Postmortem: Poptop Software's *Tropico*

By Brent Smith

In the spring of 1999, Poptop had just wrapped up development on the successful *Railroad Tycoon II (RT2)* and its expansion, *The Second Century*. At the time, Poptop was staffed by the overwhelming count of four artists and two programmers. Being that small, we had had no time to think about anything other than the current project, and suddenly we found ourselves sitting around a table, eyes still slightly glazed from the inevitable project-end rush we had just gone through, looking for new ideas.

These were uncharted waters for Poptop. *RT2* had been based very closely on Sid Meier's classic *Railroad Tycoon*. The original *Railroad Tycoon* had been an inspiration, a design manual, a blueprint for making a good game, and a launching point for new ideas. The upside was that a good part of the design work had been done for us. The downside, as we were to find out on our next project, was that it left us a bit naïve about the effort it would take to create a new game from an original idea.

As we sat around our company card table, brainstorming ideas, one idea quickly jumped to the forefront. The idea of taking a building game and putting a political game on top of it had captured everyone's imagination. With our creative energies renewed by a fresh idea and the thrill of starting a new project in our hearts, we rushed off to create *Tropico* — each of us in our own way.

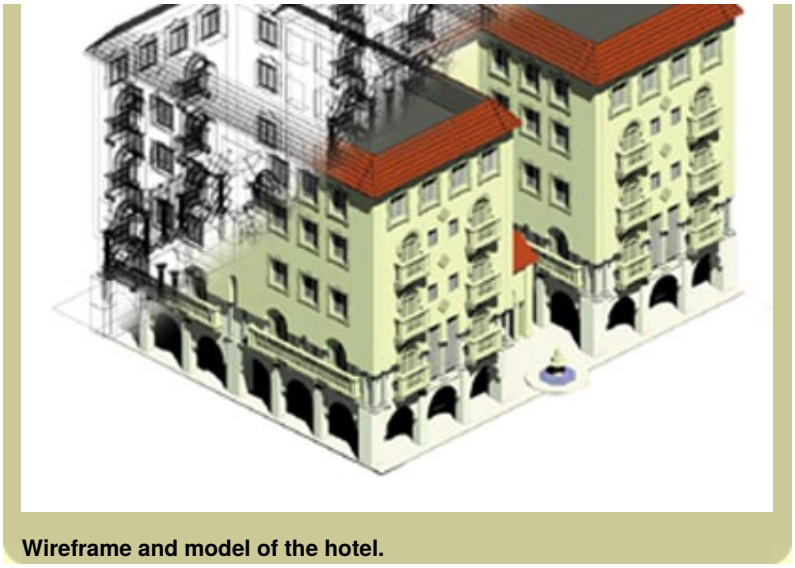
Actually, it wasn't that bad. We did discuss major elements of the game. We knew that it would have buildings and people that the player would not control directly. We knew it would use the *RT2* engine but would be more ambitious than *RT2* had been. As we rushed off to begin development, that was about all we knew — and, as we were to discover later, each person on the team didn't even share the same vision about the things we thought we did know.

During the project, Poptop grew to the bloated size of 10 employees — seven artists and three programmers. It is a testament to the talent and hard work of this team that we ended up with a strong, fun product in spite of the pitfalls that we encountered along the way.

What Went Right

1. Created "deep" characters. It was obvious from the beginning that the most important aspect of*Tropico* would be the people. If we intended to have a game in which the player didn't have direct control of the units on the map, then we had better make sure that the people acted in a reasonable and somewhat predictable fashion.

This was no trivial task. Each unit on the map (in the later stages of the game there can easily be more than 500 units) has more than 70 characteristics, which determine its actions and reactions to the player. This includes items as simple as name, age, and what part of the island the unit was "born" on, to things as complex as the unit's satisfaction with various aspects of the environment (religion, national pride, pay compared to others in the same situation, and so on). Additionally, as units live their sim lives — they are born, prance about as children, enter the workforce at a certain age, and eventually retire and die — we keep track of their families. Each unit potentially has a mother, father, spouse, and multiple children. Units also know about their grandparents, cousins, aunts, and uncles. What this means to the player is that the repercussions for treating one unit badly filter through their family tree much like you would expect in real life. Send Juan Pablo Ramirez to jail, and not only are his parents, wife, and children upset, so are his five siblings, their 18 children, and so on down the line. This added a lot to the political atmosphere of *Tropico*.



Wireframe and model of the hotel.

Our second goal with this system was hiding its complexity from the player. Part of the fun of *Tropico* is trying to see and understand what response your actions will evoke from the population of your island.

Although we kept detailed information about each unit in the game, there was a balancing act in taking advantage of it without flooding the player with information. I think we succeeded here. Our interface provides the player with in-depth information about the people in the game — making them come alive — without overwhelming the player with having to know trivial details about them.

Unit development was not easy, but because we identified this as a critical area up front and spent a lot of time and manpower addressing it, it turned into one of the strengths of *Tropico*.

2. Small, streamlined, and talented team. Being as small as we are certainly has a downside, the most serious being that we are limited in the things we can do in a given amount of time by sheer lack of manpower. However, the advantages to a team this small, at least in terms of developing a project such as *Tropico*, outweighed the disadvantages.
Everybody at Poptop knows everyone else. Not just knows them, but knows what they are working on, what their strengths and weaknesses are, the name of their significant other, and so on. There's no hiding here. If you screw up, people know it was you. If you do something brilliant, everyone knows that too.

This kind of intimacy makes us very streamlined. Everybody knows where he stands and what his job is. There is no middleman; if you need to talk to someone, you talk directly to him. There is no distraction of having team members pulled off to work on a different, behind-schedule project or promotional material. We did one thing, *Tropico*, and that's all we did.
Of course, this kind of team only works if every member is talented, and that is, without a doubt, the case at Poptop. I see it every day, and I think the players of *RT2* and now *Tropico* have seen the result as well.

3. Homegrown tools. Upon the completion of *RT2*, we had accumulated a nice suite of tools for preprocessing and manipulating art assets and interface elements. With *Tropico*, we continued this work, both improving the existing tools and developing new ones.
Our most-used tool was a program written to preprocess art assets into our custom format. It also had the ability to clip and scale the art, and also reduce animations to a keyframe and delta information for efficient storage. When *Tropico* began, we further modified this tool to allow it do work with 24- and 32-bit TIFF files. A palette reducer/optimizer was added to create 8-bit palettes from one or more higher color-depth images. We also included the ability to add parameters to the instruction set that the program used to process the files, allowing such things as tinting (which allowed us to construct placeholder art rapidly by simply taking an existing image and tinting it to another color), and lightening or darkening of the image. Finally, we added support that allowed us to read image-depth information stored in RLA files and store it with the image. This information could then be used to tell us the Z-order information of the various parts of the image, which allowed us, for example, to handle units walking behind parts of a building while walking in front of others.

Another tool that we inherited from *RT2* and improved upon during the development of *Tropico* allowed us to create, size, and position interface elements outside of the code. Using a simple scripting language, interfaces could be built and then compiled by this tool into a format that could be used by the *Tropico* code.

The dictator's mansion.

A new type of tool that we developed and began using with *Tropico*, and which turned out to be a real time-saver, was used for game data manipulation. Using MFC (which I'd never recommend for any software intended for release, but which is tremendous for quick development of tools such as these), we quickly built a very robust unit editor and building editor. These editors allowed us to manage all the data associated with a particular type of building or unit outside of the program. This capability was invaluable for balancing and tweaking the data, as it allowed us to change information in a relatively safe way while the game was running and see its effects immediately upon the game world.

4. Fun topic. Without a doubt, one of the key factors in the success of*Tropico* was the topic. During our brainstorming sessions, a number of ideas were thrown on the table, but the idea that became *Tropico* was the one that had everybody excited. While a lot of the elements of *Tropico* can be found in other games, the mixture of those elements and the setting itself had everyone eager to see what we could make. This enthusiasm translated outside of the company, too. Nearly everyone to whom we showed the game voiced their enthusiasm about the freshness of the idea. Something about the idea of ruling an island full of sun-drenched beaches and tropical beauties strikes a chord in most people's hearts.
One of the most promising indicators late in the project that we had something special on our hands was that we were still eager to play the game, and were still throwing out new ideas, even after spending two long years developing it.

5. Localization. Having been involved in the translation of*RT2* into a variety of different languages, including double-byte handling for Asian languages, we knew up front that this was something that we needed to be concerned with in Tropico.

Fortunately, a lot of the groundwork had been established during *RT2*'s development. The code contained home-grown string manipulation functions, which allowed us to maintain tight control of many of the issues associated with localization. We also had a tool that allowed us to pull strings out of the code base for insertion into a string table near the end of the project. The tool was very useful in that it not only recognized strings within the code, but it was smart enough to disregard strings within comments and strings on lines which we tagged with a special comment telling the tool that it was O.K. for this string to remain in the code (format strings, for example).

This meant that for much of the project we didn't have to concern ourselves with trying to keep a string table up to date, but when the time came, we were able to do it quickly and efficiently. Overall, localization was a breeze. Having seen what a nightmare this step can be on other projects, we were dreading the work we thought we would have to do in this area, but the final tally was less than a man-week of work spent getting the game ready for foreign language translation.

What Went Wrong

1. Lack of up-front design work. Coming off the success of*Railroad Tycoon II*, we were excited to get the chance to work on something new and unique. A few company brainstorming sessions and a game or two of Junta, and we knew that we wanted to do a tongue-in-cheek political game. We couldn't jump into the project fast enough. Ideas were flying hot and heavy. Everyone was excited.

Unfortunately, we failed to realize at the time that everybody was carrying a slightly different picture in his head of what the final game would be. Some were envisioning the stab-your-neighbor, laugh-a-second antics of the Junta board game. Others were seeing the close-up, detailed view of people in action that *Rollercoaster Tycoon* had done so successfully. Most of us were somewhere in between.

Having come from *RT2*, we really found ourselves unprepared for this problem. With *RT2*, Poptop had the original as a blueprint and design document. Design was only an issue so far as how the original game could be improved upon and how current technology could be utilized to improve the game. The game could practically write itself, with Poptop's main concern being to re-create the magic of the original game while adding a few minor twists of our own.

Now we found ourselves with a blank slate, an original idea where every gameplay detail had to be created from scratch. Unfortunately, we approached this in much the same way as we had approached *RT2*. Rather than settling on a unified design, or even trying to create one, each of us ran back to his workstation and began to create what we thought the game would be.

It quickly became apparent that we were not all moving in the same direction. People had very different views of where the game should go. Decisions had to be made on the fly. Some people's visions were cut out entirely, while others had theirs altered to the point that it became something entirely different.

Tourists relax and sunbathe on the beach.

This process was a very difficult growing pain for Poptop. People's feelings were hurt when they realized the idea that they were so excited about was not the idea that we were creating. The game lost "buy-in" from people in the company as it became something that they were less interested in, or someone else's idea that they didn't really understand.
During this time we struggled onward, trying to create this game that wasn't really what anyone had originally intended. Amazingly, we stubbornly refused to stop and consolidate our ideas in meetings or on paper so that the team could be unified in the idea and to rekindle the original excitement for the game. Eventually, working on *Tropico* stopped being a passion and became just a job for many on the team, leading to low morale and loss of productivity.
Hopefully, we've learned from this mistake, and on our next idea (original or not) we will figure out what it is we are creating before we start to create it and try to keep everyone excited about the direction in which we are moving.

2. Modifying the existing code base. One of the givens, decided before any work was even begun on *Tropico*, was that we would use the 3D engine from *RT2*. This would allow us to have working maps with many of the features we would need in *Tropico* almost immediately, thereby giving us a huge jump start on development.

The idea was a great one and paid huge dividends in getting us working almost immediately on the game itself rather than the engine. Unfortunately, in conjunction with this decision, we started from the existing *RT2* code base — not only the engine, but all the game code from *RT2* as well. We were trying to "morph" *RT2* into *Tropico*, which led to all kinds of problems and slow-downs.

First of all, a single programmer had developed almost the entire *RT2* code base. With *Tropico*, the staff of Poptop had nearly doubled in size. Each new programmer immediately faced the daunting task of getting a grasp of the *RT2* code before he could even begin to make the modifications necessary to create *Tropico*.

Second, there were huge chunks of the code base that became dead once *Tropico* was started, such as the multiplayer code. (We had decided pretty early in *Tropico*'s development to scrap multiplayer and concentrate on the single-player experience.) Of course, multiplayer code was integrated very tightly into many areas of the *RT2* code, and at first we tried to work around it. Eventually we tore it out, once we became frustrated trying to determine which areas of the code were dead and which were important.

Finally, the process of working off the original *RT2* code base was inherently dangerous at best. We automatically inherited any bug that had managed to survive in *RT2* and created quite a few more just going through the process of weeding out what code was unnecessary for *Tropico*

We would have been much better off starting with a clean slate and then pulling over those sections of the *RT2* code base that we could use. The *RT2* code was not cleanly delineated between engine code and game code. However, the process of untangling the usable engine code and importing it into a clean code base would have been inherently more bug-free and more comprehensible to those unfamiliar with the *RT2* code, and would have saved us time in the long run.

3. Fun factor versus gee-whiz factor. Because we started with an existing engine, one of the errors that we made during development was to see how far we could push the envelope with the engine, working on "gee whiz" enhancements that would improve the look and the technology of the game instead of features that would enhance gameplay.

The biggest example of this was what we dubbed "Zoom 0." As the graphics in *Tropico* were much more detailed than*RT2's*, we looked for ways to show off these gorgeous images in the game. Allowing the engine to zoom in one level closer than it had previously been able to (Zoom 1) was one of the ways that we did this. In *Tropico*, players can zoom in very close and get very detailed views of the people and the buildings.

Unfortunately, Zoom 0 is not very useful for gameplay, as it is almost impossible to see enough of the map at that zoom level to get a feeling for how you should play. The majority of players tend to stay zoomed out about two levels, occasionally zooming in or out one level as the

situation warrants.

O.K., so we added a feature that allowed us to show off the graphics even if it didn't help gameplay. What's the big deal? The deal is that we pre-scaled all of the images for the various zooms beforehand and stored them in the data file, so these high-resolution close-up graphics ate up as much space as all the other zoom levels' graphics combined. We spent a full 50 percent of our graphics budget on this one feature. As we got deep into the project, it became apparent that memory and CD file space budgets were going to be tight, but we had invested too much into this feature to be comfortable with cutting it. Ultimately, we had to cut other features to create space, features which would have improved the game. Rotatable buildings, more unit animations, and repeating animations on the buildings (such as blinking lights and moving machinery) all had to be cut to make room.

Looking back, it is apparent that tossing out Zoom 0 and putting in more gameplay-friendly features would have been a big net improvement to the overall game.

4. Waited too long for scenarios. After the scenario-intense *RT2* and its add-ons, we were more than happy to try creating a game in which the strengths lay in randomly generated maps and sandbox-style open-ended play. From day one we worked on *Tropico* with this goal in mind. A lot of effort went into creating a map generator that would create pleasing, logical, and most importantly playable maps. We modeled rainfall on what we knew and could find out about meteorology. We researched vegetation, not only to find flora indigenous to a Caribbean setting but also to find out under what conditions a particular plant would thrive and how to represent that in the game. A significant amount of time went into creating realistic mountains and series, even ranges, of mountains. We even went so far as to model the terrain under the water, so that shallow beach areas and deeper waters would be accurately created.

As we neared the end of the project, we had no doubt that our map generator could create some fabulous-looking maps, and, given the number of factors we allowed the player to tweak during map generation, an endless supply of playable maps. However, the game was missing clearly defined goals. Using the map generator, there was no way to create a map that had a specific situation to solve, and only a very limited way to create maps with unique obstacles to overcome. In other words, the game and the maps were great for an open-ended, sandbox style of game, but were lacking in goal-oriented, problem-solving gameplay. While the sandbox mode allows players to create a wide range of different maps, much more depth and many hours of play could have been added to the game if we had included a rich set of scenarios and the tools for the players to create even more.

As we realized this late in the project, we scrambled to create scenarios to include. Unfortunately, our tools in this area were underdeveloped, and time had to be squeezed out of people's schedules even to produce what we did. The result was that *Tropico* included a very limited number of scenarios (eight with the game and two others included in some promotional CDs) that weren't nearly as involved as they could have been, and certainly weren't up to the standard that we had created in *RT2*.

Another by-product of this oversight was that we never spent time polishing the map editor tools that we used in development. The original game design was oriented toward random-map play, so we never saw the need for a more sophisticated editor until late in the project. These tools ended up being disabled in the release version, disappointing many fans who were hoping to create their own maps. Fortunately for our fans, a map editor should be available in an upcoming patch.

An in-game overview of the island.

5. Lack of unified artistic vision. As I mentioned, one of the effects of essentially bypassing the design phase of this project was that there was a lack of consistent vision among team members. One of the places that this became most apparent was in the game art.
Almost all of the artists at one time or another during the project worked on creating buildings for the game. They were given only a vague notion of what type of building they were to create — a paper sketch or a very crude 3D mockup — and left on their own to move forward from there. Halfway through the project, the problems with this became apparent. Scale varied wildly from artist to artist, as did level of detail.

The same problems were occurring with the character animations, as the two artists working on those had taken different approaches. One artist was striving toward very lifelike figures with complex animations, while the other created more cartoonish parodies of *Tropico's* inhabitants with more outlandish but less complex animations. Both artists had a clear idea of what they were trying to do, and both accomplished their respective goals brilliantly, but the difference in their approaches can still be seen in the release version of the game if you compare, for example, the banker to the female luxury tourist.

At this point we appointed a person to take on the role of art producer. His job was to try to coordinate the artists' efforts and make sure they shared more or less the same vision. But the damage had been done. Team members had to spend valuable time sifting through and reworking art. Frustrations mounted as artists who previously had been able to work toward their personal vision now found themselves having to compromise to a shared vision of the whole team.

This problem could have been significantly reduced with more up-front planning and more ongoing feedback to the artists as they completed each task. We definitely learned from this process and will improve upon it in our next game.

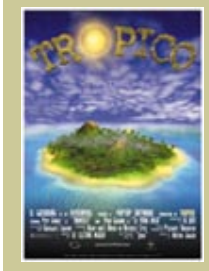
In Hindsight

It's pretty much the experience with any game project, whether the developers will admit to it or not, that you look back and see mistakes that you made and bemoan the ways that the game could have been better if only those mistakes had been avoided. *Tropico* is certainly no different in that regard.

Every member of the *Tropico* team felt the sting of loss at some point or another when a feature that they were particularly fond of was cut. Each of us can look back and think of a hundred ways that we could improve *Tropico*. That, in itself, is a good sign. At the end of two years of development, we still cared about the game and wanted to make it better. Everyone was happy with what we had created, but no one was satisfied with the details. Art is never done.

We learned a lot individually and as a team about how to approach a project and how to manage it once it is underway. This was our first attempt at a completely original idea, and although we encountered a lot of pitfalls along the way and stumbled more than a few times, I think the end result is pretty amazing — something that we are proud of and that the game's fans will enjoy. Considering that *Tropico* was done with a team of only 10 people — tiny by today's standards — the game's success is a testament to the Poptop team's talent, creativity, and hard work.

Game Data



Poptop Software's *Tropico*

Game Data Publisher: Gathering of Developers

Number of Full-Time Developers: 10 (7 artists, 3 programmers)

Number of Contractors: 1 musician

Estimated Budget: $1.5 million

Length of Development: 2 years

Release Date: April 2001

Platforms: Windows 95/98/ME/2000/NT 4, Macintosh

Development Hardware (Average): 550MHz Pentium IIIs with 512MB RAM, 40GB hard drives, and a variety of 3D cards running Windows ME or 2000 (programmers) or Windows NT (artists)

Development Software: Visual C++ 6.0, Visual SourceSafe 6.0, 3DS Max 3.1, Character Studio 2.2, Photoshop 5.5, Tree Factory plug-in for 3DS Max

Notable Technologies: Bink Video, Miles Sound System

Project Size: Approx. 150,000 lines of code (plus 20,000 for tools)