GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Freeverse's *Top Gun* For iPhone

By Justin Ficarrotta

[Freeverse designer and programmer Justin Ficarrotta recounts what went right and what went wrong with the development of the iPhone game Top Gun -- particularly focusing on how fans should always be in mind when working on a licensed game.]

When we, at Freeverse, got the opportunity to create *Top Gun* and *Days of Thunder* for Paramount Digital Entertainment, we knew we had the right franchises to create something really fun and awesome. *Top Gun* has become especially popular, so this is a closer look into how the game came to be.

Background Information

My name is Justin Ficarrotta (or just "Fic"), game designer/programmer at Freeverse, Inc. in Brooklyn. My background is in action- and style-heavy arcade games, such as the Freeverse-published *Kill Monty* or my indie efforts *Kill Dr. Coté* (from which *Kill Monty* was born) and *Laserface Jones vs. Doomsday Odious*, which are both award-winning entries in the annual uDevGames contest.

Freeverse started off as a successful Mac game/software company in the early '90s and has since branched out into Xbox Live Arcade (you can read our *Marathon 2: Durandal* postmortem) and, more recently, the iPhone.

Our games *Moto Chaser* and *Flick Fishing* both made Apple's "Top 10 Paid Apps of All-Time" list (at #6 and #8, respectively) and our library of iPhone games has just broken 20 with the release of *Skee-Ball*.

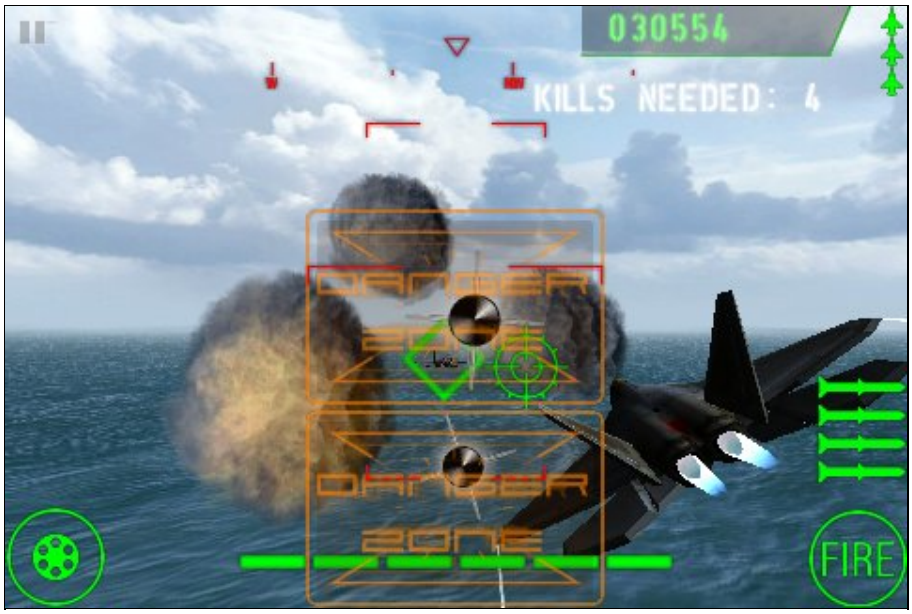
But enough with the self-promotion. Let's talk *Top Gun*.

About *Top Gun*

Top Gun is an iPhone game based on the classic 80's movie starring Iceman and Maverick. Taking place years after the movie, the story revolves around a new set of recruits under the tutelage of Maverick and Iceman, who are now instructors at the Top Gun academy.

The gameplay is very similar to games such as *After Burner* or *Space Harrier*: a third person chase camera behind the player's jet, which flies on rails through a 3D world and can steer freely within a set distance from the level path. Steering also moves a crosshair, which will lock on to enemy jets as it is scrubbed over them, and allow missiles to be fired. The player also has a vulcan gun, which does instant damage to anything under the crosshair, but it must be held over the jet as you fire.

Threats in the game came in the form of "Danger Zones" that would appear on screen, giving you about a second's worth of warning before impact occurred in that area. Enemy missiles would create a Danger Zone, as well as obstructing scenery and ground-based anti-air fire.



What Went Right

1. Getting the rights to Danger Zone

No, really. I'm not kidding.

Paramount got us the rights to cover Danger Zone and even recorded a cover for us. And shallow as that may sound, this alone can be the difference between people giving *Top Gun* a look or not giving it a rat's ass at all. I'm dead serious. Look up reviews of every *Top Gun* game ever made and every one of them mentions its lack of the song. Some reviews even list it as one of the overall "cons" at the summary of the review.

During development, just as a placeholder, I threw in the original version of the song, because it was a heinous crime not to. When word got back that we probably wouldn't get the rights to the song and we finally caved and took it out, I was actually really crushed -- heartbroken even. Not to say I didn't like the stuff we licensed instead, but it suffered quite simply from the mere fact that it was not Danger Zone. Nothing quite lived up to revving up your engine while listening to the howling roar of the original soundtrack.

While Paramount negotiated for the license to use a re-recording of Danger Zone, we played things touch-and-go while heading into the twilight of our project -- until our producer Bruce got word that Paramount had secured the rights and commissioned a cover. It was a glorious day at Freeverse. Not only was the song added, we shoved it into overdrive by adding an easter egg by which, if your pilot's name was "Danger Zone", the game played nothing BUT Danger Zone.

2. Engine Reuse

When we got the deal to create *Days of Thunder* and *Top Gun*, I made the decision early on to architect the code in such a way that it maximized code reuse. Much of the application architecture already existed in our iPhone engine that was used in *Moto Chaser* and *Big Bang Sudoku*.

Any additional code and classes would have to be designed to accommodate the requirements of two very different games. This paid off big for us in *Top Gun*, as many game logic classes were already in place, as well as a few flexible, reusable code elements. This allowed me to get a prototype with a jet flying and shooting down other jets in under a week.

I realize it's not incredibly common to know what your next game is going to be while you are still designing your current game, and this did require both games to be mostly visualized before much code was laid down for either game, but having these benefits allowed us to piece together the main elements of *Top Gun* quickly and spend our time focusing on tweaking the gameplay, optimizing the engine, and adding some kick-ass eye candy.

One example of a similarity between the games is that they both involve vehicles speeding around some fixed curve in space. *Days of Thunder* takes place on a track, and the action in *Top Gun* is centered around a curve in the air. So for *Days of Thunder* I created a system that parses an .svg curve into a piecewise bezier spline, around which our track was procedurally generated.

Cool little system, and it effectively turned Adobe Illustrator into a level editor, but the beauty was when it came time to do *Top Gun,* we took that same parser, had it parse a second curve for height, and removed the track generation. This, combined with a phenomenal proprietary level editor, allowed us to create our levels exceedingly fast.

---

3. Keeping the cheese and the unintentional humor from the movie

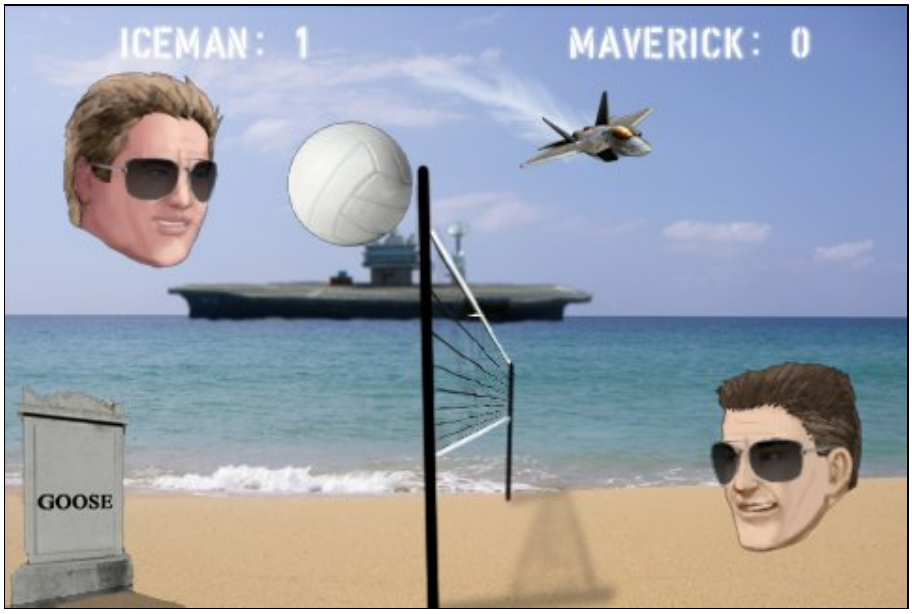
While it is still an awesome action movie, let's face it: Top Gun is cheesy as all get-out. Right out of the gate, the entire team was in agreement that we should both acknowledge and embrace the cheesiness. Paramount evidently agreed with us, too, when they gave us dialog to use that oozed with '80s machismo and corny come-on lines.

And in every gaming blog post, every review, every sneak peek, someone would invariably ask "But does it have a beach volleyball mini-game?" And I am incredibly proud to answer to each and every one of them:

"Yes. Yes it does."

It's tucked away as an easter egg, but if you enter "Volleyball" as your call sign, you will get to watch Maverick and Iceman settle their long-standing tensions once and for all. Goose's gravestone makes a cameo appearance.

It's such a tiny extra touch that 99 percent of the players will never see, but it got us on the gaming blogs again (a few dedicated a whole post to listing our hidden easter eggs and codes) and really, it just took a few people flipping out upon realizing that "yes, there is volleyball" to make the effort more than worth it.



4. Nailing the gameplay mechanic first

This might seem painfully obvious, but it was a major goal for Freeverse and Paramount to nail down fun gameplay before going gung-ho on content for *Top Gun* which why it is as fun as it is. As you read above, I had the*Top Gun* prototype going in about a week, and this allowed me tons of time to tweak every aspect of the controls and dogfighting to be as fun as possible.

I was able to try out several different functions for moving the jet relative to the accelerometer, and got to test a lot of values for missile speed, reload speed, lock-on speed, enemy firing rate, and tons more variables that all needed to click to get the game kicking ass.

The benefit to this is twofold. First, with regard to gameplay, having a wicked fun mechanic locked down early on allows you to use it as a foundation when designing levels, enemies and obstacles. I've worked on games where the mechanic hasn't been locked down, and isn't fun, and had these things built upon that flimsy foundation.

The problem arises when you try to then go back and change the mechanic after the fact. You run the risk of making a lot of the levels and obstacles worthless (for example, maybe it's more fun if you jump twice as high and move twice as fast, but this makes all those pits you put in a level all way too small to present any challenge whatsoever.)

You also run the risk of not being able to change that mechanic at all, because of the time it would take to re-design the rest of the game around it, and you have to ship with sub-par gameplay. Thankfully, the mechanics of *Top Gun* clicked early on and we could feel good about making levels and enemies that complemented them.

Which brings me to the second benefit: When you are working on a game for four months (and sometimes as long as eight to 12) it does a lot for the morale of the team when the game kicks ass right out of the gate. When people say they want to develop games, what they really mean is, they want to develop AWESOME games. Being a part of an awesome game motivates and inspires the other people on your team, and they in turn produce better work.

A fun gameplay mechanic can be the difference between "eh, it pays the bills" and "I'm staying until 10 PM some nights because the game is going to be so freaking TIGHT." This applies especially to QA and testers: when the game is fun and people actually WANT to play it and beat it, testers will experiment with more ways to best your game, and find bugs and loopholes you never knew existed.

5. Having an awesome brand and an uncrowded market

This might seem shallow, but if you're going to make a game based on a brand, the brand should kick ass. And basic marketing says that if you want to be visible, you need to pick a market that doesn't already have tons of products all fighting for the top spot.

When we began initial design work for*Top Gun* and *Days of Thunder*, I was very familiar with Top Gun, but hadn't even yet seen Days of Thunder. I had to go watch it as a crash-course.

And people who *haven't* seen either movie are more likely to know what Top Gun is about -- it's a wicked strong brand. The Top Gun anthem is known far and wide. There are plenty of Top Gun games out there already, and many parts of the movie are iconic (Danger Zone, the "need for speed" line, among others).

Top Gun also had an advantage over*Days of Thunder* in the market because of the history of car racing games. While we were working on *Days of Thunder*, there would be a new racing game announced once a week without fail. With*Top Gun* we didn't have to worry about positioning at all, simply because there were no games like it on the App Store at the time.

What Went Wrong

1. Not using the same model jet used in the movie

Top Gun is an iconic '80s movie and features an iconic jet. All the trainees in the original movie fly this particular model aircraft. When we developed the game, we decided to not use the same jet from the movie, but instead, a more modern model that is currently in service. Big mistake.

Yes, the game takes place quite a few years after the events in the movie. Yes, the particular model jet used in the movie is now completely retired from the US Navy. But those facts shouldn't have mattered.

I think the final decision to use the new aircraft was a result of that 3D model looking the best in-game. And when we learned it's not even used by the Navy (haha whoops) we classed it as a prototype and just acknowledged that in the dialog.

But people didn't care; they wanted the jet from the movie. And in retrospect I totally agree. We eschewed reality in every other aspect of this game in favor of awesomeness, but when it came to the jet we tried to rationalize not having the same jet as the movie. It's retired, it's old, the new one looks cooler, etc etc etc, but what we should have done was just continued the trend of fan service we had going.

The kicker is, we have the old jet in-game. Iceman and Maverick fly it in the missions they appear in. But it was in there as a throwback and not the main aircraft.

The bottom line is, we didn't give players what they wanted, and not featuring the same jet as the movie was the most frequent complaint we got about the game after its release.

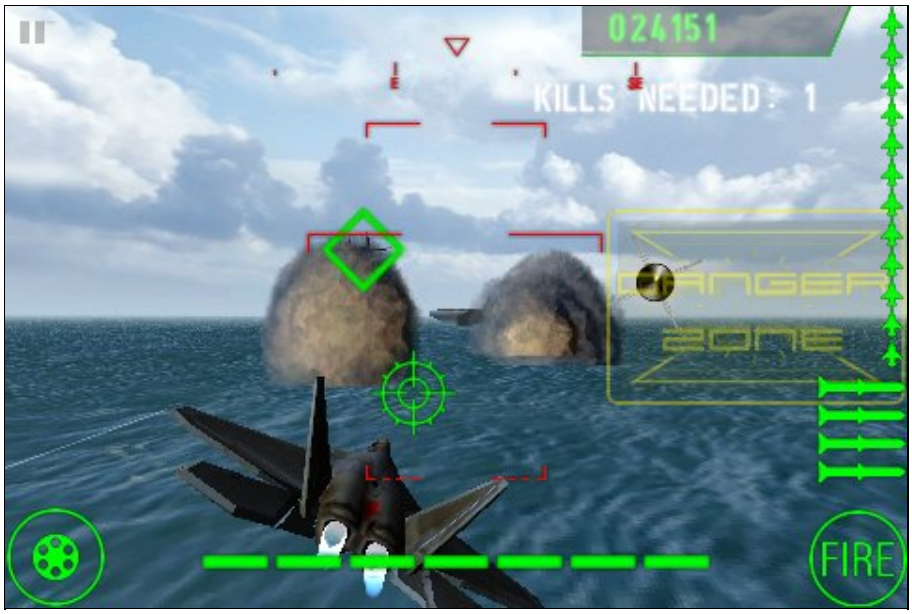
2. Ending on a cliffhanger

We also sinned big time when it comes to the ending of the game. In fact, in my opinion, we committed two separate sins. The first was ending the game on a cliffhanger. I think the story escalated tension and spun a few different arcs very well, but instead of reaching resolution, they all end at the end of their second act.

Originally, our story spanned 16 levels, and when we had to condense the game to 10 (due to time constraints) instead of condensing the story, we decided to leave it open for a possible sequel. Not sure I would have gone along with that again in retrospect, as the result is the end of the game has no story payoff. It still ends with a climactic battle with an enormous boss, so there is a gameplay/difficulty payoff, but the story remained unresolved.

Speaking of no payoff, the other sin we committed was having a quick ending sequence. After the final mission, the game simply cuts to a "To be continued" screen, then kicks back to title. This one still kills me, because I hate when games do this to me (I remember playing through over a hundred levels in *Rampage* for the NES as a kid only to get the word "Congratulations". Total BS!)

We just did not have enough time to complete that portion. I had a whole credits sequence visualized too, set to Cheap Trick's "Mighty Wings". If only we had a little more time (not to mention the rights to the song).



3. Not making it perfectly clear we were not a dogfighting game

Not featuring the jet from the movie might have been the most frequent complaint, but the most *vocal* complaint was from people who hated the fact that the game isn't a full roaming 3D dogfighting game but is, in fact, on rails.

At no point was the game ever intended to be free-roaming 3D. I've played a few games that do that, and frankly, I find games like *After Burner* and *Space Harrier* much more engaging and exciting, and that's the path I took when designing *Top Gun.* Dogfighting in games tends to be much slower, strategic, and frankly boring. I much preferred the sense of speed, tension and chaos you get with a game like *After Burner*.

And yes, not all players will agree with me there, and that's fine. Play what turns you on. The problem arose when those players saw *Top Gun* and assumed it was a free-roaming game, only to buy it and find themselves disappointed that it wasn't.

And they were right to be disappointed -- when your most visible tools for marketing yourself on the App Store are your title, icon, and first screenshot, it was tough to convey that to people. A few in-depth reviews on blogs like TouchArcade that explained to people that yes, the game is on rails, and no, that's not a bad thing, helped mitigate the confusion somewhat, but obviously you can't reach everyone.

The best solution in retrospect for this is tricky -- perhaps releasing more footage, or even a lite version. A trailer or even one level for free could have shown people that the game is not free-roaming, and if you really didn't want it if it wasn't, well, we could have saved you a few bucks. And with more footage or a free version, these people could have seen that it is still very fun anyway, and at least could have bought it when it was an informed decision.

---

4. Mistakes in level design

We made a few mistakes in level design that I will list for you here.

Not progressing the game mechanic or obstacles along with the story.

Two examples of this are your missile supply and enemy jets. The missile lock-on system is actually very customizable in code: the number of missiles you can fire at once, the speed at which they lock on, the speed at which they reload, and your bullet speed and power are all customizable, but at no point did we allow for it to be upgraded or changed.

This would have been a great place to allow the player to progress over the course of the game, allowing them to kill enemy jets with greater efficiency. Similarly, we added a few different types of jets towards the end of the project, to add variety to the enemies.

But instead of introducing these jets over the course of the game, they appear from the first level on. So instead of having new enemy types and new abilities be rewards that are unlocked over all the levels, we blew them all on the first level.

Having three separate level designers each working independently on their own subset of levels.

Because we were short on time, and because making the levels was the "fun" part, the level design was split up to myself and two other guys. While this allowed the levels to be completed much faster, it resulted in a less consistent level progression.

There are three noticeably different styles of levels present in the game. These styles include level difficulty, cutscenes, and even the personalities of the characters expressed through mid-flight banter.

For example, I tend to value challenge more than most people, and my levels are noticeably harder. With a mission that lasts only a few

minutes, I think it's perfectly okay to have a mission that players will fail once or twice. The result is that you might find a particular mission that I did to be much harder than the next few missions after, which were done by others.

Another example is when the characters reference the enemy during the mission. Some of us followed suit with the movie and never explicitly mentioned the origin of the enemy jets (which is why you never see it in some missions, and the cutscenes) but in other missions the pilots will call them commies or drop Russian references. (Even though I still maintain the enemies in the movie were not Russian at all, but North Korean -- but oh well.)

5. Leaving no extra time for testing

We scheduled the project in such a way that we were making levels right up until the deadline. While we did make sure to test all the levels upon completion, we spent the vast majority of time testing the stability of the levels, and simply whether or not they worked. Whether or not a part of the level was too easy, too difficult, or too boring -- these were all things that received less focus than they deserved, and we ended up shipping with a few annoying gameplay issues:

At the end of the project, the issue of "hours of gameplay" was raised.

To address this, instead of adding more levels (which we had no time to do) or adding more gameplay elements, such as leveling up or unlockables (again, no time) we instead simply raised the number of jets that you needed to kill in each dogfighting segment.

Where originally you would have to kill 10 jets in a dogfight to continue, you had to kill 30. While it added a few minutes onto each level, it also killed the pacing of levels and made the dogfighting sections more tedious, as the limit of even my patience is about 20 jets. When I say it was changed at the end of the project, I mean in the last few days.

Enemy difficulty balance was off.

A few enemies and targets don't make themselves vulnerable for very long, or stay out of targeting range for absurdly long periods of time. One example of this is seen in the gunboats in the third mission. In Mission 3, you have to take out 10 gunboats scattered around a few islands jutting out of the water. The problem is that a few of them are only visible and therefore targetable for a moment. If you miss it, you have to make another pass at the island.

The larger problem is that the person that tested the fun factor of this level was me. I knew exactly where the boats where, blasted them all out of the water, confirmed the level worked, and moved on to the next level.

Similarly, the huge bomber boss jet, which uses the same movement patterns as the small jets, has a few targetable points on the wings and tail that are off-screen for large stretches of time. The result of this is the player has to wait until the bomber swerves far enough to the other direction before you try to target those points.

Project Overview

Development time: 4 months

Development team: Producer, two programmers, one 2D artist, one 3D artist, one audio designer, our crack-QA squad, office cat

Platform: Apple iPhone/iPod Touch