The Content Problem

Vincent Agerbech, Allan Johansen & Brian Johansen

Aalborg University Copenhagen
Group 806, Semester 8, Medialogy
Semester theme: Virtual Worlds

Supervisors: Louis Emilio Bruni & Salvatore Livatino

Spring 2005

Abstract

In this project we will try to uncover and describe the content problem in games. Initially we set out to create a virtual world in the form of a one level game. This project is about why we did not succeed. Some say that if you don't know where you're going, you can't get lost. This was our problem in a nutshell. We left with out a roadmap to lead us home safe. But luckily for the next traveler we mapped the journey along the way. This map comes in the form of an assets pipeline description, illustrated as an easy to use chart. This assets pipeline serves as the core product of our project, where as the game prototype has served as our testing ground for the experimentations we have had along the way.

Additionally we have tried to identify way points in the different areas of the creation of a virtual world.

We hope this can be of help for future works in this field allowing the next travelers to steer free of the obstacles we never saw coming. Along this path we will cover interesting aspects of game production and design considerations. An understanding of these aspects has given us an insight of the huge process from idea to realization.

Content

Chapter 1

Introduction

One developer had said that; “Making games used to about fun, but now it is all about blood sweat and tears”. Game making has gone from an enterprise driven by enthusiasts to a true mass-market industry. During this project we surly bled sweated and wept. This paper attends to investigate the mechanisms behind that development and find a path bag to the fun and creative process of game making.

We have chosen a truly problem based approach, looking at the situation form the eye of the tornado. We have discovered that creating a virtual world involves an order of operations. The problem we encountered when creating a virtual world was concerned with the creation of the actual content constituting that very virtual world. This amplifies the importance of finding a solution to this problem. We felt that dealing with problems inside a world we could not create seemed useless. We were face with a decision to either accept defeat or get back up and fight. We chose the latter. We were determined to see if we could find a solution to this problem.

In order to illuminate the dark shadows of this content problem we have tried to attack it from most possible angels. An in depth understanding of a problem is the key to finding the solution. To unveil the context of the problem we have tried to first analyze it as a result of a general situation. We have then turned to see it in relation to our own project. This is the motivation behind the structure of the paper. We go from a general perspective to a subjective and lastly we give some suggestions to what might lie ahead.

We have focused on this problem in an attempt to come up with a possible solution. We hope that other projects in this field can benefit from the lessons we have learned. With this in mind we have paid special attention to how these solutions are presented. We felt that it was important to present them in the most accessible way.

We have favored novelty over trivia. We have not seen it beneficial to re-describe textbook content already described in depth else where. We have focused on the problem and our findings and chosen to only deal with theoretical aspects related to that problem. This reflects what we understand with problem based learning.

The theme under which this paper is written is virtual worlds. We will used this term to also include virtual worlds in games. We will use the terms virtual world and games as meaning the same. This will naturally exclude games that are not developed around a virtual world such as puzzle games. This is coherent with definition of virtual worlds and the consensus of other works in the field.

In the game industry yesterdays news are old news. It is an extremely dynamic and fast moving industry. In our research we have taken this into account by only looking at the latest information. The main source of information has been conference proceedings. We have also applied a *learn from the best* approach. We have tried to identify the biggest authorities in the field, and have mainly relied on there work.

Chapter 2

The Genesis of a Virtual World

A virtual world is a computer generated reality. A virtual world is by a large part defined by the process used to create it. In the same sense as the real world is understood and interpreted as the result of the evolutionary process producing is. Following that analogy we could say that the creation theory governing most virtual world generation is still pre-Darwin. The common approach to virtual world generation resembles the bibles description of the Genesis. God created the world by modeling and adding content. He set up the rules and laws to govern his creations. This is very similar to how we create a virtual world by producing and adding 3D content too a virtual universe. The virtual world designer will set up the rules and constraints which will affect the added 3D content. As we defined a virtual world as a computer generated reality we will have to also consider the computer generating it.

2.1 Computer games and technology

Computer games have always walked hand in hand with the technology available. When computers and electronics were invented researches were stunned by the “huge” processing power and could see the potential for these machines. In 1958 the chip was invented, which could calculate a staggering amount of functions and took computer technology to a whole new level.

"What we didn't realize then was that the integrated circuit would reduce the cost of electronic functions by a factor of a million to one, nothing had ever done that for anything before" - Jack Kilby

Through the 1960's the technology began to evolve and researchers began experimenting. This resulted in some very simple games. However in 1962 the first interactive computer game (Spacewar) saw the light of day. It was created by Steve Russell at MIT. This remarkable breakthrough created an interest in programming and a spawn of clones appeared, derived from Steve Russell's idea. It was first in 1972, that Nolan Bushell (who later became co-founder of ATARI) created PONG (Ping Pong in modern terms), which again set a standard of what could be archived with the technology at hand.
The games up till 1977 were integrated with the hardware. This meaning you bought one unit, which had one game build in, but this was about to change. ATARI released in 1977 the first successful consol, which made it possible to replace games using tapes. In the early 1980's there evolved many consoles and arcade games, some more successful than others. Because of this the market of video games crashed in 1983, because there was too many who wanted a piece of the cake, resulting in poor games. However the industry would rise again and in 1985 Nintendo released NES (Nintendo Entertainment System) with 8 bit graphics, which gave new life to the game industry. They set the standard for game design to business planning and they were the first to allow third party developers to manufacture games for their consol. There games with Super Mario Bros. were received well, because of the innovative gameplay. They became and still are a synonym with Nintendo. Nintendo would later on develop an even better consol (SNES, 1991), but it failed, because they couldn't produce new innovative titles despite there success.
Through the 1990's there evolved several impressive consoles, but many never really made it onto the market.
Then it happen, the 32 bit era arrived with Sony's Playstation (PSX) in 1995. The consol backed with a huge campaign took gamers with a storm. The consol

had amazing graphics and there was a huge variety of titles to choose from. The possibilities seemed endless. In 2000 they released the Playstation 2 (PS2), which had improved technology and gave game developers more resources to generate stunning graphics. In the wake of PS2, Microsoft developed the Xbox. The reason for this was because they wanted to compete with Sony, and could see the huge potential on the market.

Today the industry of consoles is mostly divided between three companies, Nintendo, Sony and Microsoft, who hold a big share of the market.

Despite we've only covered the consoles; there has also been a huge leap in the technology of personal computers (PC). In the 1970-80's the PC allowed users to program their own games and the distribution became easier with floppy disks and other storage media.

The technology is now beginning to feel the pressure of modern games and their content heavy games. This has evolved in that hardware developers are starting to create separate cards for your computer, much like graphic cards, but instead to handle physics. (See Chapter 4)

The evolution of computer performance has been explosive. The computers of today can now render millions of polygons and the most amazing visual effects. The question is how much the game developers gain from adding a lot of polygons, animations and dynamics. Will Wright the creator of The Sims explains that his game characters had over 22,000 separate animations, which all were done by hand. This meant that he had an army of animators working on the animations, but does this make the game twice as good if it only had 11,000? This his illustrates why the budget for creating a big game easily exceeds \$20.000.000.

Along side the development of computer power has been the development of storage media.

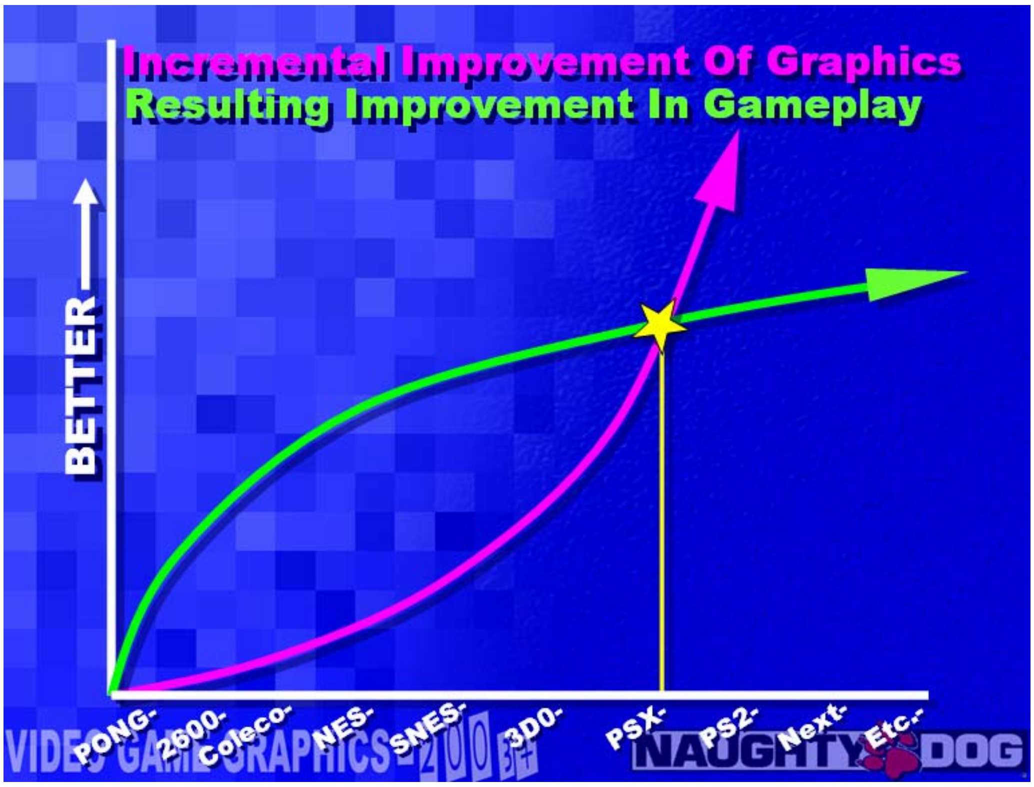
Storage media technology has gone from small hard drives and floppy disks to CD-ROMs / DVD and giant hard drives of several hundred gigabytes. Technology has provided game developers with more resources; games have gone from storage spaces, where a floppy disk of 1.44 MB could contain a whole game to several CDs and GB of hard disk The data size of games is going through the roof, because they have large amounts of storage space for content e.g. CDs. Take for example Half Life 2, where a demo takes op nearly 800 MB disk space and the full game occupies 4.5 GB.

More storage space means more content. More content means more work. But the people to create this content can not just install a faster graphics card. The artwork modeling and texturing are still man made work. The work has gotten harder as the models are more detailed as well as the texturing. The game industry has felt the impact of this mechanism. Games can no longer be sold on better graphics as we have reached a level of detail where the extra graphics improvements does not improve the game play of the games. The distinguishing factor lies in the attention to artistic detail.

"A race to distinguish games based on graphics could start to resemble an expensive arms race."

DFC Intelligence, 2005

Game developer and Co-founder of Naughty Dog Jason Rubin have suggested this interesting graph (next page), which illustrates how the advancement in technology has caused gameplay to drop.



To sum up the current situation we could say that we are closing in on a state where computer possibility is exceeding human capability. In his talk at the 2004 Game Developers Conference Will Wrights made a projection on the budget and team size in game development. On his project The Sims he had a team of around 25 people. He went on to estimate that we will reach a point during this century where a game will need a team of around 2.5 million developers and have a budget of around 500 billions dollars. At the Game Developers Conference 2005 he felt that the issue of content and content generation was one of the key problems among game developers.

So what do we do? It seems that there is no one simple answer to this problem. In the following we will try to outline the different solutions.

2.2 From genesis to Darwin

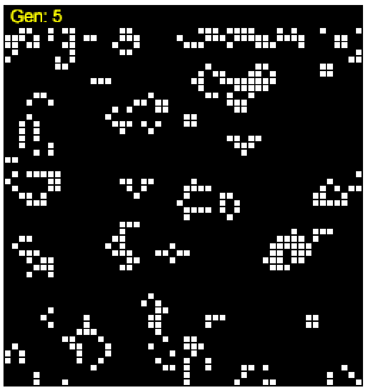
The evolutionary theory of Charles Darwin as described in his master piece; the origin of the species, stated that it was not God to be held responsible for the creation of our world but nature itself. Nature is not controlled by an almighty force with a final blueprint of its creation. Nature in modern evolutionary theory is controlled by propagation and emergence of successful survival strategies. An idea, which turns the focus from content to process. Content can be view as a result of a process. In order to identify such processes the attention turned to nature itself.

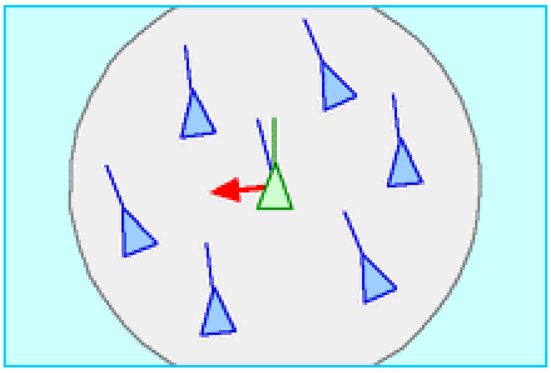
This leads us to the procedural content creation solution as one of the solutions to overcome the high demands on content creation. A solution based on getting computers to generate the content automatically. To understand this strategy we have to first take a look at the evolution of computer science to see how these processes have emerged.

At the turn of the century science was caught between two not unified theories of physics. We had the classical theory including Newton's mechanics and Einstein's relativity theories on one side and the quantum mechanics on the other side. The classical theories were good for describing large systems like planets while the quantum mechanic could explain sub-atomic phenomena. But none of these could fully explain the reality around us, however what they could explain and help developed was the first nuclear weapon. This project called The Manhattan Project was also the first example of large scale deployment of computer simulation. The birth of computer simulation has enabled us to deal with natural phenomena that were previously out of reach.

This brings us back to virtual world creation where computer simulation can be used to implement self propagating and emerging dynamic systems to our virtual

worlds. Ironically the development of the most real world destructive device became the savoir of virtual world generation.
Two examples of implementations of such ideas can be seen in flocking algorithms or in the Game of life by John Conway.





Creating more procedural content can lighten the human resources for creating games and instead shift it to creating new great innovative games. Simulations of content creation are growing steadily with the use of procedural techniques and physics. Even though procedural techniques have been used in games, it importance has been dismissed until now. They give room for a lot of content using minimal memory and computational resources.
With procedural techniques the game developers can through algorithms describe many processes, which are usually done by separate content creators. Terms like procedural texturing, procedural animations are already well known. Now game developers are looking at procedural content with even more interest, because of the bottleneck they are becoming stuck in. This step is going to revolutionize the way games are developed, but it's also opening new doors and ideas for more innovative games. In 1999 Populous was released and here procedural techniques were used. This created a new and exciting game, which gave a whole new perspective to strategy games. The creator Peter Molyneux already an established game developer went on to make Black & White (2001).

Black & White made it possible for players to influence the character through how they played. This was an interesting aspect and had already been seen to some degree in The Sims (2000). The essence in The Sims was to lead a normal life; buy a house, get a job, buy the interior and as the game progressed you could change your life style if you had a good economy. They game include interaction with NPC's and a family could be established. In multiplayer the player could interact with friends, which gave a new dimension to the game. You lived in some sense a virtual life in a virtual world.
Utilizing such techniques allows us to produce much more complex systems with a considerably less amount of data.

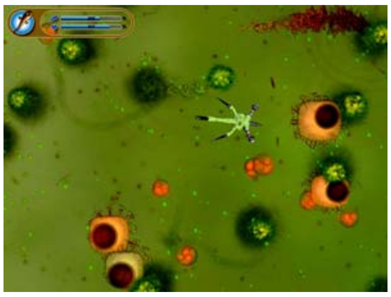
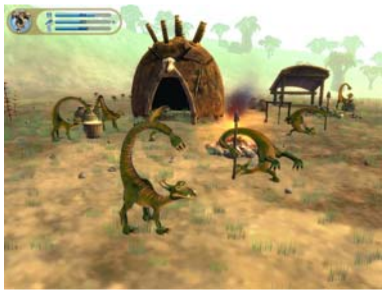
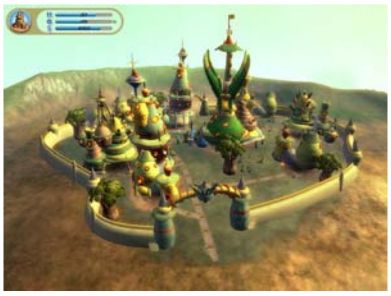
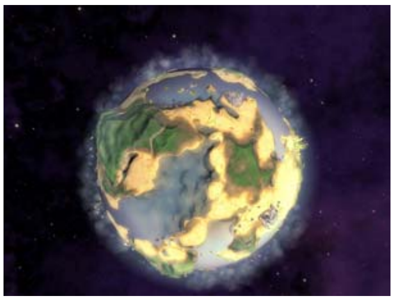
Will Wright learned and realized through his creation of SimCity and Sims, that people like content and especially if they can customize it. The customization of content helped people to immerse themselves, giving them a sense of ownership. It was refreshment compared to other games on the market, which all revolved around war, violence and destruction. The Sims became an instant hit and Maxis, who were behind the Sims created add-on packs, which basically included more content of things to buy, build and do. For Will Wright it also became a revelation, which opened his eyes to the growing content problem in the industry.

Will Wrights new game project Spore is an example of a game that takes full advances of procedural virtual world generation. His vision is making a game design that allows him to distribute the content creation amongst the players of the game. We could say that all the different ingredients are what you get when buying the game, but it is the different players that make the different recipes. Recipes are small and easy to move and share over a network. Almost everything designed around the player is done procedurally. This allows for an extremely high compression rate around 5000 to 1. The high compression means small file size which minimizes friction over the network.

The unconventional approach is based on a variety of naturally inspired programming techniques. He started by assembling a team primarily consisting of people from the demo scene. A team of crackers oppose to game programmers. The demo scene has for many years been a meeting place for people doing procedural programming of anything from animations to whole games.

Spore, which he first talked about on the Game Developers Conference 2005 left people amazed and speechless and later on at the E3 convention with the same result. The gameplay in Spore can be compared to an evolutionary game, but in Spore you decide how you evolve as a character and a civilization. In the game you start as a microorganism and the world is in 2D, much like living in a Petri dish. As you eat other organisms you suddenly can lay an egg. Laying an egg gives the ability to create the next generation in the evolutionary step of your creature. The editor tool where you create the next generation allowed you to give the creature vertebrae and other features. The creature editor can be seen as a modeling program, which leaves the need for a lot of content modelers to a minimum.

With these new features the creatures animations were dynamic and it moved depending on how you had modified it. This means that the animation weren't done on forehand by an animator, but procedurally. Will Wright had created an amphibious creature with six tails and the game would figure out how a creature with six tails moved.

Going from a microorganism to a primitive amphibious creature, the world shifts from 2D to a 3D world. It’s amazing to think that the content is procedurally generated and leaving the player in control. The game goes on, as evolution does and you can gradually go from water to land or evolve in the water. Cities and civilizations can and will arise and the skies the limit, taking you to space and other worlds where life has evolved differently.

As mentioned earlier games are also using more physics, which allows and produces more interactivity with the game and its content. One of the many areas where it has really is being used is with body physics. In for example Hitman you can drag corpses around which you have killed, where the different limbs collide and twist on how you drag it. In Spore this element is taken to another level. In the creature editor described above, as you have evolved your creature gets a skeleton. This skeleton can be modeled just the way you want, while it also affects the mesh of your creature and how it moves. Another unique feature in Spore is that skeletons can be torn to pieces. Will Wright in his preview showed an example, where his creature first drag its kill, where after tarring a limb of.

Spore is truly going to set a landmark in computer game history with its procedural content generation and management.

2.3 Physics in games

2.3.1 Introduction

We have seen how computer graphics and game play has evolved around each other and how graphical possibilities are beginning to exceed human capabilities. We have also looked at how this has led to a content problem. We have used Spore as a case story presenting one possible solution to this problem. As the first part belonged to Will Wright and Spore, this chapter will revolve more around Peter Molyneux and his work. We will take inspiration form Lion Heads creations The Room and Black and White 2. We will also look beyond those projects to suggest what the future might hold. We will draw parallels between the evolution of computer-graphics and physics driven virtual worlds. We also compare how both of these emerging technologies has influenced game-play.

2.3.2 Physics

Physics is the science of the physical world. The physics used in games only covers a small subset of the field of physics. Physics concerning atomic particles and what happens at near light-speed does not affect a virtual world in any way and is of course excluded. Even with these limitations the physical properties that can be modulated and simulated in a virtual world is still comprehensive.

The branch of physics that is interesting when working with generating virtual world mechanics and in particular dynamics. Dynamics is that part of mechanics dealing with the motion of bodies and the forces that make them move. Altogether programming a physics engine for virtual world is a massive undertaking. The laws of physics are well understood and described for those reasons we have chosen to focus on the relation between physics and game play. This is still an area of debate. It is an area where most of the improvements lay a head yet to be discovered. Here these implications will be described and discussed later we turn our attention to what the future might hold in realm of the physics driven world.

2.3.3 Physics and game play

Physics has an integrated part of game design since the very beginning. The first game ever made was Space Wars. In Space the player could accelerate his small space ship around and in Pong two players can bounce a ball back and forth between each other. These games were only dealing with a small set of physical laws. As the computational power of computers increased so did the possibilities of implementing more advanced physics. Racing games evolved into advanced car simulations considering more and more of physical properties of driving a car. With the development of the GPU followed a strong increase for content in games. Implementing physics is highly depending on the amount of objects that the implemented physics has to work on. If we consider a feature like collision testing, we see a dramatically increase in the cost of computation in relation to the complexity of the collision testing. For those reason what we often see in today's games is a simplified and "flat" physics implementation.

The reason that physics is still an interesting feature for virtual world development is its ability to generate game-play on a global scale. Game play can be divided into two categories, a local and a global. Local game play covers all interaction specific for a level or part of a game. The local level design is often created by level designers. Applying specific interaction in a virtual world can easily lead to inconsistency. A solution is to apply the interaction on a global scale.

2.3.4 Physics generates global game play

Global game-play covers features of interaction that applies to the entire virtual world. A global solution takes less work in the end. Building a physics engine for a virtual world represents a way to produce global game-play.

The physics driven virtual world can be said to be procedural game play. As the player is driving his Jeep around the heavy terrain, he might encounter a hill too steep to climb. In a non-physical world, this would have to be controlled by a trigger. There could have been an area in front of the hill starting an event, preventing the car to advance further in the direction of the steep hill.
With physics it simply just can not climb the hill for physical reasons. The game-play element is simply a side effect of the properties of the terrain. In the non-physical world we would have had to consider a number of options for the player in order to engage the steep hill. In the physical world all, these options are just there as physical properties that can manipulated. He might consider getting more traction for his car by changing tires. Maybe if he slowed down and found the right gear it would prevent the wheels from slipping. It could help to engage the hill more from the side given more weight on the wheels. We can let the player interact with the different parameters controlling our physical world to generate a vast field of game play. In the game prototype, we have taken advantages of the build in car physics engine of Virtools. We can see how the car we have in the game is set off by reading the information of three lists of values. Each value represents a physical property of the car. The car in our game is set to resemble the driving experience of an off-road vehicle. This is done by setting

a low limit to the maximum speed of the car and giving it a very low gear ratio. This kind goes well to illustrate the interaction design can be done, working in a physics driven world. There is one more thing to notice when describing how we designed the car in our game. The considerations of minimizing computational expense of the car world interaction. The visible car is actually just a decorative shell on a invisibly low poly car model. By using a much simpler model to act as the physical object, we can cut the computational cost significantly. We can do this since it will not have any large effect on the overall result.

It is the same kind of approach, which allowed the massives amount of different car models in Grand Tourismo. Each different car is internally just a different array representing that one car. All the core game-play is lifted to a global level, leaving the designers to concentrate on making racing tracks and the 3D models of the cars. They could then easily add the customizing of the cars feature to the game. As you customize your car, you just change the values in the underlying array holding the physical properties of the car. The game is still the best selling car game ever. Thanks, to its design relying on physics driven game. A simulation would be a more suited name instead of game.

In other games where physics, have been applied, with game play altering effects, is the first person shooter genre. The physics has allowed the player to alter a level pushing around crates. It has been used to make cool looking kill effects like flying limbs and splashing blood. Quake was one of the first games in the genre to introduce physics. It was now possible to bounce grenades off walls. This was a good example of how the application of physics can influence game play.

Sports games have benefited from using physics. From simple games like curling too pool. In sports games using physics makes it possible to let the player apply forces to objects with the mouse. This gives a very fluent and realistic feel to the game play.

2.4 Virtual Reality and games

Virtual reality (VR) can be seen as a description of an environment simulated by a computer. The environment can be displayed on a computer screen or by the help of special glasses and displays, which gives an immersive 3D representation or illusion of the simulated environment. The technology within VR is still developing and evolving.

Games and VR have often been discussed and especially now where technology and graphical power within consol and PC's have surpassed expensive super computers from a few years ago. This remark was made by Paul Rajlich from VISBOX, who specialize in virtual reality. However he also noticed that despite the huge advancement in computer technology and games with more realistic environments and content, the interface is still the same. Monitor, mouse and keyboard.

The strong point with virtual reality is that it allows for total immersion. Not only can you walk around in an environment, but you can also interact with it. This has a great appeal for game developers, but there is catch.

VR equipment is expensive, very expensive and this is why only government, academic and industrial institutions have acquired the technology. This does of course not include HMD (head mounted displays) and shutter glasses. Virtual reality is a strong tool for visualizing data or concepts. It gives the user a possibility of presenting data for ex. in urban planning or architecture it really becomes beneficial. The user can walk through their designs and concepts in realistic scale, which may uncover flaws and possible changes there have to be made. This saves contractors a lot of money.

The high price and often bulgy equipment must be brought down if the average consumer should consider buying the technology.

This is VISBOX's mission, making high-end display and interaction technologies more affordable and easier to use. This sought of mission statement is what the industry needs and especially if you want the attention of the game industry.

Paul Rajlich was the first to implement a serious game title to virtual reality. He took the Quake II game engine and applied it to a CAVE (Cave Automatic Virtual Environment). The effect obtained compared to displaying it on a monitor was an increase of immersion. This is mostly, due to that everything becomes more believable, because of the realistic size ratio. Applying a head tracker gave the ability to interact with the game. If you wanted to look down, you looked down and with a twist of your head you could peak around corners. Another input device that was also used, called a "wand". The device represented the gun in the game and was tracked according to your perspective with the correct orientation and translation.

This game has become popular amongst the community and Quake III and Unreal Tournament have also been applied to the CAVE. For the game developers of the original titles, it's been a learning experience to see their product under a whole new platform. It has evoked an interest, because of the new dimension it brings not only in interfacing, but also in game play.

2.5 Distributed content creation

Due to the problem in content creation, developers are moving towards middleware solutions and pre-made 3D content.

To give an example of this we can see that most virtual worlds will contain human models. With the high graphical demands on creating a human character this has become a substantial task. Today it is possible to purchases such a model from a third hand vender. This is massively time and money saving. A similar development is happening for software tools. Earlier all software was developed in-house by game developers where as now developing a game involves an increasingly amount of middleware. This is an ongoing process that has by no means reached its summit.

Today a lot of games come with a level editor. These tools will allow players to build there own levels and content. These editors have been very popular with the players. They have formed communities where they exchanges there self made levels and content.

The introduction of an increasingly amount of middleware has spawned a whole new array of problems. As more and more different software have become involved a main issue has become integration. It has become important to find out how and what software can communicate. It becomes important to keep organized.

This is why we have decide to focus this project around laying out a fast flowing and well documented assets pipeline for producing a virtual world. We have developed a prototype to serve as a testing ground for the work of finding the best assets pipeline. We wish to compress our findings into a chart illustrating and describing the main key aspects of the pipeline layout. We have chosen the format of a chart to give a bird eyes perspective of the process. This is of great importance due to the dense complexity of the pipeline. It should be seen as the blueprint of the construction site behind building a virtual world. The focus will be entirely on prototyping. It will be a pipeline designed for maximum flexibility and creativity and testing of new ideas. The hopes are that this chart can be of use for future projects in the field.

Chapter 3

The Prototype

3.1 Motivation

As stated in Chapter 2 we are nearing a point where the computers possibilities are exceeding human capability. This means that we have to come up with an approach to optimize the content pipeline, so as to keep up with the massive amount of content which has to be made for next generation games. If such a new approach isn't created the development time of a virtual world such as a game, will become much longer. As Noell Llopis from Day 1 Studios puts it;

> *"Game content will also continue getting larger and more complex, as the amount of it going into AAA games doubles every few years. Yet since there's no hardware upgrade for the artists and designers themselves, content doesn't get created any faster"*[1].

One key thing for the content pipeline is that is has to be robust, because of its vital position in the development of a game. If it breaks, you quickly have a development team waiting for it to be fixed. This could mean the end to any project.

[1] Game developer magazine April 2004 page 36-44

3.2 Distributing the tasks

In order to create virtual worlds and immersive experiences it was necessary for us to have the right tools for creating our assets for the game. Many tools exist, some very specific and some with a broad range of features.

We chose Virtools as our primary development platform, because it offered us a versatile solution for developing and testing our prototype. We could see the results instantly and edit while the game was running. Furthermore a VR package can be bought allowing you to have an immersive 3D experience. The whole development suit, it straight forward to use and you quickly get into development. All in all, a great tool to work with.

For making the terrain we used a middleware product explained in the environment section. Although middleware makes it easier to create content, it also makes it also has the downside of format compatibility. This we meet several times doing the development of our prototype. Another problem that arises is finding a program which can convert your files into the right format. For a smooth content pipeline this is not very efficient, because too many are used.

As for content creation we used Maya, which is a stand-alone out the box tools used for creating 3D for games and movies.

3.3 A Solution

3.3.1 Game design

Here we will analyze and discuss what we could have done better form a game design standpoint.

We will work though the different steps of the process to see what we can learn from the problems we have encountered along the way. After this section we will provide a chart illustrating our vision of a flexible flowing pipeline that can be used by students making a game.

3.3.2 Story

On an early stage of the project we had some general discussions on how to choose a story for our game. We discussed the idea of making a game over an already existing story and were looking in the direction of comics and novels. We actually did some early prototyping over Bergson's historical novel called the Long Ships. This was done using the Aurora toolset, which is the editor tool for the role playing game Never Winter Nights. This worked well but was abandoned for several reasons. Firstly we did not feel that this solution gave us the freedom we wanted. We found our self constrained to the different objects already available in the toolset. This did not include a Viking setting which is the world of the Long Ships. Secondly we felt that this approach would lack academic aspects. For a project solely concert with narrative and storytelling side of creating virtual worlds this would however have worked very well.

Comic books were considered for their visual content. From the beginning we were aware that we lacked resources in the concept art department. This problem would be easily solved by turning a comic book in to a game. The problem here was simply finding a story we felt met our expectations for a good game. Also we have to add that this approach could have been a really good choice. A comic book is basically a ready to use storyboard. This would have left us with the work of deciding on the right game design for the story. This is an approach we will leave for others to explore.

As all of the above approaches were discarded, we were left with having to develop our own story. We did this as we have already described using different creative techniques. At this point we were only evaluating our different ideas, from one angle. We were simply looking for the coolest game, the kind of game that we would love to play our selves. We chose to continue working on an idea we gave the working title Pulitzer. We were imagining a Splinter Cell meets Tomb Raider style game. At this point we should probably have considered that a game like Splinter Cell took 3 years for a 100 people team to develop. The budget for a game like splinter cell is just over 20 million dollars.

With pioneering spirit we ventured into the unknown of developing a third person spy style game. We soon discovered that the tasks were beyond our capabilities. We then decided to turn our heads to our 5 semester project Trophy Hunter. Taking up this project provided us with a lot of 3d content plus good idea on the game design.

From the beginning of the project we felt a drive towards investigating the narrative aspects of game making. We wanted to explore different techniques for adding an emotionally immersiveness to the game experience. The implementations of these were scheduled to be implemented as the rest of the virtual world was created. As we came to complete the creation of our virtual world we never got the implementation of theses elements. Even so we will try to evaluate this idea.

As the project neared its ending we discovered some work done in this field. David Freeman's book Emotioneering which we have purchased but not studied. This illustrates a research mistake partly brought about form lack of time and from not looking for right terms to search on. As this is 500 page book published last year, we have to conclude that we were properly in over our heads on this subject. Emotioneering is the first book on this novel subject. Applying and working and evaluating different techniques of emotioneering could surly have formed the base for whole semester project.

So what's to learn from all this? The missing ingredients were risk management. Risk has never been a part of our vocabulary though out this project. Knowing what we do now, some of our first story ideas based on existing work would properly have worked better. Another part to consider is choosing what part of the story to use for the prototype. Cropping a story with out losing the general understanding of the story line will not always be an easy task. Our advice is to start small, really small. In our case we should have boiled it down even more than we did. Find a place in the story that conveys the essence of the story and see if it is fit for prototyping.

In general we can say that the story has to be seen in relation to the available time, team size, and tools. To conclude this section we wish to set up some useful guidelines for other projects as part of our prototyping handbook.

Guidelines for developing the story:

Pick your risks with great caution

Look for short cuts (comic books, Novels, earlier projects)

Identify the essence of the story for the prototype

3.3.3 Game play

Game play involves interaction and animation. Character animation can quickly turn into an endless pit of never ending tweaking and adjusting. This is important to consider as a game play is chosen. A game play that demands a lot of animation should be avoided. For our game we looked for a more test based approach. We did test of the shooting system just using boxes as dummies. In general the simple stand-in's help to develop the game play. It allows us to test and implement game play features before the final 3D content is created. Replacing dummies later in the process requires a consistence in naming convention. We will return to this issue in our chapter on the assets pipeline.

It is also important here to keep the possibilities of the general platform on which the game is developed. Some game engines will allow for some game play options, which others will not. Working with Virtools as our authoring tool enabled us to make physics driven interaction part of our game play. To have physics driven car in the prototype is a direct result of the possibilities of working with Virtools.

Working with the character interaction is a special case that can not always be tested with dummies. The complexity involved with the character has to be taken into account from an early stage. Here the actual design of the character will often be a result of what his role is to the game play. If the character has to grab objects in the game, the design of the character has to include hands that can grab. In our game we wanted to have a realistic shooting animation. This forced us to design a character with full hand dexterity. It also important to determine how the camera set up will be for the game. If the game has close ups there should be paid more attention to the details of these features. In our game we have a Rover that the character can drive around. This is a really good example of the coherence that has to exist between models and game play. As we decided to have the option of a camera inside the car we had to also make sure that the interior of the car was modeled in detail. However we also felt that it would add to the game play to have the option of seeing the car from the outside. This meant that we had to also make the character respond to the turning of the car by actually turning the steering wheel.

However on the headline of what we should have done only one thing really sticks out. Design game play to minimize the need for character animation. We will return too a more detailed discussion of these problems under our character section.

3.3.4 Environments

The environment of the game is not to be overlooked, after all this is the world you move through when playing the game. There are many considerations to be aware of, one of which is how to make the terrain. We first set out to model it in Maya directly, but after several hours of modeling we found out that this wasn't optimal, also how where we going to texture it. We began searching the internet for middleware, and found many different solutions to the problem, but our eyes set on the Advanced Level Editor from DyVison Works. This tool let us make terrain fast, and with the texture we preferred.

The environment we created was made on the fly. As a player you begin your adventure driving a rover through some mountainous terrain, covered in tree. At the end of the mountainous region the savanna is revealed. The environment was then imported into Virtools with textures.

This leaves us with another problem; vegetation. As we need hundreds of trees and grass, we know from previous experience that making one tree or grass and then duplicating it, saves a lot of computation. The reason for this is that duplicates share the same geometry, whereas ex. a lot of different trees do not. We were also able to create diversity, by scaling and rotating even though we used the same tree and grass.

Although this technique should save a lot of computation, we encountered a problem. With all the trees, moving around the terrain became a lot slower, even with LOD. The reason for this is unknown, but if we had more time, a solution would have been made.

To give a sense of the environment being an environment, an essential thing is a sky and sun. The sky was done by making a skybox around the terrain using 6 different images. For the sun we placed a lens flare on the light, lighting up the terrain. This gives the light a glow like the sun.

So what should we have done? We shouldn't have made such a large terrain, the bigger it is, the more content there has to be. As in the story section we will set up some guidelines.

Guidelines for developing the Environment:
Keep it simple
Use duplicates if using more than one of the same geometry
Search for alternative solutions in middleware
What is essential to test your prototype?

3.3.5 Character

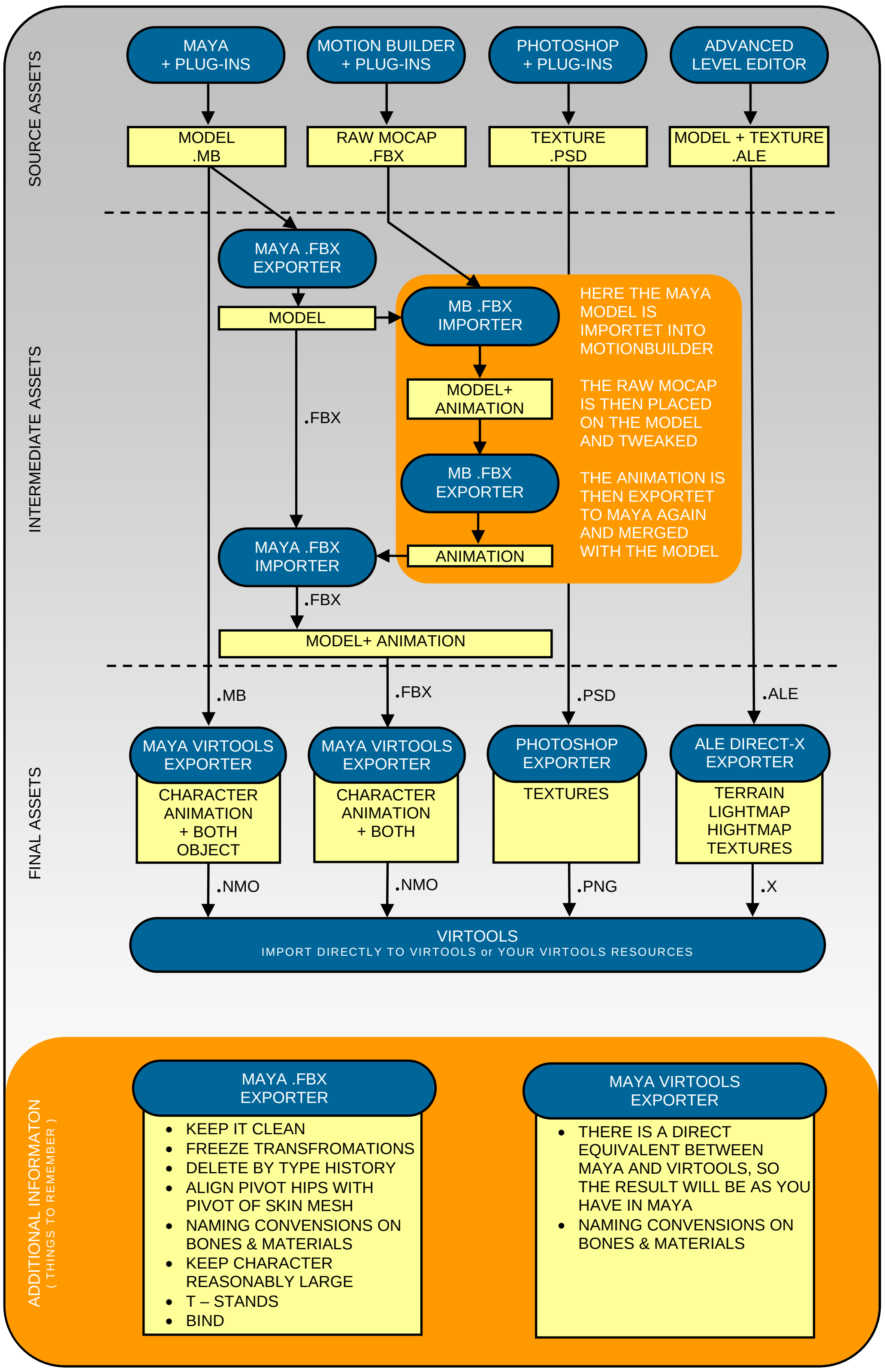
After having modeled and textured our character, he looked lifeless. But how should we give him life? There were two approaches to this problem, hand animation and motion capture. Hand animation is a trade which has been used for generations, but we weren't from a generation of animators. So there was only one answer; motion capture. This gave us the power to breathe life to our character by using movements captured by sensors placed on ourselves. The technique would make our character more believable, by using our motion, even small and awkward ones which otherwise are hard to imitate using hand animation. So we proceeded onward in our venture to bring "life". First we made a list of what we wanted our character to do, and then started to record. Like a mother we gave life to our character via an umbilical cord of wires.
With the captured movements recorded, we had to nurse the character, by tweaking his movement so they where perfect. This of cause also implied that

feet belong to the floor and one step should overtake the next seamlessly. This might seem as a simple task, but problems arouse, as they do in real life.

These problems need guidance to be overcome. First of all, before doing any motion capture a list of movements has to me written down and rehearsed. In our case we made a list of typical game movements, but as we later found out it should have contained specific moves for our game. Another important note when rehearsing and recording is if your character uses a prop ex. gun, find or build one. It is easily noticeable if you do not use them, because of the weight difference and the size of the object.

When starting to record, use your own character, and an actor with the same build, because intersections can occur if you character has wider hips than you. Also before recording take into consideration, that you only have a limited space of 2x2 meters, and be aware of you foot position as described in chapter 4.

3.4 The Chart


SOURCE ASSETS
MAYA + PLUG-INS
MOTION BUILDER + PLUG-INS
PHOTOSHOP + PLUG-INS
ADVANCED LEVEL EDITOR
MODEL .MB
RAW MOCAP .FBX
TEXTURE .PSD
MODEL + TEXTURE .ALE
INTERMEDIATE ASSETS
MAYA .FBX EXPORTER
MODEL
MB .FBX IMPORTER
HERE THE MAYA MODEL IS IMPORTET INTO MOTIONBUILDER
MODEL+ ANIMATION
THE RAW MOCAP IS THEN PLACED ON THE MODEL AND TWEAKED
MB .FBX EXPORTER
THE ANIMATION IS THEN EXPORTET TO MAYA AGAIN AND MERGED WITH THE MODEL
.FBX
MAYA .FBX IMPORTER
ANIMATION
.FBX
MODEL+ ANIMATION
FINAL ASSETS
.MB
.FBX
.PSD
.ALE
MAYA VIRTOOLS EXPORTER
CHARACTER ANIMATION + BOTH OBJECT
MAYA VIRTOOLS EXPORTER
CHARACTER ANIMATION + BOTH
PHOTOSHOP EXPORTER
TEXTURES
ALE DIRECT-X EXPORTER
TERRAIN LIGHTMAP HIGHTMAP TEXTURES
.NMO
.NMO
.PNG
.X
VIRTOOLS
IMPORT DIRECTLY TO VIRTOOLS or YOUR VIRTOOLS RESOURCES
ADDITIONAL INFORMATON
(THINGS TO REMEMBER)
MAYA .FBX EXPORTER
• KEEP IT CLEAN
• FREEZE TRANSFROMATIONS
• DELETE BY TYPE HISTORY
• ALIGN PIVOT HIPS WITH PIVOT OF SKIN MESH
• NAMING CONVENSIONS ON BONES & MATERIALS
• KEEP CHARACTER REASONABLY LARGE
• T – STANDS
• BIND
MAYA VIRTOOLS EXPORTER
• THERE IS A DIRECT EQUIVALENT BETWEEN MAYA AND VIRTOOLS, SO THE RESULT WILL BE AS YOU HAVE IN MAYA
• NAMING CONVENSIONS ON BONES & MATERIALS

Chapter 4

Future Solutions

4.1 From motion capture to capturing the moment

Motion Capture

One of the most difficult tasks of game development is character animation. Motion capture has been the solution of choice in the resent years. It serves well to take some weight of the animators. MoCap works well as a replacement for the man crafted animations. However MoCap does not present any expansion of the possibility space of game play. It does not allow for in-game real-time responsive animation. MoCap may be cheaper than hand made animation but is still expensive. It is not uncommon for new game to have between 1000 and 3000 different moves. MoCap brings more realism and subtle expressions to the character animation. Some however will argue that the use MoCap in some cases can add an artificial feel to the characters. When the motions look real we become more aware of the fact that the characters are not. It seems to be more the case for movies, where as for game we are more accustomed to the artificial nature of the experience. Games rely on interaction and gameplay to create the immersion where as movies only rely on the visual expression. It seems that the biggest limitation of MoCap is its lack of dynamics. MoCap are motion frozen in time, recorded and then imposed on a 3D character. That means we can not just tell a character to walk across the room accordingly to our MoCap data. The problem can be compared with the problem we have of tilling a texture. In order to make a texture tile we need to make its opposite borders match with each other. The reason for tilling textures is the limit of data memory and production time. The same can be said for character animation. An animation recording will take up a considerable amount memory. A normal character like the one we used has around 200 bones. Each bone has six degrees of freedom. If we say that that the values of all these 200 time 6 equals 1200 degrees of freedom is stored as an float value which is an 32 bit binary number we reach an amount of 38400 bits. The animation will be sampled at 24 samples per seconds giving us a total of 38400 times 24 times10 for a ten second animation. This clearly illustrates that animation file size is a problem to consider. That brings us to need of tilling or cycling animations. Instead of having long sequences of animation we can make small cycle bit that we can tile after each other to simulate the notion of a fluid and continues motion. Tweaking a motion captured animation into a small cyclable bit is by no means an easy task, and will require good planning and the right tools for the job. It can be recommended to put a lot of work into the planning and focus on the most needed motion such as walking running and standing plus what else might be specifically important for the game at hand. Creating a cyclable animation will suffer from some loss of the original recording. The more cycle friendly the original MoCap recording the less the loss of authenticity. The reason for this lies in the fact that the end and beginning of the cycle has to be matched together. The bigger the offset is between the end and beginning of the clip the more the computer has to calculate an interpolation between the two posses. Computer interpolation will never become as realistic looking as real human motion thereby adding this artificialness to the animation. Blending between different motions is another problem to consider. There seems to be two different solutions to this problem. One is to let the blending happened in-game. This can be done if the game engine supports it. It will suffer from significant loss of realism however it is also significantly time saving. The second solution is to make the blend as separate animation an play them as the play commands the character to go from one motion to another. This solution can give a more realistic look but has another set back. It will be completely arbitrary when the player decides to start a new motion. That means it will almost always be in the middle of an ongoing animation. Then the ongoing animation will have to play to finish before the blend animation is played. The randomly prolonged hesitation cased by this is highly undesired. It diminishes the player feel of control as he will continuously experience this lack of respond to his commands. And lack can lead in game speed leads to lack of immersion. It is a difficult trade off question that actually exposes a fundamental problem with how character animation can be done today. This moves us along to have a look at what lies beyond the current state of charter animation.

4.1.1 Capturing the moment

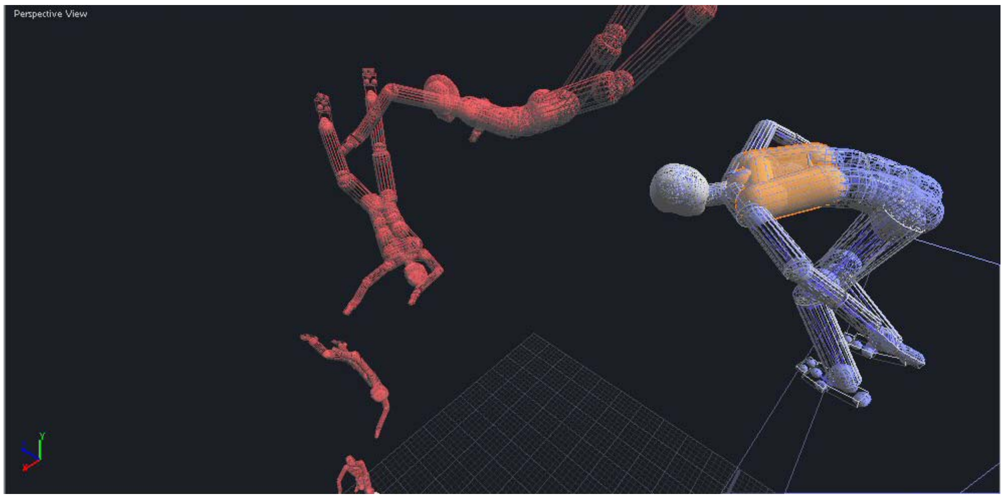
As said earlier physicists driven environments are becoming standard for new games, there are still other aspects of a game world to follow along this path of physicalization. One of the most interesting areas in this field; is character animation.

It can be questioned if animation is still the right term to use in the case of a dynamically physics driven character. Intuitively we tend to think of animation as something following a predefined path. On that same node it would not make sense to call a physics engine programmer an animator. An apple is not animated towards the ground it just falls. One of the first games to introduce this was the tile Hit Man. They were the first in the field introducing what they named rag doll physics. As an enemy character is shot he will react to the physical force that hits the body part, slinging his dead body though the room or up against a wall. It also enables the player to drag a dead body around automatically following the contours of the surface.

This ground breaking work as impressive it might seem is still only a scratch in the surface of the possibilities that lies ahead. Working with a dead character clearly is nothing compared to dealing with a living one. The complexity of the human kinesthetic system is vast. Considering the problem of just getting a person to balance him self is a massive a task. When we as humans want to perform a task we are not really concerned with the details. We just command our body to walk run or stand, the rest is simply taken care of. A similar goal oriented approach would be really beneficial for game design. The game designer or player just commands a character to grab an item or walk to a new destination and it is simply take care of. We could also consider trying to keep is equilibrium as the ground shakes or he is walking along the top of a moving train. This opens the door to a similar situation as what we can currently do with car physics. When dealing with a fully physically driven car in a game, we just have to set the different properties of that car. We alter the amount of horse power, the stiffness of the dampers or the power of the breaks and then we can add force be stepping down the pedal. For a human character this would mean he could have different properties like; senses of equilibrium, muscle strengths, dexterity, and stamina. To make a character appear drunk in a game we would simply have to turn down his sense of equilibrium. A character with good dexterity could have faster learning time for new specialized moving patters like dance moves etc. A character would run out of breath according to his stamina, automatically slowing down. However there is still a big difference in complexity form a car to human. A car only performs a small set of tasks compared to a human. A car only has one engine where as a human have around 500 different muscles that has to work simultaneously. These facts bring fourth the need of a self aware character that has a “real” simulated physical representation in the virtual world he resides. Theses tasks have recently been solved by the new company NaturalMotion. They were the first and still the only company on the market with a character animation tool offering what they have named; Dynamic Motion Synthesis.

Dynamic Motion Synthesis is based on research done at Oxford University. They have built a physical, biomechanical-realistic model of a character and implemented an appropriate brain structure usually in form of a neural network. In combination with optimization techniques such as artificial evolution they are able to create the adaptive behaviors to be performed by the character. Endorphin as this ground breaking tool is called; is still a stand alone tool for character animation. The next step will be to make this sort of technology an integrated part of a game engine. Animating with Endorphin means applying force or behaviors to the character such as a push in the chest or a behavior like *look and grab*. The behavior arsenal is still some what limited and the system requirements quite high.

Rescues lies just around the corner in the form of the latest on the hardware side. The physX is the world's first physics processing unit – PPU. The evolution of the PPU has been compared with emergence of GPU in the early 90's and to have equally heavy impact on game and virtual world development situation. The possibilities of these new technologies are still to be discovered. We will no longer have to rely on motion capture, but will have the means to simply capture the moment.



4.2 The future of narratives and stories

There are many new thoughts on the future of narratives and stories in games. Will Wright asks, "what is it you tell your friends when you have played a game? No, it's not about the narrative and story, but what you did and how you did it". This point is well made especially if you look at his creation Sims and the increasing popular MMORPG's (massive multiplayer online games). As mentioned earlier, people like to create there own content and by doing this they create and customize their own stories. In MMORPG's where you play and exist within a massive virtual world amongst a whole community of other players, the need for stories and narratives become extinct. The stories are being told by the people who play and the way they are being told are through interacting and communicating with other players. The reason that it works can be traced back in history, where stories of great feats were communicated and passed on from mouth to mouth.

The technology has provided games with these narrative tools, which is fantastic. On the other hand, one can't dismiss that players are becoming more immersed in the virtual world and their virtual existents. A daunting parallel can be drawn to the movie The Matrix and one can only imagine, how narratives and stories will be told in the future.

Chapter 5

Conclusion

As we have seen the problem we encountered is a general problem for a whole industry. For the specific case of being a small team setting out to develop a full functional prototype, we have identified a list of things to consider. Working with a well defined and understood assets pipeline stands out as one of the most important issues. Using the most effective tools for the task and understanding there integration is a must. The criteria for choosing those tools should be maximum assistance. Any tool that can help speed up the process should be considered. Using these tools requires a good understanding of the underling functionalities. We have found that these tools do not just represent a cheap short cut but a real opportunity to focus on the design aspects of game making.

Producing enough 3D content to populate and decorate a virtual world turned out to be an enormous undertaking. We have suggested different solutions to solve this. First of all it has to be a consideration, taken in to account from the very beginning. Risk management is a necessary part of developing a prototype. We could call this the down size solution. Another solution is to seek all the help you can. Using readymade 3D content is highly desirable. It is not always possible to get the content necessary. In some cases this will have to be bought from third party companies or from sites that post freelance developed content at a low price. We found that some universities have large libraries of MoCap data to free use, a similar approach could be adopted for 3D content.

Finally we hope to have extended a helping hand to be picked up by future projects in the field.

Bibliography

1. Advanced level editor

   http://www.dyvision.co.uk/ale.htm

2. Exclusive: The Future Of Gaming - Part 1

   http://www.pcgameworld.com/article.php/id/98/

3. History of computer and video games

   http://www.answers.com/main/ntquery?method=4&dsid=2222&dekey=Hi
   story+of+computer+and+video+games&gwp=8&curtab=2222_1

4. NaturalMotion

   http://www.naturalmotion.com/index.htm

5. Jason Rubin's "Great Game Graphics... Who Cares?"

   http://www.gamasutra.com/features/20030409/rubin_01.shtml

6. Virtual Reality Check

   http://www.forbes.com/asap/2001/0402/020.html

7. Visbox

   http://www.visbox.com/

8. DFC Intelligence

   http://www.dfcint.com

9. Planning and Directing Motion Capture For Games

   http://www.gamasutra.com/features/20000119/kines_pfv.htm

10. GDC TV

    www.pqhp.com/cmp/gdctv/

11. Spore

    spore.ea.com/