

## Game Prototyping

### Lesson 1 Game Postmortems

The following are selections from various game “postmortems” (literally, “after death”). That is, when a development team sits down after a game is published to determine what worked and what didn’t to improve their future games; often the game’s **prototyping stage** is discussed. This collection of game various game postmortems written by the development teams will help you understand the significance of game prototyping.

## Guidelines for Developing Successful Games

### **Prototype Early**

Prototype all-important systems and technologies to proof the concept as early as possible. Prototyping is not only useful from a technology standpoint, but is also critical for testing gameplay. Designers are usually left guessing until their games can be played. There are always surprises when a game is first played, some good and some bad. Prototyping for gameplay testing is especially useful for strategy and other empty map games that do not depend on pre-planned or linear story lines.

### *Rise of Nations* from **Big Huge Games** What Went Right

**Prototype method of game design:** Part of the core vision for *Rise of Nations* involved introducing gameplay innovations inspired by our experience making turn-based games into the “classic” real time strategy mix. We had 10 to 15 “wild” ideas about what might take real-time strategy in new directions, but we knew that only some, maybe only a small few, were going to work, and we didn’t know which ones. It was essential that we find out as soon as possible which ideas were worth implementing, and we knew from experience that the only sure way to accomplish this is to throw the ideas into a playable prototype right from the beginning.

We got a playable solo prototype running within the first month, and a fully playable multiplayer version more than two years prior to ship. We could throw new ideas in and see the results almost immediately: some concepts needed a little tweaking to be fun, while others got trashed almost as soon as they went in. The value of prototyping is that core concepts end up being continuously refined over years, while providing lots of time to balance the game.

As part of the prototype approach to design, we make sure that everyone in the company is playing the game on a regular basis. After each daily play session, a member of the design team compiles everyone’s feedback and sends a summary to the rest of the designers. However, we found that we had to be willing to wade through some resistance to new features or gameplay tweaks. People would get very attached to certain strategies for playing or even just conventions of RTS games of which they were hesitant to let go.

## *Neverwinter Nights* from **Bioware** What Went Wrong

**Incomplete prototypes:** Even though we put a lot of effort into prototyping important game systems, on some occasions we completed what we thought were full-featured prototypes of major game systems only to find out later that they didn't address a number of important issues. In our haste to get into full production on *Neverwinter Nights*, we didn't properly analyze all of the questions that needed to be addressed by the prototypes. This resulted in spending time late in the development cycle sorting out problems with key systems of the game.

Development of our new game engine was an extremely long process; as a result, some of the initial prototyping lessons were forgotten or inadequately documented. In some cases, we didn't thoroughly review our original goals when implementing features later in the project.

As with any new-engine game, there was too little time available to prototype gameplay. Our prototypes focused instead on technology and the individual features of the game. While this kind of prototyping was important, it would have been very useful to have early feedback on how the game *played*, particularly with regard to the interface and story line.

Because NWN was a rule-based game, and rules implementation was at the end of the schedule, we were only able to test actual gameplay near the end of the development cycle. Due to our inability to prototype a number of design components, we ended up reworking them. As a result, we plan to prototype story lines in future games earlier in the development cycle. One of the ways we plan to do this is to reuse the BioWare Aurora Toolset as a rapid prototyping tool for story design, even for games with radically different interfaces and rules systems.

## *Ratchet & Clank* from **Insomniac Games** What Went Right

**Prototyping:** We had been prototyping gameplay since *Spyro the Dragon*, but never to the extent that we did with *Ratchet & Clank*. The game featured more than 35 weapons and gadgets, all of which had to be fun to use. The big problem we faced was that every weapon and gadget was woven into the macro design and the story. If we had to pull one out during production, the macro design would collapse, which would be disastrous for the production schedule.

We spent three months building and programming the weapons and gadgets. Many of them didn't survive the prototype phase because even though they sounded good on paper, we just couldn't make them work. A good example was the Revolverator, a weapon featuring a large drill bit which would spin enemies around and fling them away. We discovered that the spinning slowed down gameplay, and that it was difficult to hit enemies, since the collision for the drill bit had to be narrow to be believable. Another good idea on paper was the Mackerel 1000, a fish that would be a replacement for Ratchet's wrench. It sounded funny, but when we put it in the game the humor lasted for about three seconds.

We also prototyped enemy layouts and behavior to a much greater extent on this project. The majority of our enemies were well tested and tuned before each level went into production. This process saved us a massive amount of time, since we only built final models and did final coding once we were sure that the enemies would work. Conversely, on the *Spyro* series we were always ripping things out and starting over during production, since we rarely prototyped gameplay. With *Ratchet & Clank*, and for all of our future projects, gameplay prototyping has now become an ongoing process.

Finally, to clearly establish the look of the game, we used our I5 engine to prototype two of the game's planned environments before we had the real Ratchet & Clank technology up and running. It was all smoke and mirrors, but it allowed us to show on-screen what we imagined the final game would look like and put to rest a lot of our own fears about whether or not the game would stand out visually.

## *Tom Clancy's Splinter Cell* What Went Right

**1. A prototype that helped determine resources and scheduling.** Compared to the full production team, our prototype team was tiny. We started with five people, comprising two engineers, one level designer, a 3D artist and myself. We added four more engineers and an animator two months later.

Besides proving to upper management that we are able to solve the technical issues of porting, the prototype we first worked on was supposed to provide a solid base for production. By August 2002, we were quite confident of the outcome: The systems for generating *Splinter Cell's* lighting and shadows on PS2 were integrated into the engine, as were other special effects. What made us even more confident was that we had to create these systems from scratch; it wasn't possible to re-use this code from Xbox version.

We studied how best to allocate hardware resources between the CPU, GPU and memory under the *Unreal* engine. By the time we completed the prototype, our team had a detailed list of how much memory was allocated to each part of the game (map data, character/animation, engine, UI, textures, and so on), and understood how to work with the constraints of the map data to ensure the good frame rate. We tried to be pessimistic while setting those constraints, which turned out to be a good decision -- later on we encountered some bad surprises related to system resources, but because of our earlier estimates, we had built in a sufficient margin of error to deal with them.

The prototype also proved crucial for organizing the production schedule and estimating the necessary project resources. The effort required to port one map was split into several steps, and we used our experience creating the prototype to accurately define the data flow between steps and estimate the overall resource costs.