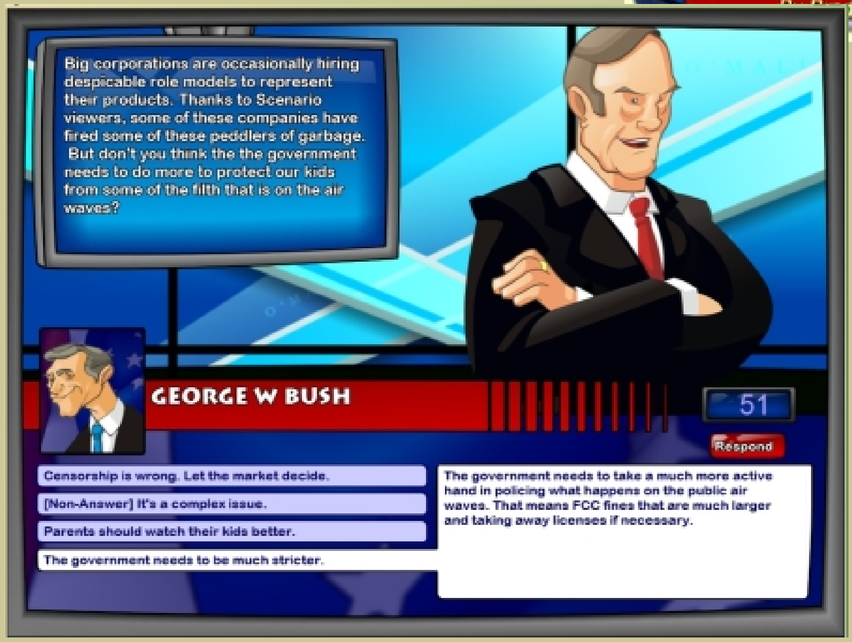
Postmortem: Stardock's *The Political Machine*

By Brad Wardell

The Political Machine is a PC strategy game in which players take on the role of campaign manager of a Presidential candidate. Players can either manage real-world candidates (Bush, Clinton, Kerry), historical candidates (Reagan, FDR, Lincoln, Jefferson), or create entirely new candidates from scratch. Players then compete for the 270 electoral votes needed to win the Presidency. Candidates can take out ads, make speeches, raise funds, win endorsements, hire operatives, and even go on cable TV shows such as "The O'Malley Factor", "Hard Hitter", and so forth. Even if you're not into politics, we tried to build in strategy game mechanics that everyone would find solid, and we're happy with the final game that has resulted.


Big corporations are occasionally hiring despicable role models to represent their products. Thanks to Scenario viewers, some of these companies have fired some of these peddlers of garbage. But don't you think the the government needs to do more to protect our kids from some of the filth that is on the air waves?

GEORGE W BUSH

51

Respond

Censorship is wrong. Let the market decide.
[Non-Answer] It's a complex issue.
Parents should watch their kids better.
The government needs to be much stricter.

The government needs to take a much more active hand in policing what happens on the public air waves. That means FCC fines that are much larger and taking away licenses if necessary.

Cable TV shows such as "The O'Malley Factor" allow you to express your stance on hot button issues and pander to the public in *The Political Machine*.

That said, the game was a huge departure for Stardock. Most of all, it was the riskiest project we've ever undertaken. In order to have the game available at retail at the exact right time (so that the first 90 days after release were during the peak interest in U.S. presidential politics: second week of August to second week of November), we would have to finish the game by June. No "it's done when it's done"-type philosophy would work here. It was all or nothing. Miss the ship date and you have a very expensive coaster.

As if time constraints on finishing the game weren't enough, there was the fact that the game had to be rock solid out of the box. So even if we wanted to take the "we'll fix the game with a patch post-release" route, we couldn't, because by the time the patch came out, the election would nearly be here. In effect, we had an AAA console-type quality requirement on a PC shareware-type schedule.

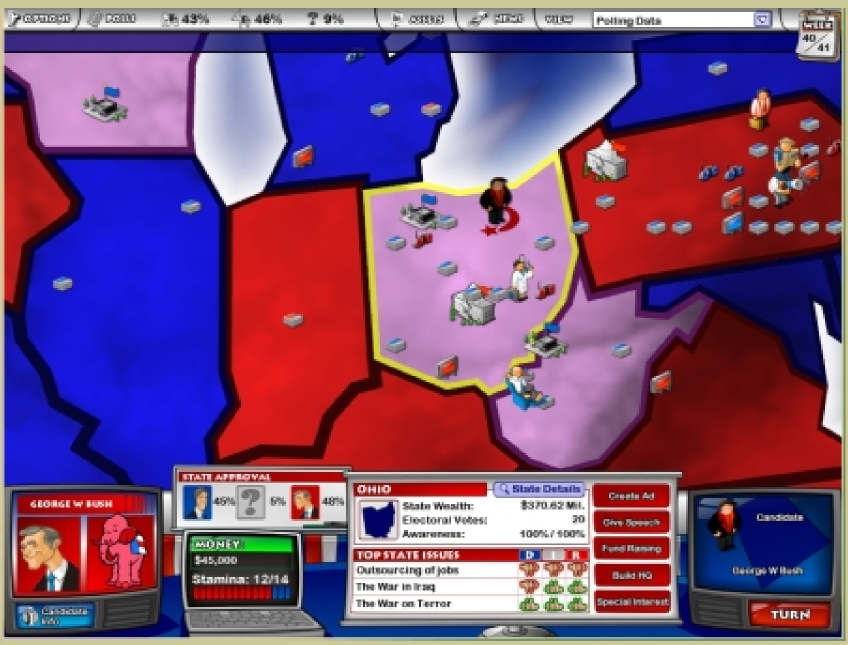
But to top all that off, there is the fact that no one has ever tried to make a political strategy game aimed at the mainstream (there are election "simulator" games out there for the hardcore, but there's nothing that's designed to be on the shelves at Wal-Mart). Would people want a game like this? Could we make a game like this "fun"? Could we make a game like this even remotely accurate?

What Went Right

Flexible Design. We started development of this game in January 2004. That gave us *6 months* to write a 3D engine for it, come up with the screens, the gameplay, multiplayer, and, of course, put together the statistical model and computer AI. In January we had a vague idea of what kind of game we wanted. We pictured it would be a kind of *You Don't Know Jack* for politics with a map. The final game, for those who don't know, is nothing like that. The design we started with doesn't remotely resemble the final game, despite the fact the title was sent to manufacturing right on time, 6 months later.

This flexible design extends to how our developers worked. We put the artists and the developers in the same room. and so they worked out more efficient ways to work together. For example, because we were so short of time, we needed a way to be able to crank out interface screens very fast. In a correctly managed project, screens are mocked up, gone over by the producer, marked up, and then submitted to the development team for implementation. We didn't have time for that process.

So, we used Stardock's own Windows desktop object creator/manager, DesktopX to create the interfaces. With that, the mockup was the interface. If we wanted to make a tweak, we could do it on the live interface (fonts, positioning, sizing, color, etc.). We then exported our interfaces as DesktopX .dxpacks, which we then read in the game and use. If someone didn't like a screen, no problem, they could tweak it themselves. Without this, we wouldn't have been able to finish the game on time. This wasn't planned, by the way. In fact, we were desperate, and our lead graphics designer (Paul) and lead engine developer (Cari) came up with this on their own, to solve the problem of needing the mockup to have live interface functionality.



The 3D engine allows seamless zooming-in on individual states.

Using DirectX 8/9. Microsoft deserves some credit for the changes it has made to DirectX, making it possible for us to easily combine our 3D engine with 2D elements. This is good news for us, and bad news for those who make 3D engines for games that they license.

Now, what's the big deal about DirectX 9? Well, the biggest thing that we got out of it was the ability to have TrueType fonts. Most games have to include their own encoded fonts. But with DirectX 9, we were able to just include and use TrueType fonts, which saved a lot of time, and made GUI elements look very attractive with little effort. It also allowed us to mix sprites and 3D elements much more easily.

For example, in our last game, *Galactic Civilizations*, there is no zooming in or out of the map. That's because it's a 2D sprite-based game. But with *The Political Machine*, you can zoom in and out on the map all you want, and it's very smooth and fast, even though the actual units are just sprites. Also, thanks to 3D acceleration, you can play the game on amazingly low-end hardware. My old Pentium II 400Mhz running Windows 2000 plays the game fine, even though it just has an old TNT graphics card in it.

Tools. This was the first game where we actually made tools to create the data. Anyone familiar with Stardock games knows that our "data" is usually just a collection of text files. But because of the complexity of the political issues and interview questions from the cable TV shows, we needed to create some tools. This made us able to create much more sophisticated interaction with the TV interviewers than would have otherwise been possible.

The tools were integral to the game because they let you focus on the content, rather than worrying that making a single typing error in a file would cause the game to crash. When you develop tools to create your content, you can then control the way the data is presented in a much more considered fashion. For instance, in a game like The Political Machine, it's vital that the game be equally fair to Republicans and Democrats. The tools would let us visually see how a given issue was tilted much more easily, compared to sifting through text files and looking at raw data.

New Bug Tracking System. Bug tracking at Stardock has been a long and questionable journey. We've tried out various systems and always ended up back to a spreadsheet. This time, we settled down with FogBugz which has worked terrifically for us.

In the past, we just used spreadsheets for our bug tracking. This works to a degree, but it's not as efficient as something like FogBugz. With

FogBugz, our beta testers could email in a bug report, and it would automatically show up in the program. Then the project manager could go through and assign priorities, and the team could then jump in and see what they needed to work on. It's important to stress again - we started/finished this game in 6 months, which included development time for the 3D engine, all the artwork, and the multiplayer. We had to be very efficient in how we did it.

Good QA from Ubisoft. Ubisoft really provided great QA, and were very easy to work with. They were absolutely essential to the game's success because of their greater experience in dealing with problems of 3D engines. Their clout helped too. When we had problems with ATI or Nvidia cards, Ubisoft could ring up someone at one of those companies and get a response to help us with.

So what didn't go right? Well, a lot of things nearly ruined us along the way.

What Went Wrong

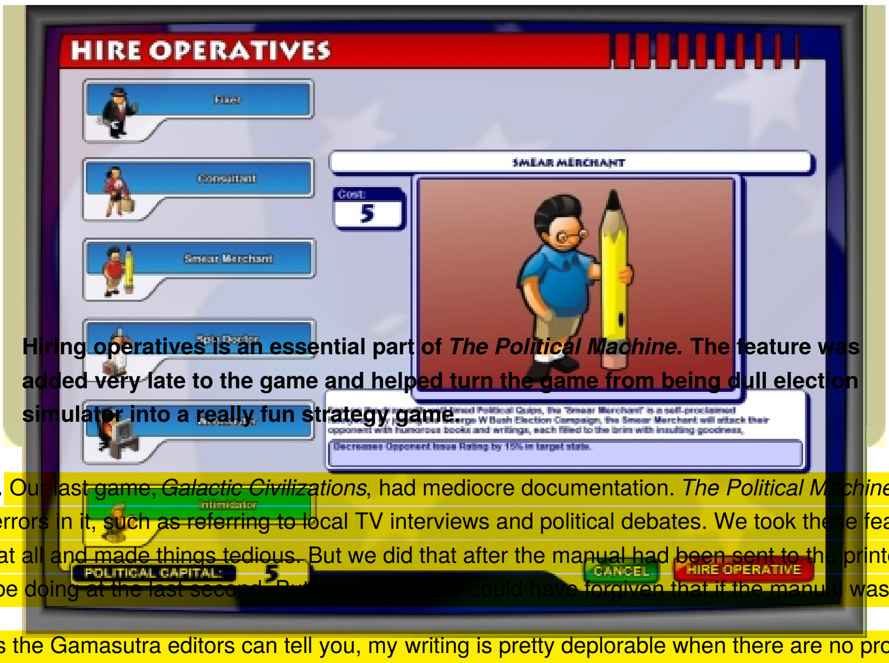
Play Testing. On May 1, 2004, a little over one month before being sent to manufacturing, *The Political Machine* sucked. It wasn't any fun. There was a cloud of despair hanging around the offices. We didn't know what we were going to do. So we started brainstorming, and lots of good ideas came out. What's particularly startling is the fact that these ideas I mention below all were devised, implemented and bug tested in 30 days:

- Political Capital. You can build political capital to spend on various political favors.
- Endorsements. Political Capital could be used to win an endorsement from a special interest group (Christian Confederation, NCLU, and so on.)
- Operatives. Smear Merchants, Spin Doctors, Intimidators, Fixers, Webmasters, and Consultants were all thought of and implemented, each with their own power.
- Political Campaign. In this mode, players could choose a political party and work their way up a ladder of increasingly difficult opponents.
- Multiplayer. We were not originally going to put in multiplayer, but instead have a 'Metaverse', where people could submit their single player scores to a central database. I might add that no one on the team had ever done a multiplayer game except myself, but I didn't program the multiplayer part of that title.

So, I kid you not, all of these features were added in about 30 days. But why is this in the "what went wrong" category? Most of it nearly didn't get play tested. The guys got so enamored playing each other at multiplayer that no one had, for example, tested to see if you could actually *win* the campaign. Fortunately, in one of the last release candidates, I sat down and started to play the campaign for fun. And as the guy who coded the computer AI, if I can't win the campaign, nobody could. When I got up to George Washington to play, he won every state. I didn't even win a single state. He smoked me. So over the course of that weekend we began massive play testing to make sure the game played how we wanted. We then submitted to Ubisoft a new release candidate that Monday (someone from Ubisoft is probably reading this thinking: "Oh god, I didn't want to know that!")

Insufficient compatibility testing. When we decided to use DirectX 9, we assumed that was fine. We had no idea that the game would have compatibility problems with some hardware configurations. Our view was that, as long as we followed the rules, we would be fine. We were wrong. Firstly, the main reason we used DirectX 9 was to get access to easy TrueType font support. But it turns out that the older integrated Intel video cards don't support that. So essentially, we have a game that doesn't work on those machines. There's no "workaround" for it, other than us scrapping TrueType fonts and going back to encoding fonts again, which we don't have the budget for (especially since this only affects one type of video card).

But on a game that targets "casual gamers", you have a high number (in absolute terms) of people who have these low-end video cards, and who don't understand why a game that seems to have relatively mild requirements wouldn't work on their system. This results in very mad users. Unfortunately, mad users don't just return games, some of them go to review sites and take out their anger there. It only takes one angry guy giving a game a rating of 1 out of 10 on a game reviews website to really mess up the overall rating. Even though this is technically Intel's fault, the user doesn't care. If I had to do it over again, I would probably have required us to put in some sort of "fallback" mode with basic font support, so that the game would work on those Intel machines to a certain degree. Alternatively, we could have pushed Intel to fix their drivers before the game's release (the newer integrated Intel chipsets--or at least, those that are less than 2 years old - work fine).



Hiring operatives is an essential part of *The Political Machine.* The feature was added very late to the game and helped turn the game from being dull election simulator into a really fun strategy game.

Horrible Documentation. Our last game, *Galactic Civilizations*, had mediocre documentation. *The Political Machine*'s documentation is even worse. First, it has gross errors in it, such as referring to local TV interviews and political debates. We took these features out of the game because they weren't fun at all and made things tedious. But we did that after the manual had been sent to the printers. Needless to say, these are big changes to be doing at the last second. But users probably could have forgiven that if the manual was better.

I wrote the manual, and as the Gamasutra editors can tell you, my writing is pretty deplorable when there are no professionals such as themselves to convert my writing into something readable. *The Political Machine* had little editing, since there just wasn't enough time. Next time I'm going to find someone to work with me early, in order to create a good manual out of my terribly written design documents.

Excessive Initiative. The game succeeded because we had good personal initiative by the development team. But sometimes, developers would sneak in a feature without passing it by the rest of us. A windowed mode was added in Release Candidate 2 without anyone knowing. 4X anti-aliasing was also added without telling anyone. These were all good things, but imagine if there'd been some terrible bug as a result?

This also led to people having their own styles of coding. Stardock is growing quickly, and our programming guidelines aren't that specific. So we had internal battles on whether to use CString classes or just plain oh char szMyString[255];. We also had issues resolving just how functions should work. (Typically, it is against Stardock coding guidelines for functions to modify a parameter. They should always return a new value at which point a variable can be updated.)

Multiplayer. The biggest regret I have, in hindsight, was the decision to have a multiplayer mode in *The Political Machine*. The game features a full-blown matchmaking service, in order to make it relatively easy to play multiplayer. I love playing games multiplayer, and I've played a lot of games online with people. That said, based on the sales statistics, and based on the server stats, less than 1% of players are playing the game multiplayer.

As a gamer, I demand multiplayer in my games. It affects my purchasing decisions. But as a game developer, I recognize that people like me are an extreme minority. Outside of a handful of games, most games don't reach critical mass in online players to make a successful multiplayer community. We would have been much better off putting that time into enhancing the single player game.

We had several other features in mind that we had to cut because of lack of time, time we would have had if we hadn't done multiplayer. For instance, we couldn't include multiple maps, and I'd love to have had the ability to play in Canada, or the UK, or the whole of Europe, or even a worldwide map. That's not even counting made-up maps we could have created. I think the inclusion of different maps to play on would have increased the replayability of the game a great deal.

I would have also liked to have given different candidates "special" powers, to make each one more unique apart from statistical differences. But the time in development and testing on multiplayer eliminated those kinds of things. When gamers demand multiplayer, what they don't realize is that something has to be sacrificed for it. You have a finite budget and a finite time to use it, and so, to quote Spock: "The needs of the many outweigh the needs of the few." With that in mind, I wish we hadn't done multiplayer in *The Political Machine*.

The Politics

In a postmortem, the section dealing with what went right or wrong is usually the main part. But there's more in this particular game. In a political strategy game, especially in a hotly contested year such as this one with Bush vs. Kerry, we had to put a lot of effort into making sure the game was fair to both sides. People would be looking for bias in the game, and probing for any hidden agendas. We wanted the game to be accurate enough to the real world that political junkies wouldn't be turned off, but we wanted to also make sure it was a fun game. This is a *strategy game*, not a simulator. We think that the game is reasonably fair to both sides (or equally unfair, if you prefer).

But to make it fair, we had to do a lot of research. We had to look at census data, polling data, exit polls, and watch a lot of cable TV news. I can tell you the pet topics of pretty much any cable TV host, even those not in the game. I can tell you that Fox News isn't particularly conservative, and I can also tell you that there isn't a *partisan* bias in any of the cable TV shows. They focus on the issues, and some issues favor Democrats and some issues favor Republicans. But the host's preference to discussing a given issue doesn't make them partisan on its own.

Overall, we're very happy with how the game turned out. But we'll never ever do a game under these kinds of deadlines again. That was insane. Now, if you'll excuse me, I have to get back to working on *Santa Saves Christmas* which has to be done by… ah, never mind!

Game Data

The Political Machine

Publisher: Ubisoft
Developer: Stardock Entertainment
Budget: $200,000
Number of full-time developers: 4
Number of part-time developers: 1
Number of contractors: 1
Length of Development: 6 months
Release Date: August 10, 2004
Platforms: PC
Development Hardware: Various PCs
Development Software Used: Microsoft Visual Studio and DesktopX
