GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: *MotoGP '06*

By David Jefferies

Introduction

It was in early January 2005 that we received our first Xbox360 development kit from Microsoft and were tasked with moving the MotoGP series onto the next-generation of hardware. Over the previous 5 years we'd developed three versions of the game on Xbox and PC but this was the first time the game was going to receive the radical overhaul needed when jumping a generation.

Our Core Technology Group had been writing our tools in preparation for the next-generation of consoles for up to 2 years previously, but most of the game team were coming from PS2 and Xbox projects with little idea of what to expect.

Next-gen buzzwords were everywhere: normal maps, HDTV and HDR were all new and all being touted as the next big thing.

At that stage in a console's life cycle you have to make a lot of decisions about what's important and what's not – separating the technological wheat from the chaff. There's very little information to go on and because near-launch games have to be developed in such a short space of time, there's little chance of rectifying big mistakes made early on.

What Went Right?

1. Tight Focus

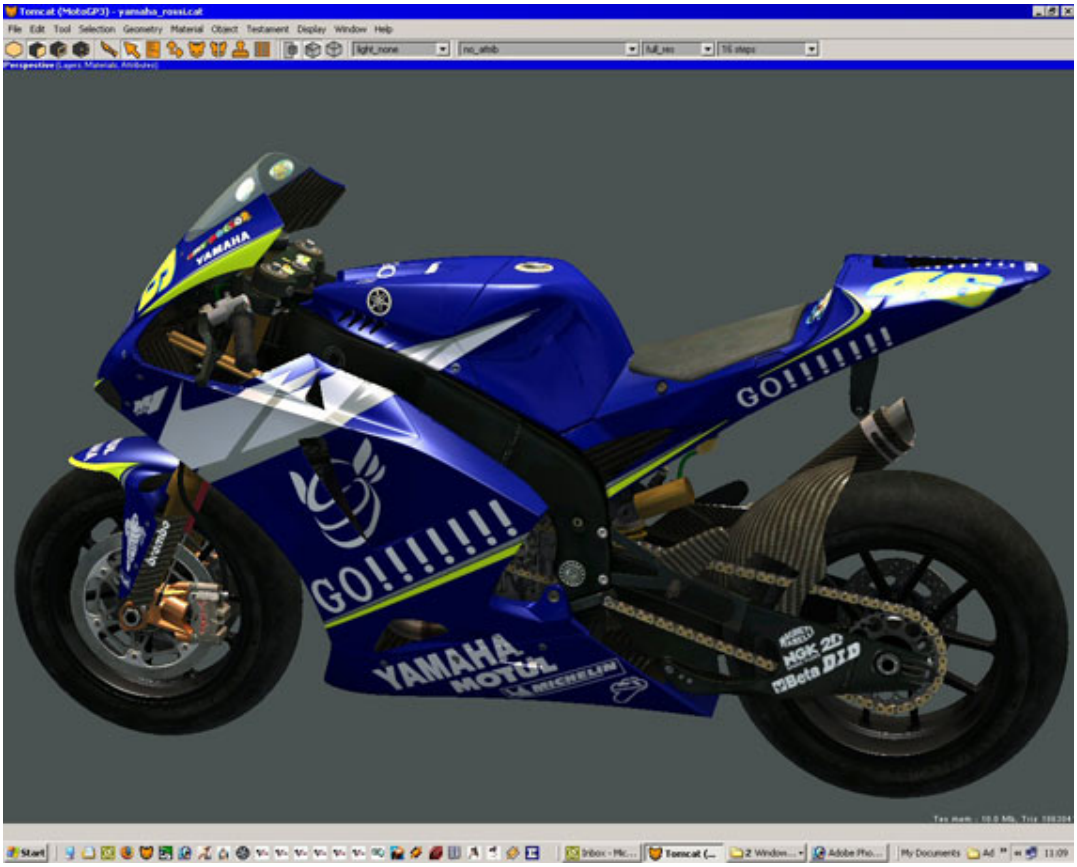
Right from the beginning we knew the focus had to be kept tight. MotoGP had always received praise for its handling, game modes and Live implementation. It was decided, therefore, that these key areas would receive only a light reworking leaving us to concentrate all our resources on a complete overhaul of the graphics systems.

2. Tomcat

Our artists don't model in Max or Maya but in a modelling package that we write for them, called Tomcat. Work had started 2 years earlier on the tool that was based on our SuperTools modelling and texture package 1

The original incentive for writing our modelling tool came because of the lack of curved surface capabilities in the other popular packages. All our tracks and bikes are modelled with curved surfaces and then, at export time, we tessellated the model into any number of polygons. The artists are happy and productive working this way and it has the added benefit that all our resources are totally scaleable in terms of number of polygons. The tool is also much faster than the other packages.

The Tomcat renderer is the renderer that we use in game on MotoGP'06, so the artists have a true what-you-see-is-what-you-get interface.



This was invaluable at the beginning of the project because it allowed the artists to hit the ground running. They didn't have to hang around waiting for us to write the in game renderer – they could get on and model their assets safe in the knowledge that when the time came their assets would work in game. They also knew that we could scale the polygons and texture sizes of their assets according to the power of the target hardware.

They also got busy designing their own shaders. Tomcat has 50 or so small shader fragments that the artists can combine in whatever order they like. This allows them great freedom to experiment with different material types and because the game and tool share renderers, anything they do in Tomcat will look the same in game.

---

3. HDR, Lightscattering, Veggies

When we started the game some of the team were finishing off MotoGP 3 for the Xbox and the rest of us had just come off some PS2 projects. We didn't know much about next-gen, no one did.

We had to make a lot of decisions about what technology would be important. We already had an advanced model renderer that could handle most things thrown at it – normal maps, cubic environment maps, fresnel – we had all those buzzwords covered, we also had some post-processing effects like DOF (Depth Of Field) and bloom, but now we needed to decide on the new technology that was to gel it all together.

We eventually plumped for an HDR (High Dynamic Range) renderer, light scattering and a comprehensive veggie system.

HDR Renderer
The HDR renderer allowed us to encode the intensity of a colour as well as the colour itself. So the sun, instead of just being white RGB(1,1,1) was also encoded at an intensity of 1000 to distinguish it from the white hoardings or white track markings that are much lower intensity. We used this intensity to determine when to bloom the sky or specular reflections.

We can also change the exposure value for the renderer allowing us to over or under expose a scene. This looks great in a tunnel because we'd ramp up the exposure value so as you emerged back into sunlight the scene was brightened and bloomed for a split second, before returning to normal. This simulates the expansion and contraction of the pupil.

---

4. Light Scattering

Light scattering is an algorithmic technique that models atmospheric phenomena. It simulates the way that light from the sun interacts with particles in the atmosphere to recreate the deep red of sunset or the grey of dawn. With the adjusting of a few sliders the artists could recreate the atmosphere from Dubai to Donington. The shader fragment is appended to every shader in the scene giving a very consistent lighting effect.

Veggies
In real life many racetracks around the world are in large fields with lots and lots of grass. In the past the only way we've been able to represent this grass is by plonking a grass photo over lots of flat polygons.

This time around we invested a lot of time in our veggie renderer that could draw up to a million blades of grass giving a really good volumetric effect. It also gives a great sense of speed when you see the grass whooshing by.

The Team

Even with our fairly mature tool-chain generating the assets for the first game of a new generation is always a difficult affair. The programmers are often so busy trying to get the game working that the artists are left with buggy tools, feature requests get forgotten and work has to be re-done because plans have changed.

It's easy at times like this for resentment to build amongst the team.

Thankfully this didn't happen on MotoGP'06. Our artists and designers are extremely experienced and were patient and tolerant when things went wrong.

What Went Wrong?

1. Framerate

All the MotoGP games by Climax have to run at 60fps. It's a completely non-negotiable part of the project. And so when, in January 2005, we sat down and hammered out the feature set for MotoGP'06 with THQ, the requirement of 60fps was writ large.

Getting launch or near launch titles to run at 60fps is always going to be challenging. You start development on hardware (or sometimes an emulator) that bares scant resemblance to the final product – and final hardware usually only shows up very late in the cycle and even then you're lucky if you get more than a handful of kits.

In the beginning we planned fastidiously to hit the magical 60 mark, but at that stage we had little idea of the final hardware so our only option was to make all our assets scaleable. We're lucky that all our tools are built around modelling higher order surfaces and so at a touch of a button we change a bike from 1,000,000 polygons to 1,000 polygons. The same goes for the environment. And our exporters will scale the textures or vegetation to whatever limits we desire. Basically, we thought we had all angles covered.

Now, I'm a console programmer as are most of my colleagues. It's been 10 years since I released a PC game. This lack of PC experience led us to overlook something that would have been obvious to a PC coder. The single biggest performance drain on MotoGP'06 wasn't the number of vertices or textures but the number of draw calls.

In November 2005 our game was running at 12fps, with the render loop taking nearly 2 frames.

That month news came through from Microsoft that the changes to the Xbox360 SDK that would allow us to circumvent these draw commands wouldn't be ready in time for our launch. We were in very serious trouble indeed.

Neil, one of our engineers, set about moving the render loop onto its own processor core. This was a major engineering challenge that took 6 weeks to complete, but even then the renderer was running over a frame.

Another engineer, Matt, eventually solved the problem by writing an offline automatic occlusion system. It would travel round the track and take snapshots of the scene every few yards – and query the GPU to see which draw calls resulted in pixels being written to the screen. At run time it would not execute draw calls that didn't contribute to the scene, and this cut our draw call count in half. The system has to revert back to frustum culling if you stray too far from the track.

---

2. Loading Times

The new generation of consoles have up to 8 times the memory of the previous generation and yet the DVD read speeds have increased by only about 3 times. Even assuming a perfect data read rate it would take about 32 seconds to fill 512Mb of memory. In practise you need to factor in seek times as the game loads different files, and so MotoGP'06 takes about 40 seconds to load a level. And 40 seconds is a long time.

There are many different ways of speeding up load time from having fully stream-able worlds to keeping as much as possible data resident in memory. The old MotoGPs employed none of these techniques. They'd never needed to. They could fill the Xbox1's memory in 12 seconds and better than that they could dump all that data to its internal hard-drive so that next time round it loaded 10x faster.

On the 360 we were aware of our shortcomings but the engineering effort required to rectify them was so huge, and the launch window so close, that they never got addressed.

On our next game this will be a high priority.

3. The Data Build

When we started on MotoGP'06 we wrote the exporters alongside the renderer. As features were added to the game so the exporter was extended to support those features.

Crucially, the exporter never went through a consolidation phase and as is all too often the case it became a second-class citizen to the game. Its code became entangled and its performance was seldom monitored.

Our build machine, which was a fast piece of kit, ran the exporters. By Beta it was building 39 tracks and 40 bikes. This process took 17 hours. After the build machine finished we needed to copy the files elsewhere and delete some unused archives and manually go through the process of creating a DVD ready version and a magazine cover version. The process simply wasn't as automated as it should have been.

As we went through Beta and on towards Submission we were submitting builds more and more regularly eventually reaching 3 times a week. Soon, we knew, we would have to submit builds on a daily basis and with a 17-hour build time we were heading toward meltdown.

It became clear we hadn't fully thought through the consequences of the huge data sizes that were needed on these new consoles. A development build was 10Gb in size and our creaking 10Mbit network simply couldn't handle it.

At this juncture we accepted the help of Shep, Climax's resident build guru. A gigabit network was hastily installed and, in double quick time, he wrote us a distributed build system that built the data on two machines and collated it on a third, file server machine, ready to be pushed out to all the development kits. It also built a DVD ready version and the magazine demo.

It did all this in 9 hours which meant we could kick it off before we left in the evening and everything was ready for us when we arrived in the morning (even in crunch the team was sensible with its working hours and we seldom had staff in the office between 10pm and 7am the next day).

---

4. Pre-Release Development

It's always difficult developing for a console that doesn't yet exist. The development team are relying on hardware or emulators that are constantly changing and don't really represent the final console.

Decisions are made based on best guesses and features you've been promised get delayed and delayed.

It's always the same and an inevitable side effect of developing a game as the manufacturer is developing the console.

However, Microsoft did a great job of supporting us through development. Their tools are great and they were always really helpful whenever it came to problems, feature requests or optimisation help.

We don't use .NET to build our game because we find it unsuitable for large projects and cross-platform development. We use a system

based on Jam, but it's a testament to the quality of Microsoft's compiler technology that, utilising precompiled headers and the incremental linker, we can recompile the whole project in less than 5 minutes. A single file can be recompiled and linked in a handful of seconds. It's difficult to over-emphasise the productivity gains that this gives you.

Last Word

We learnt a lot from making MotoGP and we got some things right and other things wrong, but the acid test is always the reaction from the journalists and public. So we were very pleased when THQ showed the finished game at E3 2006 and it went down really well, getting a clutch of awards including IGN Racing Game of the Show2.

THQ launched the game in early summer and the reception has been brilliant. Soon after, we made our annual pilgrimage to Donington Park to see Rossi and his chums burning round the track. It's great to see it in real life and gives us loads of ideas for how the game could be better, and of course… the artists can never get enough umbrella girl reference!

1. http://www.gamasutra.com/features/20020626/hargreaves_01.htm
2. http://uk.games.ign.com/articles/709/709355p3.html

Return to the full version of this article