GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Westwood Studios' Command and Conquer: Tiberian Sun

By Rade Stojsavljevic

Ever since the release of Westwood's *Dune 2* in 1992, real-time strategy (RTS) games have become the hottest-selling computer games around. Countless RTS games were released soon afterward including *Command & Conquer (C&C), Red Alert, Warcraft II, Age of Empires,* and *Total Annihilation.* These games have propelled the genre to new heights and have drawn an increasing number of fans.

After the success of *C&C* and *Red Alert*, the team at Westwood Studios started work on *Tiberian Sun*, the sequel to *C&C*. To build the game, we assembled a team that consisted of veterans from *C&C* and *Red Alert* along with a couple of new faces, including me. We started with the goal of taking what made *C&C* fun and expanding it even further.

To begin the development process we reviewed what makes a great RTS game and came up with one answer: tactics. Westwood doesn't build games based on a specific technology and we never sell technology over the game play. We have a firm belief that a great strategy game must have interesting, fun, and new tactics that afford players a multitude of unique ways to play a game.

We wanted *Tiberian Sun* to appeal to a broad audience, yet also appeal to core game players and fans of the series. Towards this goal, we continued to apply a "wide and deep" approach to designing the tactics we created. Wide and deep essentially means a nice assortment of diverse yet readily apparent tactics that, under the surface, contain an even greater number of tactics. With this approach, you can provide first-time players with a number of different things to do while letting more experienced players discover new and advanced tactics on their own. These design goals made working on the game more challenging — as if being the biggest project in Westwood Studios' history wasn't enough.

Concept sketch of a GDI Titan.

What Went Right

1. Maintained *C&C* style of game play

One of the most difficult tasks we had to overcome during the development of *Tiberian Sun* was to maintain the feel of the original. When making a sequel, the question that always has to be answered first is, How far do you stray from the original game to make it compelling, yet still familiar? The intent with *Tiberian Sun* was to maintain, as much as possible, the feeling of the original while providing new and interesting tactics for players to master. To aid in this goal, when adding a new feature we asked the questions, "Is this consistent with *Command & Conquer*?" and "How can we make it easier and even more exciting?"

In this area, it really helped to have a development team that worked on the previous games. They were able to draw from previous experiences to create a consistency in the game dynamics. This gave the team a great deal of independence since everybody already had a good idea of how the game was supposed to look, play, and feel.

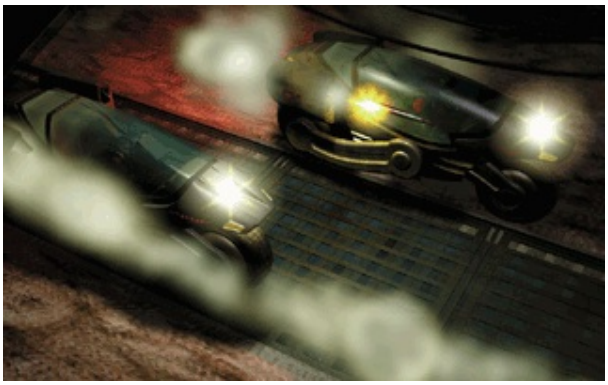
A Nod obelisk of light incinerates its attackers.

The main areas we focused on in order to be consistent with previous games were the user interface and unit behavior. We knew we had to keep the sidebar metaphor for unit construction but we wanted to update it to accommodate new features, such as unit queuing, waypoints, and power/energy control. For unit behavior there was a set of rules that we had to conform to, specifically how a unit deals with player commands so that its internal logic never overrides a player's orders. One of the times we tried to change the rules was when harvester threat-avoidance logic was introduced. I remember hearing lead designer Adam Isgreen screaming at his computer when his harvesters refused to obey his orders to retreat. We decided to scrap that idea shortly afterward.

It was important for the overall visual presentation of the game to bear a resemblance to its predecessors in order to maintain a consistent artistic style. We decided to alter the perspective slightly, rotating the camera to create a three-fourth isometric perspective that afforded a better sense of depth and realism in a 3D perspective. It was at this point that we decided not to use a polygonal engine since it wouldn't be possible for us to keep the system requirements low enough to achieve the mass-market appeal that we wanted. Also, at the time we planned to release *Tiberian Sun*, 3D accelerator cards and systems weren't fast enough for us to maintain the visual detail we wanted for the hundreds of units and structures on-screen at once.

2. Working on a sequel to a successful franchise.

Being the fourth RTS game Westwood has done, there were a lot of lessons learned that the team was able to carry forward into *Tiberian Sun*. First, we had an established and streamlined user interface. This user interface has been a cornerstone of Westwood RTS games since *Dune 2* and we've been gradually improving it ever since. Anyone who has ever played a Westwood RTS is immediately familiar with the controls and can jump right into the action. Additionally, the interface is simple and intuitive enough to let new users become comfortable with it in a short time.



NOD bikes fire at an underground UFO.

Another nice benefit of making a sequel is that we had a set of basic features we knew worked based on previous games. These provided a solid foundation that could be expanded upon and modified as needed. We started with features from the previous games that we knew we wanted and updated them to fit a world that was 30 years in the future. Tanks evolved into two-legged mechanized walkers, soldiers could now use drop pods launched from space, and cloaking technology advanced to yield a stealth generator that hid many units and buildings at once.

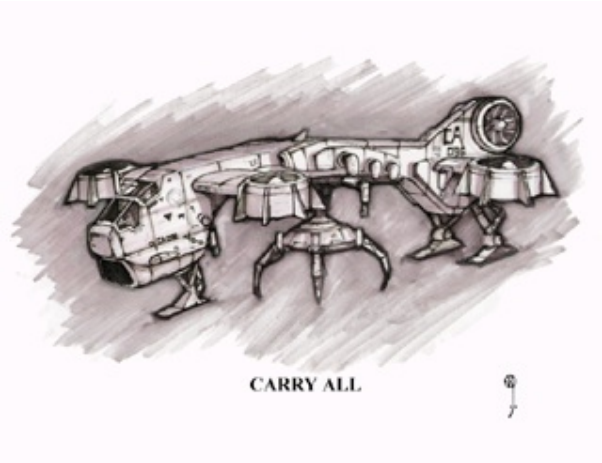
When it came time to create the story, we already had the basic framework in place. There was a very rich and fascinating world to draw upon when creating new characters for this story. The one difficulty encountered was making sure the story could stand up on its own and be accessible to new players without subjecting players familiar with previous games to mind-numbing exposition. To solve this problem, we set the story 30 years after the end of the original, which provided an opportunity to create an outstanding introduction that showed players what had been going on in the world.

3. Team experience and cohesion

The *Tiberian Sun* development team is one of the most experienced and professional teams I've ever had the privilege of working with. For many of the team members, this was the fourth RTS game they had done (the previous being *Dune 2*, *C&C*, and *Red Alert*). This level of

experience was key in allowing the team to conquer all the obstacles thrown in their path. Even though I had worked on half a dozen titles before I started on *Tiberian Sun*, at first it was a little unnerving for me to be working with a team of this caliber.

Several members of the programming team had worked together on previous Westwood RTS products and were accustomed to each other's coding styles. New programmers were quickly assimilated into the team and were able to adapt well. The coding rules and Westwood libraries allowed the programmers to familiarize themselves with each other's work with minimal difficulty.



Concept sketch of a GDI carry-all.

The designers had worked on previous RTS games and were very familiar with the universe before we started the project. This saved several months since no one had to familiarize themselves with anything except the design for *Tiberian Sun*. The tools used were derivatives of the *C&C* and *Red Alert* editors, which also minimized the ramp-up time required before they could produce missions. The designers worked well together and were friends; something that helped a lot when there were differences of opinion. It proved to be very beneficial to know that you could argue your point and not have to worry that the person you were arguing with would hold a grudge afterward.

Without the technical knowledge and creativity of the artists on the project, we would have suffered a great deal of pain when integrating artwork. Like most projects, *Tiberian Sun* had a specific set of technical criteria that had to be satisfied when creating art for the game engine. On this front, we reaped the rewards of having artists who had done it all before. They had worked with our programming team and knew the tools well enough that they were able to head off potential problems before they could get out of control. The cinematic artists had much of the same experience; they didn't have as many technical restrictions as the in-game artists, which allowed them to be able to express unbridled creativity. The cinematic artists didn't have to deal with frame limitations or palettes. Also, compared to previous games, the movie player in *Tiberian Sun* allowed for full-resolution movies (as opposed to previous games where every other line was cut out) using 24-bit color depth and a 15FPS frame rate. I still remember the first time we saw the movie in which the Mammoth Mk. II laid waste to an entire Nod base by itself; it left everyone in the room speechless.

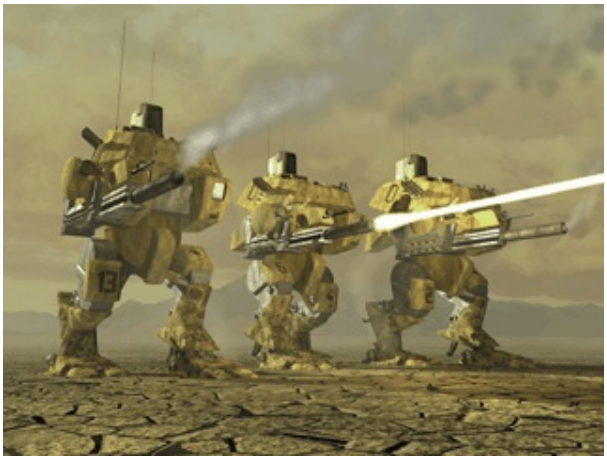
An Orca carry-all transports a hover MLRS.

The final piece was the management team. Under executive producer Brett Sperry's strong leadership, we established systems to deal with routine tasks, facilitated communication between the teams, and were able to avoid a lot of problems early on. Brett has always been very protective of the *C&C* franchise and with *Tiberian Sun*, his clear and consistent vision of where the game should be was absolutely critical to the project.

4. Balancing process

Balance is one of the things that can make or break a RTS game. It's one of the hardest things to do on the design side of the product since you're essentially trying to optimize an equation with dozens of independent variables. If you get it wrong, you'll have a boring game and a horde of disgruntled fans cursing your name forever. When the issue of balancing comes up, you'll often hear about the "rock-paper-scissors" idea, but I like to think of it more in terms of a chess game. You've got a lot of different pieces, each with a unique function and set of strategies that takes a long time to master.

Having made several RTS games before, the team knew how to balance a game. We started with two approaches: one scientific and one artistic.



GDI Titans lay waste to the Nod base.

Using the scientific approach, we started with the relatively simple idea that in a steady state units with an equivalent cost should do equivalent damage to one another. The basic idea is that if I have $1,000 worth of units and you have $1,000 worth of units and they fight, the fight better be really close. From here, we kept adding variables until we had a relatively playable game.

The next step was a lot more artistic and was where experience really paid off, keeping the team from long periods of fumbling around blindly. We played countless games with each of us championing one side vs. the other, carefully noting how effective units and tactics felt against one another. We would get together after each game to compare notes, argue our points, get into fights, and then make one change at a time to the game and try it again until we were all satisfied with the results. The whole process took about three months for *Tiberian Sun*, compared to six months for *C&C* and four months for *Red Alert*. Even after the countless games we played against one another, we still got into shouting matches during close multiplayer games. When this happens, you know you've got a winner on your hands.

5. Mission Design

Mission design is one of the most important elements of RTS games. Based on experience with previous games, Westwood has established a series of processes that are used whenever a mission is created. We've designed these processes to foster creativity, maximize efficiency, and promote communication between the design, programming, art, and management groups. This process has been refined on every project and we've taken it to the next level with the upcoming *Firestorm* add-on.

Drop-pod infantry surveys the battlefield.

The process begins with a mission design proposal submitted to the lead designer and producer. The proposal is a two- to three-page

document that contains summary information about the mission such as name, side, difficulty, map size, mission type, and so on. The mission briefing is included along with a description of what the briefing movie should be and all of the critical information that must be revealed to the player. Mission objectives are listed as they would appear in the game, along with specific information on how to achieve the objectives. Win and lose conditions are created, as well as descriptions of the victory and defeat movies that play at the end of a mission. The last things included are all of the new voice and text messages used in the mission.

Once this proposal has been approved, the map for the mission is sketched out on paper. We've found that this process can save a great deal of time since it eliminates distractions and allows the designers to get an overall view of the map quickly. When the designers finish sketching their mission, they proceed to the editor and begin to create the basic battlefield. Terrain is laid down first, followed by buildings, roads, trees, and pavement.

A hover MLRS fires a volley.

The final step to complete a mission is to take a map and add scripting, which takes approximately two-thirds of the time to create a mission. One of the great things about *Tiberian Sun* is that the editor is tied directly into the game, which allowed designers to switch rapidly between the editor and the game. This feature also proved to be a liability, however, because if a bug appeared that prevented the game from running, we couldn't run the editor, either.

Tiberian Sun features a good blend of production (such as building bases) and non-production missions that keep the pace of the game interesting and challenging. We tried not to do the same mission twice and added variety by combining mission types into non-production/production missions that switch from one to the other when players reach specific objectives. Branching missions were added to give players the option of completing sub-missions before they tackled the main objective. By playing sub-missions first, the player makes the final objective easier and it gave the designers added granularity when creating the difficulty levels for the game.

What Went Wrong

1. Unrealistic expectations

The degree of hype and expectations that *Tiberian Sun* had to fulfill was staggering. We had a team of experienced developers who wanted to beat their own expectations while simultaneously building a game that would be everything the fans of the series expected and more. This was not a realistic goal since it's just not possible to make something that will meet everyone's expectations.

One of the things that we did not do was explore all of the new features to their logical conclusions. This would have allowed us to do a lot more with a smaller feature set and provide an even better game. A perfect example of a feature that was begging to be used more is the dynamic-battlefield concept. The basic idea behind the dynamic-battlefield concept is that players' actions alter the battlefield. For example, a player could set fire to trees to burn a path into an enemy's base. We wound up cutting this particular feature because it caused path-finding problems. Also, battles with heavy weapons would cause cratering of terrain which hindered unit movement.

A disc thrower waits for reinforcements.

We could have used it to create more new strategies for players, and since it was one of the more expensive features in the game, we could have squeezed a lot more use out of it.

Trying to fill the shoes left behind by *Red Alert* proved to be daunting. If you had asked a dozen people what they expected out of *Tiberian Sun* before it was released, you would have heard a dozen different answers. We devoted a lot of effort to add as many features into the game as possible instead of just trying to make the best game we could. Getting into a feature war is one of the worst things that can occur during development because it siphons effort away from adding the "fun" to the game.

2. Feature creep

Tiberian Sun started strong and we developed a robust and large feature set we intended to fulfill. The project started smoothly, but as we progressed, the temptation to add new features not included in the design document grew. These features arose out of shortfalls in the original design, omissions from the original design, and input from fans.

Everybody stresses the importance of working off of a design document and not deviating from it. Unfortunately, this just isn't realistic since every product evolves during the course of development and sometimes the original design proves to be lacking. A team has to be able to incorporate new ideas during development if the final project is to be better. However, the flip side of this idea is that the team must be able to cut features diplomatically when it is in the best interest of the project.

A Wolverine on the firing range.

Tiberian Sun's development had many challenging moments when features had to be cut for one reason or another. A perfect example of this was the ability to order a limited number of units through a drop-ship loading screen before a mission. This sounded like a great idea on paper and we had already coded it and incorporated it into the game. It wasn't until we actually played with it that we realized it just didn't fit and had to be removed.

Looking back at the project, I think we could have been more aggressive in cutting or changing certain features to make sure their returns were really worth the development investment. I'm a firm believer in the idea that less is more and that fewer but more fully developed features are the way to go. If a feature isn't amazing, you should cut it or make damn sure it becomes amazing before you ship the product.

3. Post-production complications, compositing woes

Tiberian Sun features the most complex and highest-quality cinematic sequences Westwood has ever done. These movies help drive the story elements forward. However, these movies came at a very high price.

Westwood has a soundstage with a bluescreen and in-house post-production capability that allowed us to handle the entire production ourselves. We've done several different projects with video, including *Red Alert*, *Dune 2000,* and *Red Alert Retaliation* for the Playstation. Based on these past experiences, it was decided that we would push the limits of what we could do in *Tiberian Sun*.

Chandra, McNeil, and Brink pose on the Kodiak Bridge.

We started by fully storyboarding every scene in the script. From the storyboards, we built concept sketches of the major sets to be constructed (practical as well as computer-generated) and proceeded to build the sets. Before the shoot there was a three-month lead time for our team of six 3D artists to build the sets. We wanted to have the sets 100 percent complete so we would have camera and lighting information to match up with the live actors.

For various reasons, the pre-production for *Tiberian Sun* was much shorter than it should have been. If you've ever worked in film or television production, you've probably heard the phrase "we'll fix it in post." Believe me when I say there's a reason why this little phrase can spook even the most veteran members of any production crew. Anything you have to fix after the fact winds up being ten times as difficult and ten times more expensive than planning for it in the first place.

Everybody on the team knew this and we tried as hard as we could to work out all the details before we started the shoot. The problem was we didn't have enough time and couldn't change the date of the shoot because we wouldn't have been able to get our two main actors, James Earl Jones and Michael Biehn. Going into the shoot, we had a pretty good idea of how we were going to work out all of the technical details such as camera tracking on a bluescreen, matching lighting to computer graphics, compositing, and so on. However, we ran into difficulties because we didn't allow enough time for the more complex shots and were forced to edit on the fly during the shoot.

Umagon prepares for a take.

An unforeseen problem during the post-production was that we dramatically underestimated the storage and network requirements of working with 60 minutes of digitized video. Westwood has a very robust and fast network with a large amount of storage space, but it was never designed to meet the needs of video post-operation. An amazing effort by the MIS staff and a couple of called-in favors got us enough storage space on the network to keep going.

From the start, the team struggled to get video from digital beta to the SGI- and PC-based compositing systems. Footage was digitized on an Avid system and copied to file servers for distribution to the PCs. The SGIs grabbed the footage directly from tape using built-in digitizing hardware. From the compositing stations, various shots were completed and transferred back to a file server to be compressed and put in the game. This, along with the fact that many individual scenes were worked on by several artists, multiplied the storage requirements several times over. In the end, the video assets were spread across four separate file servers and took up well over 500GB of space.

Not only was space a problem, but moving hundreds of megabytes of files a day from machine to machine became a bottleneck. A few minutes here and there to transfer files doesn't sound like much until you add it all up. If we had it to do over again, we could have alleviated the problem by building a very specialized (and expensive) network, by getting hardware that allowed artists to digitize their footage directly from tape, or by reducing the scope of the project and sidestepping the problem entirely.

4. Locked documents too early

One of the side effects of schedule slippage was that we locked our documents too early in order to achieve the localization plan. We knew this was going to wind up causing us significant pain, but at the time there was nothing we could do to avoid it. The result turned out well, but a lot of time and effort was spent to make everything work together.

GDI forces destroy a vital Nod caravan.

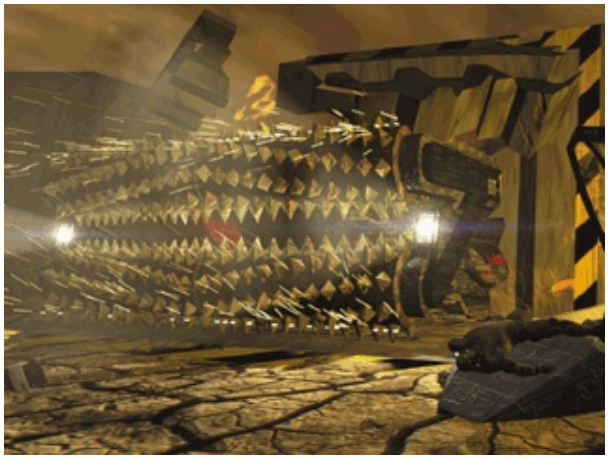
At the point when we locked the audio script, mission design and balancing were not complete. As we played through the missions, we realized that certain objectives were not clear and needed to be explained further. The previous method for doing this was to have the in-game AI persona (Eva or Cabal) relay the information to the player through voice cues. This was not an option for *Tiberian Sun*, however, since we made the switch to professional voice talent for Eva and Cabal. Costs and scheduling didn't allow us to do as many pickup recording sessions as we wanted. Also, the locked audio scripts were already localized and recorded, which made recording additional lines out of the question.

The only option available was to redesign the missions or add text to the missions to make the objectives clearer. Redesigning the missions would have added at least a month to the already late schedule, so we quickly ruled that option out. We wound up going with text that popped up in the missions, although the original design called for all text in the game to be accompanied by a voice-over.

5. Scheduling problems

As with most projects in development today, *Tiberian Sun* suffered from scheduling problems; ours resulted in a nine-month delay. There wasn't a single reason that caused the product to be delayed, but rather a series of seemingly minor contributing factors.

Brett Sperry has a rule of thumb that we often refer to when scheduling projects. When you add one fundamental new technology to a project, it can cause slippage up to 90 days. When you add two fundamental new technologies it can add a year to the anticipated release date. When you add three or more new technologies it becomes impossible to predict the release date of the project accurately.



Devil's Tongue Flame Tank crashes a gate.

Tiberian Sun features three new systems that resulted in an unpredictable schedule. First, we switched our core graphics engine to an isometric perspective in order to enhance the game's 3D look. This resulted in a cascade effect of broken systems that weren't anticipated. Bridges that could be destroyed and rebuilt, for example, wound up taking over ten times as long to program as we originally estimated. Adding bridges complicated systems such as path-finding, Z-buffering, rendering, unit behavior, and AI.

Scripting was another area in which we added a slew of new functionality. We added an increasing number of triggers to the game to allow the designers flexibility in creating the missions. Each new trigger added was more specific than the last and was used for increasingly rarer conditions. Since triggers could be used in combination, we ended up with an overwhelmingly large number of events that needed to be debugged. We would often fix one trigger to work in a specific situation and inadvertently break the same trigger in a different situation.

AI and unit behavior was the third main area that used new technology. We set out to create a challenging and fun AI that could react to the player's actions and change tactics to compensate. We should have focused on fewer areas of the AI instead of trying to redesign the whole package from the ground up.

Overall Tips

With *Tiberian Sun*, we built the game we originally set out to build over three years ago. Almost all of the new engine features we designed were implemented in the final product, and many more were added along the way. We built a game that is as easy to play as its predecessor while offering up lots of new units featuring interesting tactics. All of this was done while keeping the system requirements low enough to run on most systems: a 166MHz Pentium with 32MB RAM and a 2MB video card.

We learned, or relearned actually, a few more things about making RTS games that weren't listed above. They are:

- If the game has Internet or multiplayer capability, build this functionality as soon as possible since it will let you get into the game and balance it early.
- Don't shield yourself from reality. If your game supports Internet play as well as LAN play, don't play only LAN games and assume that Internet performance is acceptable.
- Keep the story tightly focused on players' actions and don't treat the story as a separate entity. Remember that the player is always the main character.
- Wherever possible, try not to mix disparate technologies (3D visual systems with 2D, for example) that have inherent problems working together. Instead, go back and modify the design

Concept sketch of a GDI Orca bomber.

In the sense that *Tiberian Sun* was a game with lots of expectations for a sequel, it was a lot like *Star Wars: The Phantom Menace*. No matter how the final product turned out, there would be people that complained that it was too much like the original and others who thought it wasn't enough like the original. As a company, we set out to deliver what we intended — a fun new RTS game that offers players a slew of new tactics.

Orca fighters escort a transport.

After three years of working on *Tiberian Sun*, it was a great feeling to finally finish the game and see it on the shelves. No matter how many products you ship, that feeling never goes away. *Tiberian Sun* broke Electronic Arts' sales record for the fastest-selling computer game in the 17-year history of the company with more than 1.5 million units sold so far. But best of all, the team is proud of the product they created and can't wait to get started on the next one.

Tiberian Sun

Westwood Studios
Las Vegas, Nev.
(702) 228-4040

http://www.westwood.com

Release date: September 1999

Intended platform: Windows 95/98/NT 4.0

Project length: 36 months

Team size: 25 full-time, 15 part-time developers

Critical development hardware: Pentium Pro and Pentium II machines, 200 to 450MHz dual-processor with 128 to 256MB RAM, Creative Labs sound cards, Windows 95/98/NT, SGI 02 workstations, BlueICE accelerators

Critical development software: Microsoft Visual C++, Lightwave, 3D Studio Max, Discreet Flint, Adobe Photoshop, Adobe After Effects, Adobe Illustrator, Avid Media Composer, Filemaker Pro, Deluxe Paint

Rade Stojsavljevic was the producer for *Tiberian Sun* the expansion pack *Firestorm*. Before coming to Westwood, he worked on military simulations and adventure games at various small development houses. When he's not out getting doughnuts to bribe the team with, you can usually reach him at rade@westwood.com.

Return to the full version of this article