

Dissecting our Baby: Auto Assault Postmortem – the good, the bad, and all the ugly.

Scott Brown
Founder/President - NetDevil
scorch@netdevil.com

Hermann Peterscheck
Producer – NetDevil
hermann@netdevil.com

Abstract

Massively Multiplayer Online Games have become more and more popular over the last few years. The monstrous financial success of World of Warcraft has exponentially increased this interest. A third party developer is faced with unique challenges, beginning with the pre-contract phase, pitching, prototyping, content development and deployment.

Auto Assault went through many changes on its journey from concept to release; some of them positive and some of them negative. In order to create Auto Assault we had some difficult problems to solve. We had physics simulated on the server; in some cases thousands of objects. Also, the game play itself was not something that was easily copied from another game. Auto Assault is also a game of mixed genres: Action, Car Combat and Role Playing and there were no guarantees that this kind of game is something that people want to play.

Our publisher, NC Soft was willing and, in fact, excited to take these risks and their support made the development and deployment of the product much easier than it might have otherwise been. In the end, Auto Assault was not as successful as it could have been, but many of the obstacles were overcome and we learned a lot about what works and what does not work when making a unique type of game. We also learned a lot about the publisher/developer relationship and the experience helped both sides understand each other better.

1. Introduction

Every game starts with an idea. Usually this can be expressed in terms of what came before it with some kind of modification which explains what is different. In the case of Auto Assault it was “Diablo in Cars.” The idea seemed very simple. Diablo was very fun and sold very well; two categories you must fit into if you want to make a commercial game. Additionally, most of us were fans of the Origin game “Auto Duel” therefore we naturally assumed that putting the fun of Diablo in cars was a great idea.

This paper is about how that idea became a game; and the steps (and missteps) that we took along the way. It all begins with signing a contract and that comes down to the pitch. The pitch process in Auto Assault was somewhat unusual because of an existing relationship between executives at the publisher and NetDevil.

Once the deal was signed we immediately started on development which was scheduled in the usual way: prototyping, feature completion, content development, and deployment. Before

starting we had a reasonably specific schedule for each phase which would be used by the publisher to track the progress of the project.

The process went relatively well through prototyping and real did not come up until the feature complete and content development stage. MMOs are notorious for having significant issues at launch and as we had experience with launching an MMO before we were able to anticipate some of these issues although we certainly had problems at launch and afterwards; but less than most.

After release we also had issues to deal with. The game’s numbers did not meet expectation which resulted in reductions in team size. This had consequences both on productivity and morale; which we were able to overcome. We also had to determine which path to take in order to try and make the game more successful while dealing with the normal post-release issues.

Although the game did not commercially succeed as well as we had hoped, it was received reasonably well both by the press and the players who continued to play the game. We learned a lot from the experience and while it was painful at times; there are few experiences that are more satisfying than finishing a game.

2. Exposition

The journey from game idea to finished product is a long and perilous one. In the case of single player games it usually takes between 12 and 18 months. In the case of MMOs the cycle is usually closer to 48 months. To put this into perspective consider that Auto Assault started before the first Lord of the Rings movie was released and was finished after the last movie was available on DVD. If you were to make one such product at a time your career would be complete after about 9 or 10 games, so each attempt should be taken seriously.

We will now begin the process of detailing those four years of development and some of the major milestones we met and obstacles we faced.

2.1. Getting the Deal

Getting the deal to begin working on Auto Assault was a bit different than most 3rd party developer deals. We had a previous relationship with some of the executives at NCSoft-Austin (previously called Destination Games prior to the acquisition by NCSoft). NCSoft was looking for interesting titles to add to their release schedule and was more willing to take risks with innovative ideas than many publishers. This is something that we have enormous respect for and is the reason games such as *City of Heroes* were made possible.

One of the first things we did is make a piece of art that reflects what the final game will look and feel like. Early on, the game was top down and you can see some of the similarities to Diablo.



Early Visual Representation of Auto Assault

It is very important to have an image like this early in development as it is what everyone refers to. It is what everyone should have in their minds as they work on the game. In fact, throughout the development process, the publisher would make references to this original shot and use it as a benchmark for our progress. If a piece of art grabs you emotionally and the game can achieve that same emotional value, it is a major accomplishment.

The nature of the deal was a fairly traditional publisher/developer model. Basically the cost of development was paid by the publisher as an advance against royalties. It is only when the royalties from the project pay back the development cost, that the developer begins to see a profit. For example; let us assume that your royalties are 10% and your project cost is \$1 million. The project would have to generate \$10 million before you would make an actual profit. If you are working on an MMO with about 40 people you can have a base burn rate of \$300-500k/mo, some MMOs costing much more; depending on all kinds of things. A slip of one year could cost you \$6 million and would require the game to generate an additional \$60 million before your studio sees a profit. To put that in perspective, if your game gets 100k users paying \$15/mo, that slip will cost you about 3.5 years; and that is assuming the \$15 is all profit, which it certainly will not be.

Additionally, payments are made by the publisher to the developer based on measurable progress. This is manifested in the form of a list of deliverables on a set schedule. If the deliverables are met, the milestone is successful and a payment is made; if not, then the developer must work until the deliverable is accepted. This allows the publisher to have some level of assurance that progress is being made as well as providing the developer with clear goals to reach.

I want to stop here for a moment and discuss some issues with this model which came up many times throughout the project. Firstly, it is in the developer's interest to not delay the product. A delay means that the cost of development goes up and therefore the amount of time before the title becomes profitable goes up

substantially. This is largely because delays happen at the end of the project when the burn rate is the highest. It is very easy for a delay of 6-12 months to cost millions. Secondly, the nature of game development is that planning out what you will be doing a year from now (or two, or three) is impossible beyond the most basic set of features. If you make a detailed time and progress schedule which is tied to payment without a dynamic and reasonably simple change process, you will spend an enormous amount of time doing the wrong things at the wrong time or in the wrong order. Lastly, most developers have no way of covering the costs of teams, in the event of missing a milestone. This puts incredible pressure on the publisher to pay for milestones they felt were not met, or cancel the project. These things can cause friction between the publisher and the developer to the detriment of the project.

There are many ways of dealing with these issues and we believe that most of them involve better communication. If your publisher is across the hall, they are able to easily see how hard the team is working; if they feel the progress is in the right direction and so on. If they are hundreds (or thousands) of miles away, there is a much degraded sense of progress and rapid and frequent changes in direction that make sense to the developer seem like haphazard and badly thought out decisions to the publisher.

NetDevil has begun using various agile methods (most notably SCRUM) to create an environment of result oriented development with frequent milestones focused on completing tasks in a presentable way. Regardless of how you solve this problem it is vital that the publisher know what is going on as quickly and often as possible (weekly builds, video meetings, weekly progress reports and so on) as well as being responsive to rapid and frequent changes during development. Everyone wants the game to be great and successful and a big part of that is making sure everyone knows what is going on. If you find yourself getting in the habit of barely making milestones by being able to "check off" the deliverables, but they are not REALLY done, you are heading for a giant nightmare in content creation, as we will soon see.

At this point in development, however, we were focusing on deciding how to go about making the prototype. There were many things to consider and the possibilities seemed endless. The team was small and focused and we were able to get going relatively quickly.

2.2. Building the Prototype

Building the prototype involved a few steps. Having a piece of concept art to shoot for was of major importance at this stage as it represents something that everyone can shoot for and also shows what is expected.

Over time I have seen the issue of this stage of development handled in many different ways. Some groups start by making the tools first, others dive right in and start making the game itself. There are strong points on either side and everything in between. One thing both groups can agree on is that you need to define the basic elements of what the game is, so that's where we began.

We were able to fairly quickly boil down the elements of what would make the game fun. The game would need lots of random items to collect (remember: Diablo on wheels). It is also important that the game be fast paced in terms of combat; specifically we

wanted players to be defeating lots of enemies quickly as opposed to the slower pace of MMOs that existed at the time. Lastly, we wanted to make sure that the game provided the player with a sense of progression. This is critical for the RPG aspect of the game and we felt that without it, the game would not be accepted as an MMO.

I want to emphasize that there were a couple of issues which were not focused on early, and perhaps this was a misstep. We were not focused on customization and socialization; which are arguably two critical components of MMO design. We did have focus on customization in terms of random loot, but not in terms of character creation.

Another thing we were acutely aware of was that the MMO space was competitive and getting even more competitive. We wanted to avoid going head to head with other MMOs, which required some additional considerations. We knew we would not have the depth of character development which existing MMOs had. Also, we deliberately avoided the fantasy setting as most MMOs at the time (both released and in development) were fantasy based. We felt that existing MMOs had a very steep advancement curve and that killing one creature every couple of minutes was not particularly exciting experience. Death penalty, we felt, was also far too steep. Our goal was to never let players feel that when they died they were better off having not played at all that day.

The prototype was fairly simple. We needed to have a car driving around on a map and some creatures to kill, loot to collect and some rudimentary advancement. We originally used Renderware which allowed us to get a prototype up and running within a few months. We also began working on a map editor fairly early which was a critical component in early development. We later changed from Renderware to Palantir (a graphics engine being developed internally at NCSoft). This allowed us to run on the same engine as other products under development and gain the advantage of mind sharing. Ordinarily a decision to change engines can be very costly, but in this case it happened early enough that it did not have a major effect.

We knew that we needed some kind of physics engine in order to implement vehicle movement and combat. We had also decided that building our own physics engine would most likely be a mistake so we chose Havok as our engine of choice. Havok proved fairly easy to implement and it started to become obvious to us that physics would become a major feature of the game.

Up to this point things were going very smooth. In fact, the progress we were making was considered quite impressive. We were happy, the publisher was happy and things were going along very well.

2.3. *One Great Level*

One of the first problems we encountered was how difficult it was to make the game “fun.” There are several reasons for this. For one thing, as you work on a game you lose sight of how easy or hard it is which means that without some kind of external feedback, it is impossible to honestly evaluate how good the game is. Another thing that happens is that you begin to measure quality as an incremental value based on your previous iteration as opposed to other products. This is a dangerous mistake. Customers do not care how much better than your previous

attempt your current attempt is; they only care about how good you are against everything else.

We did spend quite a bit of time iterating trying to get driving and fighting fun as well as making advancement feel fulfilling, but slowly we were eating away at the schedule and if we were going to make our release date we would have to start getting major features done, not to mention a mountain of content.

A few major things happened at this time as well. The first was that we started showing the game to other external parties. For a while there was the thought that we might be a good console game. When we showed the game to console evaluators, they all said the same thing “top-down won’t sell on consoles.” Remember that our game was “Diablo on wheels” and Diablo was a top down, 2D game. When we watched people control the game they always wanted to look ahead of them and were frustrated that they couldn’t.

At this point we evaluated what the game would look like if we could look ahead more. It was easy to drop the camera and see what happened. This was the first major conflict on the project. Some people felt that seeing far opened the game up and made it feel open and huge. Others felt that it made the graphics look worse and the effort required to support the change were too large. In the end it was decided that we had to drop the camera down. The implications of these decisions were fairly major. For one thing, we now needed a sky. Also, it meant that the previous height restrictions on the terrain were no longer adequate, which meant a change to the map format. Previously we had lots of small objects which looked good from a top down perspective, when we switched the perspective those objects no longer looked very good; not to mention we now had to draw many more objects which had major impact on the rendering engine. Since the player was now able to see a far distance we also needed a better object loading system which turned out to be quite a bit of work.

Regardless of whether this was the right decision or not, the impact was not adequately considered in the schedule. We still wanted to make our release date and in hind sight it may have been better to delay the product at this point until the magnitude of the changes could be realized.

In addition to the point of view changes, we started getting feedback from NCSoft in Korea. The plan was for the game to be successful in international markets and so we were sensitive to the needs of the asian markets. One directive we received was that the game had to be playable completely with the mouse. This is fairly unintuitive for a driving game and we struggled to make it work. In the end we were never able to find a mouse only driving model that worked particularly well which is why most people play the game with WASD (or arrow keys) to move the car and the mouse to fight.

Another area that we spent a great deal of time on was characters. From the feedback coming back we knew we had to make characters a much more significant part of the game. We designed and implemented a fairly complex character creation system, which ended up being too complex and eventually changed to the more intuitive model we have now. We also knew we needed great looking characters to go along with the system so we began creating concepts and modeling those as well.

Here, again, we had the dual problem of quality and quantity. On the one hand we needed 12 characters; on the other hand they needed to be great. Instead of focusing on one character and iterating until we had one we liked we iterated on multiple characters; each time getting feedback and trying again. This cost an enormous amount of time which was compounded by the frustration of having to respond to, often times, contradictory feedback. In short, we never got to the one great character we wanted and this hurt us both in lost time, team morale, and publisher support.

As time went by it was becoming increasingly critical to get major features done. We had been working on them all this time, of course, but they were starting to become the major items on paid milestones. It was at this point when we began the difficult process of supporting existing features and tools, trying to make the one fun level and completing major features in line with our development schedule. This created the additional stress of requiring engineering to do support during the day and feature development at night (you don't want your engineers implementing important features when they are exhausted, trust me).

The overall issue is that we were not able to make the one great level, character, vehicle and so on in the time allotted, and this should have resulted in re-evaluating the schedule. Something worth mentioning is that we knew that we should be working on the one great level and there was almost universal consensus on this. The problem is that this consensus did not translate in to actual contract and schedule changes. There are many reasons for this. Primarily if your contract and schedule are designed to be fairly rigid and the expectation is to follow them in a somewhat precise manner, then schedule changes will be an undesirable and difficult procedure, which means that they will be avoided. Instead, change should be expected and delays in schedule implemented at the earliest possible time.

When you are a third party developer there is always the additional fear of your project being cancelled. On several occasions we had a very real sense that the project could be cancelled and occasionally had to work on specific things in order to get the confidence in the project up. In many cases these tasks were not necessarily what we should have been working on, from a product development point of view; but the fear of your company going under and laying off dozens of people can be a strong motivator.

Both NetDevil and NCSOFT were in agreement that the quality of the game was not high enough and that we should be iterating on small amounts of content prior to going into production. This instinct was correct, but did not translate into action. Of course there was a concerted effort to make one great map, but it was watered down by the enormous amount of other work that was technically a higher priority, because those were the things that were on the paid milestone list. It would have been much better to focus on doing ONLY one great map and not working on more features and content until that was realized.

2.4. Features, features, features...

Auto Assault has a lot of features; perhaps too many. Some of them were critical and obvious, others were more obscure.

We had always planned on Auto Assault being an item centric game, so we naturally spent a great deal of time on the item creation system. Most MMOs did not have dynamic loot in the same vein as Diablo and Diablo II and we felt this was a critical feature. The object generation system in Auto Assault is something we still feel is something that was well executed in the game.

In addition to items, we wanted to have a robust and complete skill system in order to help distinguish the various classes and races in the game. We began this process by trying to define each race as having strong types. For example, the Mutants were more in tune with nature and so their summons would be more organic and they would use viruses and so on. This worked out quite well but one mistake we made was that we decided how many skills we needed based on number of levels and classes. We figured players want to have this many new skills every so often and that would dictate the number of skills. We now believe what you should do is make skills that are distinct and complete; that is that each skill should serve a particular purpose and should not be duplicated by another skill. Once again we went down the path of making all the skills on the first shot rather than making one or two and seeing if they worked, were fun and so on. I hope you are beginning to see a theme here; do not make lots of anything until you have one that is really great. Ever.

Another major feature of the game was the physics system. We wanted to have destructible environments and tons of creatures to fight. This proved to be quite a challenge and we spent a lot of time making the servers perform well enough with upwards of 5,000 objects moving around at 33ms/frame. This is another area where we accomplished something really well.

In terms of content we wanted to have a more directed experience than most MMOs. This meant we wanted to have a complex mission system which allowed designers to have great control over events. We devised a fairly open and complex trigger system which allowed designers to rig up almost anything they could think of. In addition to this we had a fairly complete mission system which allowed designers to define mission flow, prerequisites, rewards and so on.

In hind site, we failed to focus on some basic issues. For one thing we did not have a dedicated tools programmer so when designers came across bugs with the trigger or mission systems it would become a critical task which would then interrupt someone working on a feature. This happened with great frequency. Many times the designers would figure out work-arounds and occasionally exploit bugs to implement desired features. Then when a programmer fixed the bug for someone else, it broke some existing mission or trigger chain somewhere. Additionally, without a dedicated tools programmer there was no easy way to debug what was going wrong within a complex series of triggered events. This problem could have been handled better on both sides. You need to have great tools that allow for rapid iteration, and you want to have tools sufficiently complex enough that they can create a compelling enough world without making it impossible to predict what will happen.

Tools are a tricky business. One the one hand you need tools in order to build a compelling game; on the other hand you do not want them to become golden handcuffs. If you spend 3 years

making tools only to find out the tools make the kind of game no one wants to play, you will be unable to admit that you made the wrong tools, because you can not just throw away 3 years of work! The tools need to support what is great about the game, but they need to be supported in a dedicated way. It is important that before ramping up production, that the one great map, enemy, skill etc. was made with the tools that are going to be used to make the 1 gazillion other maps.

Even with all these issues, at this point we had never missed a milestone. We had always managed to check off all the issues and although, with some hesitation, the publisher agreed that we had met our deliverables. The problem is that no one was playing the game. We had some levels, skills, items, missions and so on, but there was no game per se. A few things contributed to this. Auto Assault had performance problems: the game would have frame rate drops, hitches when assets loaded, glitches when players transferred and huge memory requirements. It took enormous video resources without looking amazing. We knew all of these things, of course, but assumed we would “polish at the end.” This is another opportunity to debunk this myth. You will not polish at the end and if a feature or asset or mission or whatever is being added and it affects performance, it is not complete. Performance is part of any content or features not an afterthought that will be dealt with later.

I want to take a moment and address this from the other side. Our reasoning for dealing with performance issues “later” was that we did not want to delve into optimizations for systems or methods that will change anyway. It seemed to make more sense to finish features up, THEN go back and make them run smoothly since we would be better equipped to handle the most important performance problems first. This is logical and reasonable and completely wrong. While you do not want to spend 20 hours seeing if you can optimize that for() loop, you do not want to ignore the fact that level 20 makes the frame rate drop to 2fps when you turn the second corner. It is a tricky balance but it is a simple rule. If you can not tell it does not matter; if you can tell, it can not be ignored.

There are two problems with this logic. First of all, you never have time at the end. “Polish at the end” translates to “we’re not going to polish.” Also, if the game can not be played because of performance how will you ever know if the game is fun? Answer: It is not fun. There is no such thing as a fun game with bad performance.

Another excuse I often hear is that you can not play your game because you are too close to it, or that because you are working on it, all you will ever see is problems. This simply is not true. If you do not play your own game, why would anyone else? Imagine if you went into the Windows development group at Microsoft and they were all using Macs running OSX; how confident would you be that Vista is a good operating system? Imagine if everyone at Ford drove a Toyota; how would you feel about Ford vehicles if their own employees will not drive them?

We actually began play testing at this point, and there were a few things which happened that we did not identify as major issues. People tended to say “it’s good, but it’s not my kind of game.” This generally means that you are not going to have many customers. When people say this they are telling you they are not

going to buy your game; and if enough people say it you can guess how many copies you are going to sell. Another thing that happened is that people kind of liked the game but quickly stopped playing. There was “something wrong” but it was not easily identifiable. The tendency is to look for specific reasons why someone does not like something and if they can not articulate the reason, to disregard it. This is a huge mistake. Just because someone does not know WHY they do not like something does not mean that their dislike can be disregarded. The goal of a mainstream commercial product should be that most people who try the game like it.

Another common mistake is to think your game is too easy when, in fact, it is too hard. This was the case with Auto Assault. Since we were working on the game, we knew how it worked and how to successfully play. There were a few obstacles facing Auto Assault that other MMOs did not face. For one thing, people do not inherently know what items types do. For example, most people understand that leather armor is lighter but weaker than plate armor. No one instinctively knows whether nano-armor is lighter or heavier or weaker or stronger than light-photon armor. Auto Assault also required a mix of driving skill and RPG knowledge. These things are possible to overcome but everything else in the game must be very simple.

Here’s an example. One of the things that is really fun with driving games is running enemies over. In order to make this more interesting we created a system by which NPCs avoid your car so that you can chase them down and run them over. This seems fine, but is, in fact, exactly wrong. Players just starting out are barely able to drive, they do not have the ability to drive, shoot and aim. Instead, the NPCs should try and put themselves right in front of the car so that the player has the satisfaction of running them over.

The lesson here is that you do not have to do the same thing at the beginning of the game as you do at the end. It is fine to have a crazy fast NPC that disappears and throws fireballs while laughing at you. It is not fine to have him as your level 1 encounter. The game should ramp up from “so easy I want to kill myself” to “no one in the office can beat this guy.” The same system may not be able to solve the problem for the beginner experience and the super insanely hard experience. No problem. Build two systems!

The combination of creating new features, supporting content development, performance issues and balancing the game prevented us from getting the game to be great before we had to start creating all the content the final game would have. At this point it most likely would have been a good decision to delay the game and address the issues before going into the content development phase. It is also possible that we could have cut some features at this point in favor of improving the quality of what was there. There are many ways things could have been done differently, but we are now convinced that going into production prior to having a small, isolated area that is fun and complete is a mistake.

2.5. Content and Iteration

MMOs are generally big games. People expect hundreds, if not thousands, of hours of game play. In order to accomplish this task

you need an army of writers, artists, designers, QA and programmers to support the content pipeline. Additionally, before generating the massive amounts of content required, the tools should be done, the game should run well, iterations should be quick and so on.

At the time that we began real content creation we were still working on tools and features. Also, the game did not have “one great level” nor was everything fast to iterate on. Some things worked very well. The servers performed well, we had automated testing tools, the client was relatively stable and the content pipeline was pretty solid.

Because the game did not have one fun level and we were in heavy content production, each time we discovered some fundamental problem we had to redo all the content again. This proved to have a devastating effect on morale and efficiency. People were tired and frustrated with iteration cycles and this leads to an inevitable drop in quality.

We had never made a content heavy game like this before and thus underestimated the amount of time and people it would take to generate enough content to create a compelling MMO. On a few occasions we would be working on 3 maps at a time and then throw them all away and start again. Instead it would have been better to pay the price on one map, with a smaller team until we knew what made a map fun.

It was at this point that we started to miss milestones. It was effectively impossible to develop new features, support massive content creation, asset pipelines and do bug fixing. As a result, both content and functionality milestones became unachievable. This put enormous pressure on the developer as well as the publisher. There was a growing sense that we would not make our original release date and the ever present fear of being cancelled after so many years of hard work.

To the publisher’s credit, they did not cancel the game, nor did they lose interest in it. In fact they stayed behind us 100% and were willing to take the time and spend the money to make it work. As a result we ended up slipping. This, of course, has two immediate affects. On the one hand, as a developer it now becomes less likely you will ever profit from all that effort. Also, people have the feeling of relief that they can relax a bit, but also tend to feel like the development will never end. These two factors contribute negatively to efficiency. The other thing is that at this point the game is delayed when it is most expensive to do so. It would have been far better to delay during the prototyping or prior to feature completion when the team was 10-15 people, rather than when the team is 40+ people. It is possible that if that had happened the game may have been cancelled as well. It is impossible to know what decision will lead to which outcome, but there is no question that it would have been better to delay earlier rather than later.

Outside of the company, the effect on the community was also negative. Up to that point we had reasonably good buzz happening. People were getting excited that the game was coming soon and the press was anticipating the release. Once we slipped this sense of community dried up fairly rapidly, and we were never able to get it back to pre-slip levels.

In order to get to the point that we could actually ship we began to cut content. I feel this was the correct decision as it is better to

less better content than more average content. At the point of delaying the launch we did, in fact, have the game finished. The maps, missions, loot, vehicles and so on were all in the game, but things did not hang together the way we wanted them too, and things still did not seem great. At this time we created “strike teams” or “cabals” or whatever you choose to call them, in order to focus on very specific parts of the game. This process ended up working very well and we were able to make significant improvements in shorter periods of time. Unfortunately, engineering was still dealing with performance problems and as such was unable to give the support needed to address the result of what the strike teams found. Once again, however, we were measuring what we had with what came before, as opposed to comparing against outside competition, and this can lead to some amounts of delusion.

In the end we were able to make some really interesting content, good missions and creative story lines; but the large cost of iteration and constant refactoring as a result of not being feature complete cost us dearly.

2.6. Beta Testing

It was at this time that we began beta testing. Beta testing is an exciting time as it gives the team a sense that the game will actually be finished soon, as well as valuable feedback from the community. We misunderstood what beta tests are. Most people tend to think that beta tests exist to test the playability, stability and fun of the game; they are not. Beta tests are marketing tools, whether you want them to be or not. Once the game goes beta, people begin to evaluate if they are going to buy your game or not. Even though you say that you are “still in beta” people will say “this game sucks” or “this game is cool” based on the beta... and they will tell all their friends as well.

Do not beta until your game is basically done. The only thing the beta test should test is scalability of your servers and how robust your deployment system is. Auto Assault went into beta too early. It prompted us to delay the game, which was the correct decision; however the damage had already been done. The game had been judged by many beta testers and it would be almost impossible to get them to come back and judge again. Make sure that your beta test contains as few people as you absolutely need; and avoid using beta to test major features, fun factor and ease. High end PvP balancing, server load, login process; these are what the beta test is for. Your game should be fun, complete and play well before you go into beta.

I believe that Auto Assault went into beta too early and we never completely recovered from it. The last few months of development were largely positive. The game was finally beginning to come together and there was a strong sense that the game would release. We were internally nervous that the buzz was not as high as it should have been. It was reasonably clear that the game was not going to be a massive hit, but we still hoped that we would get a reasonably sized “niche market.” This is another red flag. It is reasonably difficult to anticipate the size of a “niche market;” especially for a game that has a somewhat unique (that is unfamiliar and uncomfortable) game mechanic.

2.7. ...and finally... Release

Auto Assault was released in April of 2006, after almost 4 years of development and an incredible amount of man hours. We had a

relatively small release party and people were generally relieved that we had finished and we eagerly awaited how the game would do.

As releases go, Auto Assault was not too bad. There was the usual large “day one patch” which players tend to hate; but the game pretty much worked, there were not massive database issues, no horrible server crashes and so on. There were also not many players. We started with 8 shards and should have started with fewer. We should have kept most of them offline until the load on one or two got too high as they would have been relatively easy to activate. Instead we had 8 shards that were all relatively empty; and people do not like to play in an empty MMO.

Over the weeks that followed it was clear that the game was not going to meet expectations. This put us into a state of total darkness. At that time NetDevil did not have another development contract and we did not have a long term operating contract. In short, people started looking for jobs and after so many years of hard work, anticipated being laid off. Due to the financial situation of the game, in the end we were forced to scale the team back. Publishers need to make money and as such this was inevitable, but it was painful and difficult for everyone. Fortunately our publisher remained communicative, optimistic and supportive. They could have been much nastier and hostile; but it was a surprisingly positive relationship where both sides were trying to make the best of a difficult situation.

2.8. ...and Post-Release

Post-Release is a tricky time for an MMO. Things tend to happen quickly and generally the team is relieved and excited at the launch but also somewhat tired. We actually had a relatively smooth launch without any major incident. There were the normal server and database issues, but nothing that ended the world.

Something we had done earlier that we could now take advantage of was stat tracking. We were able to capture almost all events in the game: quest completion, enemy kills, even each skill cast, weapon firing and so on. It was a ton of data. Being able to measure player activity proved an invaluable way to determine what should be worked on first. Our metrics indicated that we had a lot of user drop off at early levels and as such we spent a fair amount of time getting the first few minutes of the game better. We made the enemies easier, increased loot drop, slowed the vehicles down a bit and so on. It was really great to be able to change things and then see the effect.

It is not effective to ask people what they think in order to make these decisions. Good metrics are critical for determining where problems in the game are. You can not rely on gut instincts, you can not rely on asking people what they want, but you can measure what they do and how long they do it.

In addition to metrics there are some cases where you can take them seriously. A good indicator of this is when there is almost universal agreement, especially between the press and the players. In the case of Auto Assault there were two areas where this was true. One was the lack of an auction house. World of Warcraft had made the auction house a staple of gaming and since Auto Assault was an item heavy game an auction house seemed to be glaring omission. In truth it was a feature we had always planned, but got postponed in the last few months of development. In any case

there is an auction house in the game now and it is a great example of where public opinion was almost unilateral.

Changing things when the game is live is also a dangerous proposition. An exploit being taken advantage of by 5% of the people (or less) may have to be fixed in such a way that the other 95% are also penalized. Knowing when to do it and when not to is as much an art as a science. For example, in Auto Assault, skills have been tweaked and balanced numerous times (as in almost all MMOS). Players will tend to ask for drastic changes and it is important to do these things gradually so as not to destroy the game. In my experience the best indication that you are doing well is when all sides are pretty much complaining equally. If all groups agree that something is overpowered, boring or underpowered, it may not be the case that it is, but it is a strong indication that SOMETHING is wrong and you should take a look.

At post release is when choosing the correct publishing partner really pays off. NCSoft has a great deal of experience in developing, running and deploying MMOs; and this is a big part of a smooth launch. Also, they inherently understand the nature of the publish process for MMOs and the risks associated with it. The most important thing is that a good publishing partner understands that MMOs are a service as much as they are a product. You have to keep the game going and keep growing it and improving it. From day one we had dedicated QA, GMs, bandwidth, a live team and so on. This is an area where NetDevil and NCSoft always saw eye to eye and it is a major reason that we still have a great working relationship and continue to work on making Auto Assault better and better.

3. Conclusion

Auto Assault was a large and complex game to create. It involved breaching new ground in many areas while trying to maintain the things that MMO players expect. In this regard it was also a very risky game to make. It would have been safer to make a fantasy, character based MMO in the tradition of Ultima Online and Everquest, but we chose to try and go into a different direction.

Creating any game requires the hard work and dedication of many people, both on the developer side and the publisher side. Too often the relationship can become adversarial; which is mostly due to lapses in communication. In the end we are all working together to try and make a fun and compelling game experience, which results in financial success. Even after all the difficulties, NetDevil maintains a positive relationship with our publisher and personally we all get along very well.

We have learned an enormous amount from this experience and hope that it serves us well in our future game efforts. We also hope that others can learn from our experience and use them to make their own games better.

Here's a quick recap of some of the things we learned:

- **Polish as you go.** Do not wait until the end to polish your game, there will not be time.
- **Make the game first.** This is something that I think we did mostly right. We had a prototype relatively quickly and it gave the publisher great confidence and allowed us to see where the game was going. Also, it is easy to throw away less work than more work.

- **From one. Many.** Make sure you have one great and fun level that everyone can play and enjoy. Delay if you have to, but do not start making lots of content and features until you have one great area.
- **Good Enough = Dedication to mediocrity.** It is tempting to look at your game and think “Ok, it’s better than what we had before.” This is a lousy benchmark. You should strive to be better than anything else in the same vertical. Period.
- **Embrace change.** Your development schedule must be set up in such a way that you can change direction quickly and easily; and most importantly without major financial penalty. It will all come back to money; and at the end keeping your business running will be more important than anything so make sure that the money and game interests are the same.
- **Play your game!** I can not emphasize this enough. If people in your office are not playing your game, people outside will not either. Do not accept excuses such as “well, we’re burned out on it” or “it’s just not my kind of game.” Those should send off huge alarm bells in your head. Do not ignore them. Forcing people to play in the beginning is a also a good way to put focus on what is most important.
- **Beginning != End.** Just because something works for the beginning does not mean it is good for the end, and vice versa. You do not need to solve the problem for the ENTIRE game... it may very well be that one solution works in one place, but not in another.
- **Performance + Easy + Pretty = Fun!** This is, of course, grossly simplified, but in general if your game looks good, plays well and is easy to learn, you have removed most of the barriers to fun. People start your game wanting to have fun, believing they will have fun. Don’t smash them over the head and kick them in the teeth by killing them, delivering crappy frame rate or not telling them what to do.
- **Sufficiently Complex...** and no more. It is a good exercise to reduce things to as simply a state as is still sufficiently complex. Do you really need a skill that adds 10 damage, increases armor, and turns you invisible every 5 seconds for 2 seconds over 2 minutes?