms and more simple textures with minimal detail or contrast from lighting. On the other end of the spectrum are the so-called next-gen games such as Lord of the Rings Online<sup>xvii</sup> or the anticipated Warhammer Online<sup>xviii</sup>. As these games are targeted at hardware capable of much more impressive visual feats, they can choose to include a wide variety of advanced render techniques, such as specular, normal, and sub-surface mapping. This allows for the potential of a much more rich and detailed visual environment, but can also overload the player with a high-contrast and constantly varying playing environment.

### **1.2.2 Parameters of Customization**

In a discussion of MMO character and avatar, an element rivaling visual style is character customization, and the parameters/ mechanics by which that is defined. Every MMO must have some way to vary the appearance of different character, if only out of a

necessity to differentiate between different players in the virtual environment. However, considering the degree to which this may logically affect gameplay, many games carry this to an extreme, driven primarily out of player desire.

An overarching concept among all of the different aspects of a character which a player may customize is the manner in which they may customize it. As exhibited in games thus far, there are two major approaches to this question, with a potential third only beginning to form on the fringes. The most direct solution is to provide a player with a limited set of choices, which we may refer to as the component method. While this may apply at different levels, it rarely occurs in MMOs on a macro level. That is, often a character avatar's face will have components of facial features, hair styles, and coloration to choose from, with a limited set of predetermined choices in each component.

The second approach to the problem is the slider method. Instead of having a limited set of predetermined choices, a given feature for customization will be broken down into a set of parameters with minimum and maximum values defined for each parameter which can then be individually customized by the player. This is most often seen as a set of facial sliders which allow players to individually alter things such as mouth size, face width, distance between eyes, brow height, and so on. However, Second Life is notable in that nearly every conceivable parameter of character appearance, from overall build to hairstyle is broken into such a set of parameters, easily numbering in the hundreds, allowing for nearly infinite customization.

The third solution, which is only beginning to appear and may never fully develop as a workable alternative, is that of player-driven content. This allows players with sufficient knowledge and skill to develop and upload their own custom assets. This

approach allows for truly limitless customization and individuation at the cost of developer control of content and a cohesive visual style.

Another important note in the approach to character customization is that different solutions may be employed at different steps in the process. For example, a given game may have a component selection model for determining the character's race and then allow the player a slider customization of the character's face as defined by racial parameters.

Within the scope of all elements of an avatar that a player may customize, race is usually one of the first and largest choices a player faces. While it is not present in all MMOs, many present this as a critical element of character creation, and not only affects appearance more than any single other parameter, but also often affects gameplay to a significant degree as well. In fact, one of the only genres that this element is significantly (and understandingly) lacking from is that of the modern-set MMO. The variety of races a character may choose from also often serves as an overall visual indicator of the game itself, as many games are largely defined by the scope of races present. In most fantasy games these choices are limited to a set of fantasy tropes, such as elves, dwarves, orcs and trolls, whereas as other fantasy MMOs have a completely unique set of races. Thus far, there exists no set of unified trope races in sci-fi MMOs, and as such commonly have the widest variety and most imaginative races of any MMO.

While race is probably the single largest factor on overall character appearance, the character's face and head is often the area which receives the most attention in terms of customization and has the most options and parameters. With very few exceptions, MMOs allow for a minimum of three parameters, with facial features and hairstyle being

fairly universal amongst these. In games that have a slider element to their customization options, this is commonly the only place it is present, allowing for almost complete customization of the characters face and head. It is also interesting to note that, proportionally, the character's face is between 1/6<sup>th</sup> and 1/8<sup>th</sup> of the overall character appearance and one that the player rarely sees while playing the game. This is especially true of games which allow characters to don a helmet, in which case the face and head are entirely obscured.

The remainder of the character's appearance is the character's body, which games have a wide variety of customization for, and varying degrees of affecting gameplay. While a few games allow players to customize the character's overall body type and build, most do not. Instead, nearly omnipresent in MMOs as a customizing factor is that of gear, better defined as clothing, costume, armor, weapons, and other effects a character may carry on their person at all time. This element often follows the component model of customization, but often has a very large number of possibilities for any given component, though few if any of these options are available upon character creations and must instead be explored through gameplay instead. The actual parameters present vary from game to game; some follow a pattern defined loosely by armor pieces in most fantasy games, with World of Warcraft or Everquest being good examples. Others may follow actual clothing options much more closely, allowing a player only the options of shirt, pants, shoes, and perhaps a hat, as is the case with Puzzle Pirates.

These options also have varying impacts on actual gameplay. In most fantasy or martial games, a character's gear has a significant impact on gameplay, and appearance may be a secondary consideration. In other games, these parameters may be purely visual

in nature, and have no impact whatsoever on how the game is played. Maple Story and Puzzle Pirates are excellent examples of the latter.

Another interesting aspect of character appearance customization, especially as exemplified by both Maple Story and Puzzle Pirates, is the impact of character appearance on business model and economic considerations. Whereas customization options may have little effect on gameplay, they may sometimes form a cornerstone of the economic model of a given MMO. In Maple Story, for instance, players may purchase virtual items which affect and alter the appearance of their character directly from the developer for real money. Likewise, on some servers of Puzzle Pirates, nearly any item a player purchase to affect their character's appearance has a cost in real dollars, or some derivative thereof. Games such as these usual depart entirely from the subscription model of other MMOs, relying instead on these micro-transactions to generate revenue for the developer.

### **1.2.3 Summary**

Whereas avatar appearance and customization may have once been a minor consideration in video games, Massive Multiplayer Online video games have quickly advanced it to being a critical element of many video games. Characters form a critical part of a video game's overall visual style. Additionally, many MMOs have myriad options available for a player to make their avatar unique and separate from the masses of other players within the video game's virtual environment.

### **1.3 Proposal**

### **1.3.1 Overview**

For my Master's project, I will concept, model, and texture two modular, customizable game assets: a character and a vehicle. To further define modular and customizable, each of these assets will be created from an amalgam of a series of individual pieces and assets. In the case of the character these assets would be elements of clothing, armor, and other pieces worn or carried. In the case of the vehicle, the breakdown is more along of the lines of the function, appearance, and purpose of the vehicle. Some examples modules of the vehicle would be chassis size and style, seating solution, method of locomotion, and so forth. The notion of customization is fairly well-covered by the modular approach to asset creation, but beyond this individual textures will be customizable as well, in a modular fashion. Customizable textures includes such things as the material an asset is made from (independent of its shape), surface color decoration, and small details such as wear, dirt, scratches, and grunge.

### **1.3.2 Art Style**

The art style will take full use of current technology, using next-gen approximate polygon counts (5-8 thousand for a character other props scaled accordingly) as well as taking full advantage of the variety of mapping techniques available, most notably diffuse, specular, opacity, and normal maps. In an effort to use these techniques to their fullest potential, the proposed art style could be described as worn. While not necessarily strict photo-realism, every object and model will be considered as the culmination of a story and include appropriate details to match, as opposed to producing perfect or pristine objects that look brand-new, shiny, and without imperfections.

While this style can apply equally well to a broad variety of genres, I feel it would apply particularly well to an alternate present, fantasy-inspired approach. That is, the level of technology and sophistication is assumed to be roughly equivalent to modern day, but instead of these feats being accomplished with science and technology, magic and supernatural technology fills this purpose. As such, the final appearance of many assets will seem to bridge or borrow between Science Fiction and Fantasy.

One advantage that this approach offers is the large wealth of material types and substances which could be used on a variety of models, allowing for a wide variety of surface properties to fully take advantage of all current mapping techniques. While generally more industrial than a final art style I would like to achieve, the concept artwork of Brom would be a good example of the proposed art style, with a close attention to detail, broad color palette, and variety of material types and substances.

### **1.3.3 Process**

To ensure that this project meets both my expectations of my best work and the requirement of having a series of models that work together and can stand separately as complete pieces, careful planning and concepting will be necessary. The first stage of the process will be to define the parameters of customization and extrapolate from this a set of modules and asset list to populate these modules. Once an asset list is finalized, each asset can be concepted, though not to a high degree of finish as the details themselves are a parameter of customization. Next, each model will be produced one by one, giving ample time to producing clean, efficient geometry. Finally, modules of textures will be

produced as a set, such that if two assets utilize the same material (iron or leather, for example) they will have a cohesive appearance. Producing the customizable parameters of these textures is also accomplished at this time, producing different looks of a similar material (in various states of wear and cleanliness) as well as producing decals, insignias, patterns, and other elements that are only visible as surface decoration and do not affect the profile or geometry of an asset.

### **1.3.4 Display**

The final output of this project will be created and displayed in 3DS Max for user interaction. Many of the options necessary for modular customization of geometry can be built using existing Max functionality such as point objects, constraints, and custom attributes. Material customization is also well built into 3DS Max with a combination of Composite, Blend, and Shellac materials, becoming even more user-friendly through the addition of a free 3DS Max plug-in called Node Joe. Finally, a concise, simple user-interface will be created using Custom Attributes, the Custom Attribute holder, and Max Script to give users a simple method to create a custom asset from the set of modules and materials created in this project.

### **1.3.5 Schedule**

Term 6

Week 1 (Oct 2): finalize proposal

Week 2 (Oct 9): Define parameters of vehicle customization; asset list; concept

Week 3 (Oct 16): Concept; Model base Chassis

Week 4 (Oct 23): Model large module assets  
Week 5 (Oct 30): Model small module assets  
Week 6 (Nov 6): Model details  
Week 7 (Nov 13): Texture base material sets  
Week 8 (Nov 20): Texture advanced materials (patterns, grunge, etc)  
Week 9 (Dec 27): Texture decals/overlays  
Week 10 (Dec 4): Texture advanced maps (normal/spec/etc)  
Week 11 (Dec 11): Final Renders

#### Term 7

Week 1: Define parameters of character customization; asset list; concept  
Week 2: Character model  
Week 3: Character texture  
Week 4: Model costume assets (33%)  
Week 5: Model costume assets (66%)  
Week 6: Model costume assets (100%)  
Week 7: Texture base material sets  
Week 8: Texture advanced materials (prints, grunge, etc)  
Week 9: Texture decals/overlays  
Week 10: Texture advanced maps (normal/ spec/ etc)  
Week 11: Final Renders

## **2. Methodology**

### **2.1 Concept**

I initially began work on this project by brainstorming a list of some possible modules that might be used on a vehicle. At this point I did not have a strong visual concept of what a potential final vehicle constructed from these modules might look like. Additionally, while I had a rough overarching visual style that I wanted to try to stay within, it was very sketchy and did not exclude very much.

From that stage I proceeded to produce a series of two-dimensional concepts of what some of these modules might look like. While I was brainstorming ideas I attempted to categorize them into groups such as method of transport, weaponry, seating, steering, and so forth. I worked on the concepts in similar groups, making sure various modules and pieces from the same group would fit together properly and work together as a whole. For example, one of the method-of-transport groups was wheels and within this group I designed several different styles of wheels as well as different types of suspension to attach the wheels to the chassis, making sure that each type of suspension would work with each type of wheel and the whole would be visually consistent.

During this concept phase I was primarily concerned with shape and form, producing only blocky, geometric forms of the different modules and paying little attention to detail, color, or surface. I also limited myself to a single chassis design which each of the modules was designed to attach to. Also, while I intended the modules to be combined in a variety of fashions, I conceptualized them all around the central chassis with

only a single method of attachment, intending to discover new methods as I advanced on to the next stage of the project.

I executed all of these concepts in Photoshop using a new layer for each module with the base chassis as the bottom-most layer. This allowed me at a very early stage to begin experimenting with different combinations of modules to see how they would affect the overall silhouette of the vehicle and also to determine which modules would fit together easily, which would require some additional design work to fit with other modules, and which modules would not work together at all. While it was very advantageous at this stage to begin to see some of the impact of specific design decisions, it was also limited by both my drawing skill in perspective and by the fact that these modules were only drawn from a single perspective.

## **2.2 Modeling**

From this early concepting phase I prepared to model out each module in 3d Studio Max 8. I decided to produce each model in a separate max file and use the x-reference tool in 3DS Max to bring all of the assets together in a single file to experiment with combinations and view the finished asset. The X-reference tool was particularly useful in that I could reference an object into the final scene before it was even finished and, provided I never changed the name of the asset, any changes I made in the module's specific scene file would be updated into any file I referenced the object into. This allowed me to see all of the modules together at a very early stage and to continue to see the effects of a change on a single model across the entire asset.

As I began modeling modules, I quickly ran into trouble while trying to determine a target polygon count to aim for while producing each module. At this point I still had not determined a target output level (what game engine or overall graphics level) my asset would be targeted for. Additionally, while I had a rough idea of what a single asset's target polygon count might be for a given engine, being that I was producing a modular asset, I did not have complete control over the polygon count as this would be subject to the final configuration and so, at best, I could limit the polygon count to a given range. While I decided that my target graphics level would be next-generation in general using the Unreal 3 engine as a specific target, I decided to err on the side of caution and keep my polygon counts for each module relatively low, falling in a range of 200-700 for any given asset, varying by complexity.

To ensure all of the modules fit together properly, I decided to model them as I had conceptualized them: from the inside out. I modeled the chassis first and saved it as its own scene file. I then modeled each asset that was intended to attach directly to the chassis inside the same scene file, directly on top of the chassis. Before I saved these additional scene files (with new names, so as not to over-write my original chassis model) I deleted the chassis models and moved the new model to the center of the scene file. This process continued out along the prescribed chains in such a way that I would begin modeling a given asset within the scene file of the module it was intended to attach to, delete the original asset, and save the scene file under a new name.

After I had finished modeling each module I created a new scene file and x-reference each of them into it. Once the entire asset was assembled and could be configured into all its possible variations (accomplished through placing each module in

its own layer and hiding all unnecessary layers) some of the problems of working from single-perspective concepts quickly became apparent. Most importantly, several assets which were intended to be used together collided with each other, meaning several models had to be tweaked to rectify this. Additionally, because of how the modules connected together, attaching some modules prohibited attaching other models, even though this was not originally intended, because the points at which they attached overlapped.

### **2.3 Unwrapping and Texturing**

With all of the geometry for the different assets produced and corrected for overlapping, I next began unwrapping each model in preparation for the texturing process. To speed up the process, my initial unwraps were not concerned with fitting the entire piece into the 0-1 texture space, but rather paying careful attention to where seams were falling, how much the space was being stretched to make the pieces uniform, and ensuring that all of the pieces had a similar grain. To clarify, several of the textures I was planning on producing had a specific grain, such as wood or brushed steel and so forth. I wanted to ensure that when these textures were applied to the models that this grain would be preserved in 3d space, so that different pieces from the unwrap were not oriented in such a way as to break the grain.

Once each of the modules was unwrapped I began producing my ‘clean’ textures (plan, simple materials without any details, grunge, and so forth). I produced six families of textures (metal, stone, wood, leather, cloth, and rubber) with several variations of each, and a consistent left-to-right grain among all of them. Because my unwraps did not fit

into the 0-1 texture space I produced these tiles so that they tiled in every direction. I combined these in 3DS Max into six base materials with a series of options available to choose from among the different varieties, and saved this out as a material library that I could then import into any other max scene file. I then imported the material library into the final scene so that I could see what the different assets looked like with a base texture applied to them.

As before, I quickly noticed some significant problems with this approach. While the models did not look bad, this approach was not achieving the look I was shooting for. Instead of looking like an actual vehicle, the model looked more like a miniature or toy. Part of this was the lack of detail, though I noticed a problem with my approach on this front as well. I had planned to produce individual detail and grunge maps for each model, any of which could be overlain on one of the base materials, thus making the textures as modular as the model itself. However, as I delved further into the Max material system I realized that while I could achieve this (albeit with some difficulty), there would be no way to preview what the resulting texture would look like without rendering the scene, which could take anywhere from a few seconds to several minutes, depending on the complexity of the final asset. While this would function it was certainly not desirable. Additionally, I realized some flaws with my unwraps. The most noticeable of these was that my texture space between the models was not uniform, and the same texture could appear large on one model and small on another. I also realized that regardless of any other changes I made, I would have to revisit my unwraps and fit them neatly into the 0-1 texture space if I was going to produce detail and grunge maps, as these would not be able to tile as the base textures had done. Doing so also meant I was probably going to

lose the consistent grain I had among all the textures, though moving into my next phase I realized this was less important than I had originally thought.

With all of the problems that had arisen from my chosen method of texturing, I decided to abandon the idea of a modular texture and instead focus on high-quality non-modular textures for each of the models, thus simplifying the process. I first revisited my unwraps and organized all of the pieces into the 0-1 texture space. Where possible I preserved the initial grain as I knew this would speed up the process of producing the final textures (I was planning on using my current set of textures as a base), though in some cases this was impossible if I wanted to use the space efficiently (which I did). After this I set about producing the final textures for each of the models.

This phase turned out to be the longest span in the project and was much more difficult than I had initially anticipated. I forewent the idea of producing alternate textures for each model and limited each to a single material type; whichever I found to be the most appropriate. By and large this turned out to be the metal material set though cloth, leather, and rubber were also used (stone and wood were entirely scrapped). My process changed as I continued producing textures which necessitated revisiting some of my earlier textures to bring them up to the same levels as my later textures, but eventually I used the same process on all of them. I gathered a large set of reference textures taken from relevant objects, mostly mechanical or vehicular, though there was a large variety, and adapted these to fit onto each texture, thus providing detail and interest where before there had been only flat space. I also applied a series of grunge maps both above and beneath these detail layers and tailored to bring out the highlight of the modules purpose and details. Once the base color map was complete I used it to produce a normal map

(within Photoshop using the NVIDIA normal map plug-in) and specular level and glossiness maps as well.

Having all of the assets in a single file proved invaluable at this stage as it let me see not only how the texture was progressing, but to ensure that the textures remained relatively uniform and appeared to belong together when they were placed on the final asset. However, I noticed some problems as I had before as I neared completion of this stage. Probably the biggest flaw I realized was that my concepts were not sufficient for the types of models I was trying to produce, which created far more work at this stage than may have been necessary. While I knew the function of each module from the concept, the actual shape was left very rough with little attention to detail and function (beyond the concerns of the modularity). As such instead of merely producing textures, I spent more than half of my time in this phase still designing as I determined what details to add to the model and where to add them. Also, while I could add details and embellish the models at this stage, I was limited to the shape I already had without revisiting the earlier modeling stage which I was reluctant to do as it would also necessitate re-unwrapping the model and throwing out any work I had already produced on the texture.

In order to achieve the kind of detail I was looking for, I also found it necessary to work at a relatively high resolution for each texture, commonly 1024 x 1024, though some were higher still. The problem of this was two-fold. First, the amount of disk space and memory this asset took up increased dramatically, something I noticed quickly when I tried to open the final max scene file. Secondly, not every piece required a texture of this resolution. However, this is something I couldn't ascertain from the texture or the model alone. To determine the proper resolution I would have to texture all of the

modules and compare the resolutions between them to determine which ones need that high of a resolution and which ones did not.

Once all of the textures were produced and set at their proper relative resolutions I realized I still had one significant flaw remaining in the asset. While the textures had been produced at a very high level, some of the initial models had not. While I was reluctant to altering any of the models a great deal (thus necessitating reworking the textures as well), some of the models were simply too square and blocky and could be improved upon without negating the texture. Once these assets were brought up to a satisfactory level, I was finally finished with the first pass at producing a modular asset.

### **3. Results and Analysis**

#### **3.1 Final Output**

The primary objective of this project was to produce a modular vehicle composed of a series of modules which can then be configured into a far greater number of final assets. I have modeled and textured a total of 22 different modules which can be classified into 7 different groupings. In all, there is an estimable 1232 possible configurations of these assets. However, because several of these configurations may be invalidated upon comparison to the optimal final asset (several configurations may have overlapping modules, or not satisfy all of the necessary requirements to qualify as a vehicle), the actual estimated number of valid configurations is roughly one-quarter of this, or 308 configurations.

Using these figures as a starting point, it is easy to mathematically calculate the benefit of using a modular approach to producing video game assets. In the time it has

taken me (a single developer) to produce 22 different actual assets, I have produced a potential of 308 assets, increasing my productivity by a factor of 14. Even though the figure of 308 is itself only a quarter of the total number of potential configurations, if we further consider that potentially only twenty percent of the valid configurations are truly different enough to be considered different assets for our purposes, the productivity of a single developer has still nearly tripled.

From this point of view, the validity of modular assets is clearly a huge success. Designing assets in a modular fashion has the capability to increase productivity by a bare minimum of 2000%.

### **3.2 Analysis**

Several aspects of this project have proven to be huge successes when compared with my original goals for the project. Chief among these, as previously discussed, is the proven validity of using a modular approach when designing assets for a video game, as mathematically the benefits are readily apparent. Additionally, the widely varied appearance and profile of one configuration to the next I consider to be a personal success, as many modular assets I have seen undertaken in the past have, in my opinion, failed to achieve the possible amount of variety inherent in their nature. Related to this is the large scope of the modular vehicle I undertook and succeeded at. Traditionally most modular assets have been very narrow in scope. In my instance a more traditional approach would have been to narrow the scope of possible configurations to a single type of vehicle, say a motorcycle or airplane. However, I have succeeded in making a modular asset which can be configured into a wide variety of types of vehicles, including cars (4-

wheeled configurations), bikes (2-wheeled configurations), trikes (3-wheeled configurations) and a wide variety of flying vehicles as well. Finally, all of the assets belong to a unified whole when considered in terms of their visual style. The large scope itself ran the risk of making individual modules look disparate when placed together in a configuration, leading to discord in the whole modular asset, yet this has been avoided nearly altogether.

Despite these major successes, there are several aspects of the project that I feel could have been improved upon. While the initial scope turned out to be many times too large for what could realistically be achieved in the given amount of time this is, to some degree, inevitable. However, even though the individual assets have maintained a unified visual style, the final style achieved is not the style I had originally set out to produce. Indeed, my initial research into the structure and appearance of vehicles fitting my visual parameters (the self-stated steam-punk appearance) proved to be quite inadequate. The effects of this lack of research have been felt throughout every stage of the project. With additional research up front, not only could a better visual style have been achieved, but time could have been spared in every phase. Initial brainstorming could be done away with altogether as these ideas would have arisen naturally out of the research process. The concepting phase would also have proceeded more quickly with a clear understanding of what every piece's function was and where it fit into the overall design. Additionally, modules could have been designed with closer attention to detail regarding the purpose, leading to a greater refinement of the overall design as well as saving a lot of time later on in the texturing process, as no design work would have remained at this late stage.

Regardless of any other impacts or metrics on the success or failure of this project, the single greatest achievement has been the knowledge gained about the processes inherent in the design and creation of a modular asset. The insights gained into the process of designing a modular asset, the concepts and ideas inherent to the idea and the greater understanding of how to approach this sort of asset production are far more valuable than the measurable output of the project itself.

## **4. Discussion/Conclusions**

While I was working on this project, I came to realize several fundamental concepts associated with the design and execution of a modular asset. While I came across these concepts while working on the actual structure and geometry of the asset, many of them have far-reaching applications and can easily be adapted to other types of modularity as well, such as modular textures and so forth.

### **4.1 Scope**

One of the most critical considerations of a modular asset is that of scope. Because a modular asset is in effect not a single asset but, instead, a group of assets, it is important to recognize and limit the size and scope of that group early on to better understand the underlying structure of the asset and plan accordingly. The underlying structure of how modules fit together is very important in producing a well-made modular asset, and so the smaller the group of assets a single modular asset is intended to imitate, the easier it will be to find a common underlying structure among all the members of the group.

As an example (and one very relevant to video games), many video games have vast numbers of weapons that a character may own and use, and these weapons come in a wide varieties of types and styles. Converting any or all of these weapons into modular assets could greatly reduce the workload will simultaneously increasing the breadth of style and number of weapons available. However, the complexity of these modular assets is largely dependent on exactly how many weapons (or what types of weapons) it is intended to replace. If there is to be a single modular asset that is intended to be used for every single possible weapon, not only will this asset have an extremely large number of modules in order to fulfill its role, but the underlying structure quickly becomes increasingly complicated. Disparate weapon types (such as swords, pole arms, and long bows in a medieval setting) have very little in common structural, save for the basics such as a grip and a business end. However, if we instead begin breaking this single asset into multiple modular assets with little to no overlap, the complexity drops rapidly. Using the earlier example of medieval-style weaponry, if we instead choose to group these weapons by relative size and function, creating a group of bladed weapons (daggers, axes, swords and so on), pole arms, bows, and so forth, there are more overall assets to make, but each asset becomes simpler to plan and execute as each of these groups have more commonality amongst their requisite members.

Narrowing down even further, say creating a modular asset group for swords exclusively, reduces the complexity to nearly a bare minimum, but in turn each configuration has more in common than not. Determining an appropriate level of scope is critical not only to setting aside an appropriate amount of time in order to execute the asset, but also in determining how great a variety is necessary from the asset in the end.

In the case of MMORPG, with the high number of end users, making a series of modular weapons for each possible weapon type could indeed be beneficial, as the game overall could definitely benefit from the near-limitless variety that provides, whereas single-player games would likely benefit from having only a single modular weapon, or perhaps several modular weapons for large, related groups of weapons as variety is still needed, but not in the same amount as an MMORPG.

As this applies to my specific modular asset, the scope itself remained fluid throughout much of the production and overall was very large. The only limiting factor was that the asset was indeed a vehicle and as such must be capable of transporting a single person. This had the distinct advantage in that any module I could think of that did not contradict this simple purpose was a potentially valid module. However, on the downside, maintaining visual continuity and ensuring all of the disparate parts could fit and function together (as much as was possible) proved very difficult.

## **4.2 Design Methodology**

As discussed earlier, when I first began the actual design process of the modular vehicle, I produced a list of potential parts whose only commonality was that they could all be components of some type of vehicle. In order to provide some cohesiveness to this mass, when I began designing the physical appearance of the components, I started with a sketch of a simple chassis onto which most of the pieces could be attached and then worked my way out to the additional pieces. Even as I began producing the geometry for these different components, I kept the same work methodology of producing the innermost pieces first, and then working my way out from there to the outermost sides of the vehicle.

When I initially began working on this project, I used this method (which I later referred to as Inside-Out Design) because it seemed the most appropriate method for the scope and breadth of variety I was trying to achieve. As I've looked at the design of other modular assets, especially those with a much more narrow scope than what I have developed, I have realized that another equally valid method is the exact inverse of this, or Outside-In Design.

Using Outside-In Design, instead of beginning with a rough concept of the final asset group (vehicle), producing a list of potential parts, or even concepting the innermost element of the asset (in this case, the chassis), one would begin with a whole and complete asset that is indicative of the final asset group, as much as that may be possible. Using my modular vehicle as an example, if I had used Outside-In Design I would have begun by concepting out a finished vehicle that could be constructed with my final modular asset group, such as a car, or a jet configuration. Then, using this as a base for the structure, separable parts are identified and quantified as individual modules which can then be replaced by other modules. Using a car as a base, hood, windshield, suspension, seating, and wheels could all be identified as separate modules. Some, such as wheels, could even be further divided into separate modules for rims and tires, though depending on the desired complexity this may be unnecessary.

There are advantages and disadvantages apparent to using either method exclusively. From having used the Inside-Out method exclusively on this project, I can safely say that the single largest advantage of this method is the ease of creating an extremely wide variety of modules, thus contributing greatly to a more varied final asset group. However, a major drawback of this method is that there is no immediately

apparent structure to the asset. If there is only a single level of modularity (discussed later), this is not an imposing problem, but the further out the modular asset extends from its inner-most module, structure quickly becomes complicated and difficult to organize. As a result, it often happens that specific modules can only be used in very specific configuration, thus limiting the overall benefit of using a modular asset approach in the first place.

While my analysis of the benefits and drawbacks of using the Outside-In method are partially conjectural as I have not personally produced an asset using this method, many are self-evident from simply understand the approach, especially as contrasted with the Inside-Out method. The single largest advantage of the Outside-In method is that it provides immediate structure, so that when any module is conceived or created, its position in the overall asset and means of attachment are immediately apparent. Likewise, if this method is used in the actual execution of the asset as well, as soon as the initial group of modules is completed you can view a single, finished configuration in its entirety. It is also much easier to maintain visual continuity among all of the different modules. However, as a drawback, because of the greater amount of structure, it is more difficult to generate the wide variety of assets that the Inside-Out method can produce.

In the end, neither method is necessarily universally better or worse for designing and producing a modular asset, but rather their benefits and drawbacks must be weighed against the needs and concerns of the specific modular asset in question. In many cases it may actually be advantageous to use both methods; first one and then the other. From my experience on this project, I have come to conclude that the most powerful and flexible approach to designing a modular asset arises from beginning with the Outside-In method

to generate an initial structure and set a benchmark for visual continuity, and then to proceed to use the Inside-Out method to generate a wide variety of assets that can attach to the modules identified in the first method. This approach can alleviate many of the drawbacks of both methods, though not eliminate them entirely, and is a very robust solution to produce a modular asset with a strong underlying structure allowing it to replace very large groups of related assets while still having the necessary variety of modules.

### **4.3 Structure and Modular Mapping**

As mentioned previously, structure plays a critical role in designing and creating a modular asset. To clarify, when I refer to structure, I am speaking specifically of how separate modules to connect to one another: the order they attach to each other in, how many modules can attach to a single other module (or in reverse, how many different other modules can a single module attach to), which modules overlap or prohibit each other (if module A is used, module B cannot be used) and so forth. Understanding this system is crucial, although it can be very difficult to understand all at the same time. To this end, I have formulated a series of concepts that have better allowed me to understand and visualize the structure of a modular asset. All of these different concepts culminate in a technique called modular mapping which I will discuss later.

#### **4.3.1 Points of Contact**

The first concept in understanding modular mapping seems a very simple concept, but is very important in establishing standardization across a structure. This concept is that of Points of Contact. Put simply, a point of contact is the terminator point where one module ends and another begins. However, recognizing these points and understanding

how they work is very important to creating a modular asset. In some cases, these points of contact may not be explicit, but rather an expanse of space large and flat enough to fit something else on. In the case of my project, many of the points of contact follow this method. That is, once identified, most points of contact are simply large, flat areas that another module may be placed on top of (or next to, depending on orientation). While this is the simplest form of points of contact, it is also the least uniform and can lead to many problems of assets overlapping one another. That is, two given modules may use the same Point of Contact on a third given module, and while neither takes up all of the space, both take up enough of the space that they cannot be used simultaneously. Also, sometimes assets do not attach on specified planes, but rather on edges and corners. Many of the suspension solutions of my modular vehicle completely encompass a section of the chassis, taking up slices of each side as well as the edges. Because most of the Points of Contact are not clearly defined or even uniform across the different modules, it can be very difficult to visualize or quantify which modules may be attached simultaneously and which may not. However, this ad hoc approach of Points of Contact does offer more flexibility in structure, which can in turn lead to a greater variety of possible shapes and configurations.

Alternatively, Points of Contact can be specified from the beginning and be standardized across portions of an asset or even across the entire asset itself. A very good example of this in the modular vehicle is the Point of Contact between the wheels and the suspension. Any point at which a wheel can be attached is extremely uniform, being of the same shape (circle) and size across any module which can receive a wheel. Likewise, the point on the wheels at which they are meant to connect to other modules is likewise

uniform, being almost precisely the same shape and size. This offers the advantage of making overlap explicitly clear in the design. A given piece of suspension can have other modules attached to it, but only if those other modules are wheels. Likewise, if one wheel is being used on a piece of suspension, another wheel may not be used on the same piece as they occupy the exact same Point of Contact. This approach greatly clarifies confusion in the structure, but also becomes more rigid and less flexible, leading to a potential decrease in overall possible variety.

Also, identifying Points of Contact allows related modules to be grouped together. Modules which all share a common Point of Contact and are mutually exclusive (if one of these modules is used, any of the others may not be) can be thought of as a Module Group. The notion of a Module Group helps to organize and clarify structure and design, especially when it comes time to produce a Modular Map.

### **4.3.2 Modular Flow**

Related to the idea of understanding precisely where two modules attach to each and how is the notion of understanding what order they connect in. To clarify, in every modular asset there is always one inner-most, core module. This module itself does not attach to any other module, but rather one or more modules may attach to it. At first glance this concept may seem purely semantic in nature, but if the structure of a modular asset gets to be sufficiently complex, this becomes crucial to establish.

Once the Modular Flow of a given asset is known when can then break down the generic Points of Contact into two separate classes: Inputs and Outputs. Inputs are receiving points of contact in that they allow other modules to attach to the module they belong to. In the case of the wheel/suspension example, the Point of Contact on the

suspension would be an Input. Similarly, Outputs are exactly the opposite; they allow the module they are on to be attached to other modules. Remaining with the wheel/suspension example, the Point of Contact on the wheel would be an Output. These terms may also seem arbitrary depending on your vantage point of the asset, but once we proceed to producing a complete Modular Map, they immediately become self-explanatory.

### **4.3.3 Types of Modules**

After all of the Points of Contact have been identified and categorized as either Inputs or Outputs, it is then possible to begin categorizing modules themselves based on these characteristics. In examining the underlying structure of my own modular asset as well as examining several other assets, I have identified five basic types of modules, though not every modular asset will have all five present in its design. These basic types are Root, Sub-Root, Link, Span, and Terminator.

#### **4.3.3.1 Root**

A Root module is one whose Points of Contact are composed solely of Inputs. Root modules always form the core of a modular asset and are the inner-most, core modules previously referred to. They never attach to any other modules but often have several Points of Contact allowing for many different modules to be attached to them. The chassis is the Root module of the modular vehicle I designed.

#### **4.3.3.2 Sub-Root**

A Sub-Root module is a module with a single Output, but more than one Input. In other words, this module can attach to another module and can have more than one module attached to it. When I later discuss Modular Mapping, these modules will be

points on the map past the root at which the Modular Flow forks outwards. An example of this in the modular vehicle would be that of the Front Mount which only connects to the chassis, but has two Input Points of Contact where wheels or weapons may be attached.

#### 4.3.3.3 Link

A Link module is one which has only a single Input and a Single Output. These modules can connect to one other module, and likewise can have only a single module attached to them. Despite this strict definition, these modules in no way limit the possible variety of an asset, as there can be more than one Link for a given Input on a module, and there can be multiple modules which can attach to its single Input. An example of a Link module would be that of the Rear Mount, which can only connect to the chassis and only has one Point of Contact (which may be used by either the Ballista or Blimp modules).

#### 4.3.3.4 Span

Span modules are unique in that they are the only type of module which has multiple Outputs. That is, Span modules connect to more than one other module simultaneously. These modules can be quite visually interesting, but run the risk of reducing overall variety as they require two different modules to be in place before they can be used, limiting the number of possible configurations. If a Span module has no Inputs this is not a great risk, but if it does then the usage of any module which extends from the Span is dependent upon both (or all) of the requisite modules that the Span connects to.

Span modules can also overlap with other types of modules. That is, there can be a Span Sub-Root module that has multiple Inputs and multiple Outputs. There can also be a Span Terminator that has multiple Outputs and no Inputs.

#### 4.3.3.5 Terminator

Terminator modules are modules which designate the end of a chain or branch on a Modular Map; they have no Inputs whatsoever. Once a Terminator module is attached to the configuration, no more modules may be connected onto it, though other modules may still be attached on open Inputs on other modules. Terminator modules are not strictly necessary in design as any Input can simply be left open in a given configuration, but they are easier to design as there are few Points of Contact to worry about accounting for.

#### 4.3.4 Level of Modularity

The concept of Level of Modularity is one that applies to an entire modular asset. As with many of the other concepts, this one is easier to visualize in a Modular Map, but in short it signifies the possible length of a chain of modules, or a range thereof. That is, the number of how many modules can be attached one to the other in a continuous chain. This is commonly best expressed as a range of minimum and maximum values, where the minimum value designates how many modules must be strung together for a given configuration to be considered complete and the maximum designates how many modules it is possible to string together (either before requiring the use of a Terminator module, overlap between placed modules, or any other reason).

As an example, a modular character asset which consisted of a root asset (comprised of the physical body of the character) and a series of Terminator modules

(different articles of clothing or armor) would have 1 Level of Modularity, as the Root is always excluded in determining this number. The maximum length of any chain of modules which can be attached to this Root module is 1, because every possible module which can attach to it is a Terminator. If instead one of these modules was a Link instead, the Level of Modularity would be a range of 1-2, assuming that the Link could not be repeated and attach to itself multiple times.

Using the modular vehicle as an additional example, this asset's Level of Modularity is a range of 1-3. This asset requires at least one asset to be attached to the Root for a configuration to be considered valid, and no chain that can be attached to the root can ever extend beyond 3 modules.

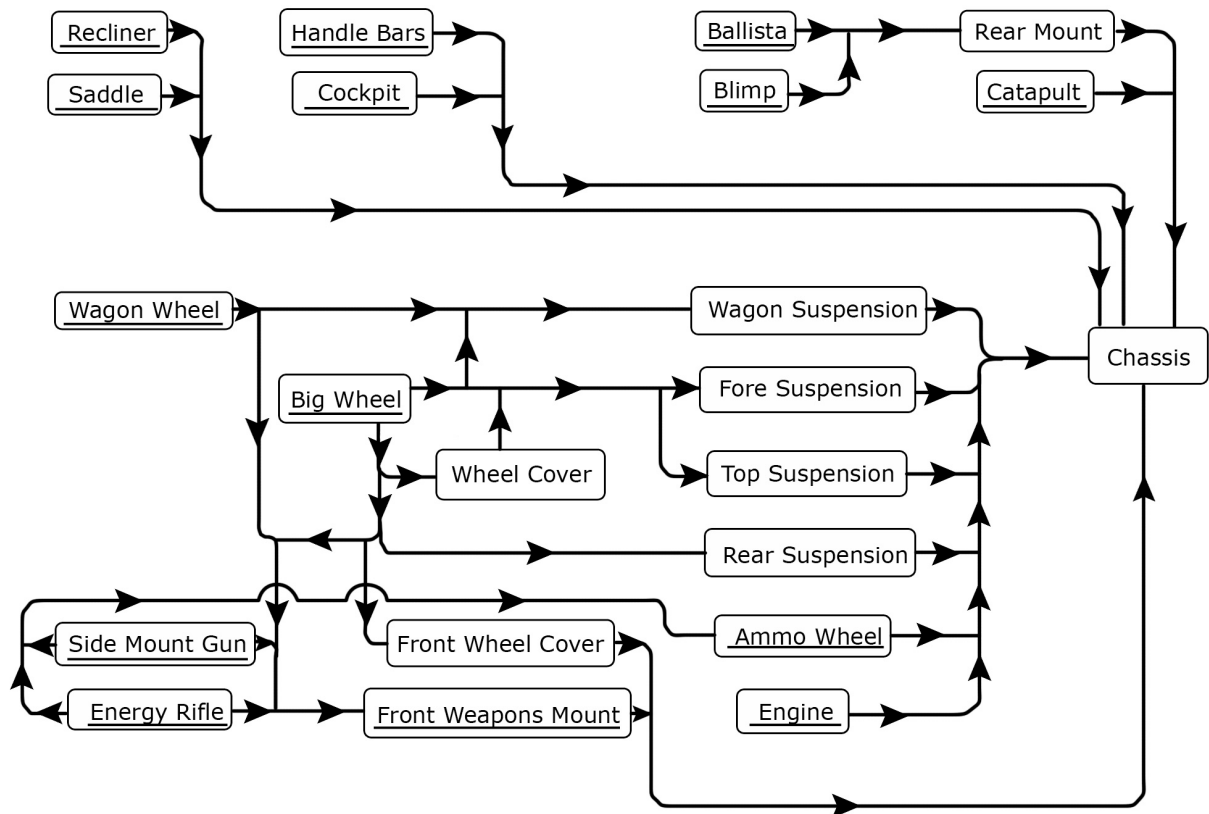
#### **4.3.5 Modular Mapping**

Modular Mapping is a means of visualizing the structure of a Modular asset; a way to abstractly represent how all of the pieces do or can fit together.

To create a Modular Map of an asset, it is best to work from the inside out and from the side of a page (or area) to another to help organize all of the information it will contain. The syntax of a Modular Map is not strict, but must relate all of the following: each Module, Points of Contact on each module, Modular Flow (by designating Inputs and Outputs on established Points of Contact), and Overlap between modules (though not necessarily at the same Point of Contact) if possible. The syntax I have chosen to use to represent each of these is as follows:

Modules are represented by a box, and each box is labeled with the name of the module. Points of Contact are represented by thick lines drawn between and connecting these boxes. A directional arrow in the center of these thick lines denotes directionality;

the box it is pointing towards is designated as the Input while the one it is coming from is established as the Output. Modules which share a common Point of Contact will merge these lines before they meet, denoting a single input on the receiving modules. This method allows the number of Inputs and Outputs to be calculated quickly and easily by simply counting the number of lines coming to or leaving a module. Finally, overlap between modules (modules which can not be used in the same configuration) is designated by a thin line drawn between the two boxes with a small 'x' in the center of this line. Sometimes physical constraints, such as room on the page or pressures of organization, make it impossible to follow these syntax rules exactly, but the closer they are followed, the more easily the final Modular Map may be read.



This is a modular map of the modular vehicle I have produced for this project. While I have not represented Overlap in this map (as it overly clutters an already relatively difficult-to-read diagram), I have denoted Terminator modules. In this instance, even some modules which have Inputs may function without them, allowing them to function alternately as Terminators or Links, depending on the user's whim. As such, any module with a line beneath its name may function as a Terminator. Additionally, in some cases the syntax is incorrect and was altered for the sake of readability. This is particularly evident in the case of the weapons group (Side Mount Gun and Energy Rifle). Each of these assets shows two separate Outputs whereas in fact there is only one.

This map also provides an excellent graphic example to clarify other concepts previously discussed. The Levels of Modularity are very easy to extrapolate from this map simply by counting backwards from the Root module (Chassis). Level 1 can be reached in a configuration consisting of the Engine, Saddle, and Cockpit. Level 3 can be reached on a single chain consisting of Fore Suspension, Wheel Cover, and Wheel. This also clearly demonstrates the utility in identifying common Points of Contact and the resulting Module Groups.

In this instance it is easy to identify module groups by where they are placed on the module map, aiding both comprehension and readability. Modules have been organized by both which Level of Modularity they belong to, with higher levels located farther away from the Root module, and by Module Group. As such, there are a total of 5 easily identified Level 1 Module Groups, each of which connect directly to the Chassis: the Seating group (Recliner and Saddle), the Steering group (Handle Bars and Cockpit), the Top group (Catapult and Rear Mount), the Locomotion group (all of the suspension

modules plus Ammo Wheel and Engine), and the Front group (Front Wheel Cover and Front Weapons Mount). Additionally, there are three other Module Groups, though they become somewhat more difficult to identify and do not all occupy the same Level of Modularity. These are the Weapons group (Side Mount Gun and Energy Rifle), the Wheel group (Wagon Wheel, Big Wheel, and Wheel Cover), and the Rear Mount group (Ballista and Blimp). Both the Weapons and Rear Mount groups are easy to identify and occupy the second Level of Modularity as they can only connect to Level one modules, but the Wheel group is a special case. Both the Big Wheel and Wagon Wheel can connect to Level one modules from either the Locomotion or Front groups, but are not wholly mutually exclusive as the Wagon Wheel can connect to some modules which the Big Wheel cannot and vice versa. The situation is even further complicated with the inclusion of the Wheel Cover. The Wheel cover can only connect to modules which the Big Wheel can also connect to, but cannot connect to all of them, and the Big Wheel can then in turn connect to the Wheel Cover. This means that the Wheel Cover is entirely a Level two module whereas the Big Wheel can alternately be a Level two or three module in terms of the Levels of Modularity. While this complexity is reflected in the Modular Map it is also the single most complicated portion of the map.

#### **4.4 Granularity**

A final concept that is helpful when designing a modular asset or refining a pre-existing modular asset design is that of Granularity. This concept is most applicable when using the Outside-In method of asset design as it directly concerns the placement of Points of Contact and where to separate single modules into multiple modules or how to combine multiple modules into a single module. In short, Granularity is a comparison of

the relative complexity of different modules. This is particularly important in relation to the impact any given module will have on the appearance and profile of a final configuration.

Using the modular vehicle as an example, there is a high variance in Granularity from one asset to another. For instance, the weapons group (Side Mount Gun and Energy Rifle) are each highly complex models with high polygon counts and very unique, complex profiles. By contrast, the Chassis is a very simple module with a plain, blocky profile and very low polygon count. There are also modules which fall between these two extremes, such as any of the suspension modules which are each more complex than the Chassis module yet simpler than the weapons modules. While this separation in the complexity of modules is not necessarily undesirable, it is something to be aware of. If all of the component modules are near the same Granularity it is easier to predict the impact a single module will have on the overall appearance and profile of a final configuration.

When determining appropriate levels of Granularity, there is one factor to consider: If there is any benefit to the modular side of things to be gained by breaking a single module into several modules. That is, will there be alternate modules that can be substituted for either (or any) of the resulting component modules, or if any of those component modules can be used elsewhere in place of existing modules. If neither of these is true, there is little to be gained in dividing the module up. Likewise, if there are two existing modules which are only used in conjunction with each other and no other modules, there is little reason to have them as separate modules. Finally, if a single module is much more complex than other modules in the asset or even other modules of

the same Level of Modularity, it may also be advisable or even necessary to simplify the design of that module to help unify the modular asset as a whole.

## **5. References**

---

- <sup>i</sup> Star Wars Galaxies. Retrieved from <http://starwarsgalaxies.station.sony.com/>
- <sup>ii</sup> Anarchy Online. Retrieved from <http://www.anarchyonline.com/>
- <sup>iii</sup> Eve Online. Retrieved from <http://www.eve-online.com/>
- <sup>iv</sup> The Matrix Online. Retrieved from <http://thematrixonline.station.sony.com/>
- <sup>v</sup> City of Heroes. Retrieved from <http://www.cityofheroes.com/>
- <sup>vi</sup> City of Villains. Retrieved from <http://www.cityofvillains.com/>
- <sup>vii</sup> Second Life. Retrieved from <http://www.secondlife.com/>
- <sup>viii</sup> Guild Wars. Retrieved from <http://www.guildwars.com/>
- <sup>ix</sup> Lineage 2. Retrieved from <http://www.lineage2.com/>
- <sup>x</sup> Rappelz. Retrieved from <http://rappelz.gpotato.com/>
- <sup>xi</sup> World of Warcraft. Retrieved from <http://www.worldofwarcraft.com/index.xml>
- <sup>xii</sup> Maple Story. Retrieved from <http://www.maplestory.com/>
- <sup>xiii</sup> Ragnarok Online. Retrieved from <http://iro.ragnarokonline.com/>
- <sup>xiv</sup> Puzzle Pirates. Retrieved from <http://www.puzzlepirates.com/>
- <sup>xv</sup> Disney's Toontown Online. Retrieved from <http://play.toontown.com/webHome.php?r=838673&r=172401&r=30757&r=865698>
- <sup>xvi</sup> Westward Journey. Retrieved from [http://corp.163.com/eng/games/westward\\_journey.html](http://corp.163.com/eng/games/westward_journey.html)
- <sup>xvii</sup> Lord of the Rings Online: Shadows of Angmar. Retrieved from <http://www.lotro.com/>
- <sup>xviii</sup> Warhammer Online. Retrieved from <http://www.warhammeronline.com/english/home/index.php>