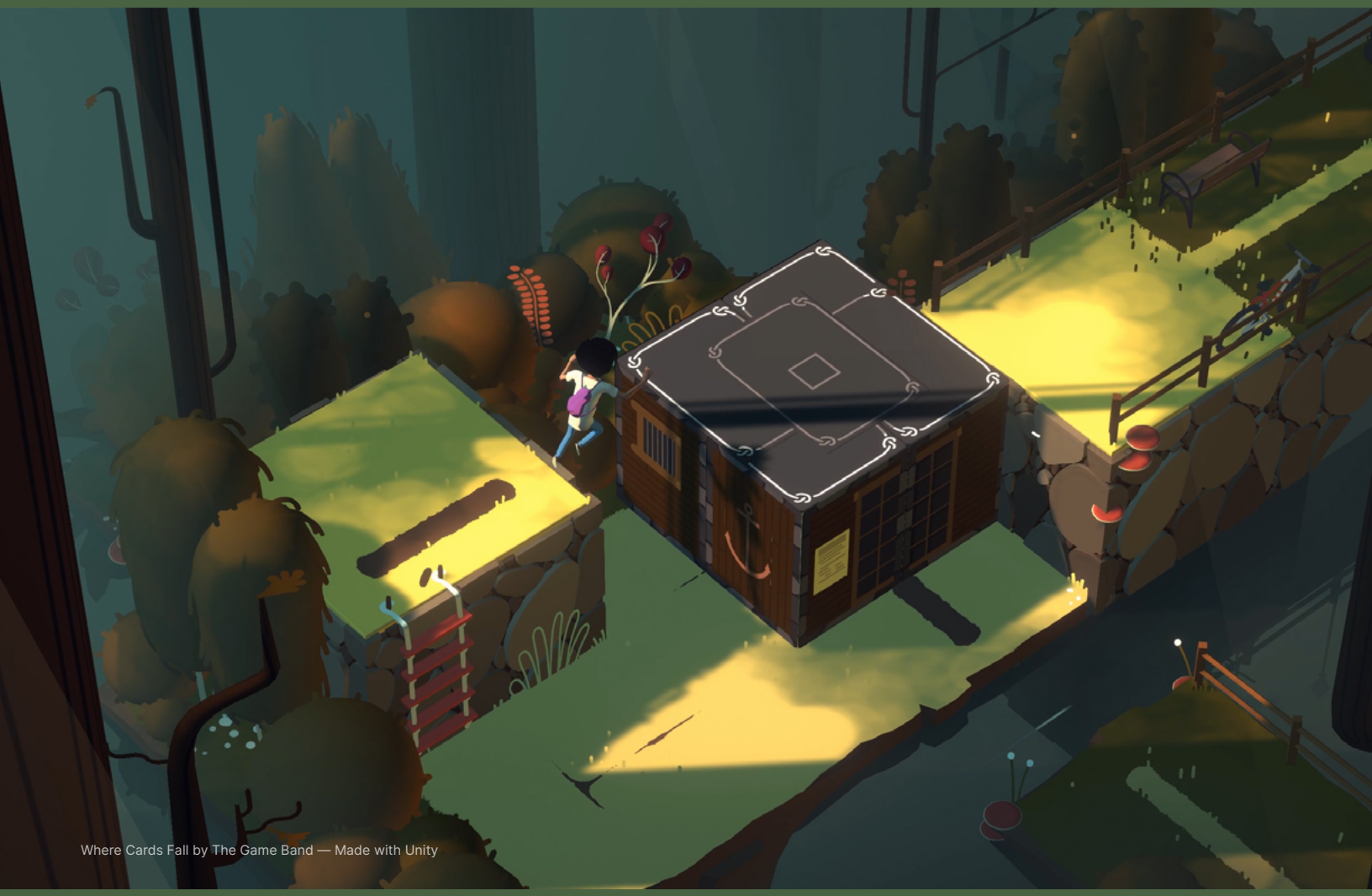


Where Cards Fall by The Game Band — Made with Unity

Top 5 game derailing mistakes to avoid

And the industry best practices that will get you to the finish line faster.

unity

Introduction

There are huge opportunities in mobile gaming right now:

- 49% of global gaming revenue will be from mobile devices by 2022.[1]
- 74% of mobile app store spending is on games.[2]
- 3.2B mobile devices are active globally.[3]

But there are also myriad ways to miss out on this highly lucrative opportunity. Game developers often ask us questions like: How do we know what platform we should focus on? How can we avoid development roadblocks? How do we ensure a successful, on-time launch?

To help answer these critical questions, a group of Unity veterans combined to address the 5 top mistakes every developer and studio should avoid. Based on industry best practices and our 15+ years leading the mobile game-development industry, this e-book will help you avoid the pitfalls associated with:

1. Improper planning

2. Not targeting the right platform

3. Wasting internal resources and using difficult workflows

4. Inadequate profiling

5. Over-engineering

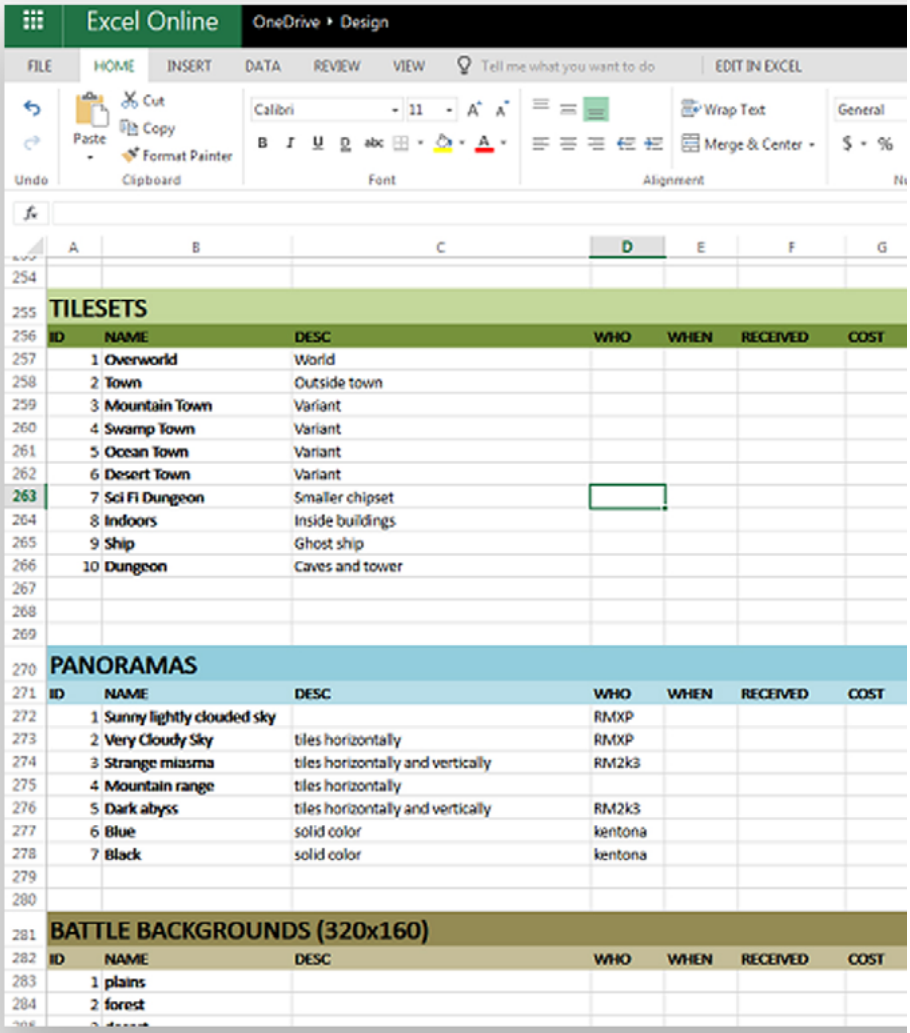
---

[1] Global Gaming Revenues Could Exceed $195 Billion by 2024 Across all Gaming Platforms
[2] App Annie: Games made up 74% of consumer app store spending in 2018
[3] Insights into the World's 3.2 Billion Smartphone Users, the Devices They Use & the Mobile Games They Play

Mistake 1 —

Improper planning

A game will only be as successful as the disciplined work and insight you put into planning. Here are the major factors that prevent proper planning:



Lack of detail: Many studios do not properly scope timelines, resources, and deliverables. There may be nothing particularly glorious about staring at a spreadsheet, but some things simply have to be calculated and documented ahead of time.

Ignoring financial realities: It's common for creators to get caught up in the dream, overlooking the importance of a timely return on investment (ROI). Chasing artistic perfection to the exclusion of business objectives is usually a costly mistake.

Missing the big picture: It's too easy to get distracted by some early innovative tech or cool aspect of the project. A prototype may be a stunning creation, but unless you plan for all its supporting elements, no one will end up playing it (or investing in it).

It's critical to keep the big picture in mind, guided by a detailed plan.

How to avoid it

Give your plans a backbone by prototyping before you grow your team

The key to successful planning is doing all the considerable prep work, researching and calculating, and then being prepared to possibly abandon it all as you adapt to unexpected opportunities and challenges. What gets a creative game developer through this paradox? How do you maintain a passionate impulse when you have to grind away at numbers and then watch that work blow away like dust in the wind?

Prototyping

Your original idea for a character or gameplay is what sparked you, and you can keep that spark alive to sustain you through the less entertaining aspects of day-to-day development. Prototyping lets you make the fun in your game tangible, it gives you and your team a solid reference point on which to base your development decisions. And you'll want to iterate any prototype quickly and often to create a unique experience for players that will ultimately help your game stand out.

From a planning standpoint, it's essential to also allow plenty of time and effort for transitioning a prototype to the official game-development framework.

Don't grow your team too quickly

Plan to add headcount as you actually need it, not while you're early in the prototyping phase. You don't need to spend loads of time and money growing your team when you're still tinkering with the foundational "fun." Once you've found that fun and developed a good prototype, you can focus on growing your team to build out the art, code, and levels.

Mistake 2 —

Not targeting the right platform

If you're going to put all your work and resources into creating a successful game, you must think ahead about which platform(s) you're building for. A lower-end device with a small screen, less power, but wider distribution? A high-end model with high frame rates and advanced textures?

All platforms have their advantages and disadvantages for different types of games and genres and, for that matter, different types of developers.

Prioritize the best platform(s) to showcase your game.

How to avoid it

Remember that a platform is more than just an OS

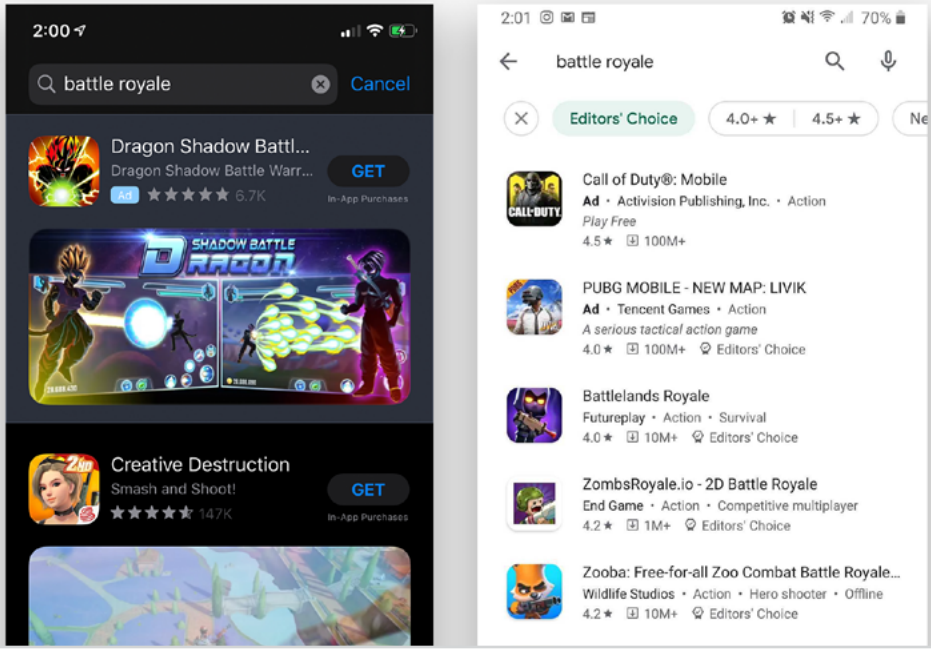
Targeting the right platform – or even targeting all platforms – requires carefully considering several factors. How do the particular characteristics of your game match up with potential models? What kind of competitive issues will you face? Are you ready to build relationships that can be invaluable to gaining a foothold in your target market?

Restraints

Many development decisions hinge on platform hardware restraints, their operating systems, and their minimum-specification (minspec) capabilities. Here are a few important points to keep in mind:

- Mobile devices are getting increasingly powerful and some are already comparable to a PS2. Yet device fragmentation issues, managing different mobile OSs and models, can be substantial.
- Whatever your target platform, it's wise to always develop for the minspec device – the oldest, least-powerful version of whatever's still in wide distribution. That device's bottlenecks will show up early in profiling and testing.
- What's hot now may cool off by the time you're able to launch. Anticipating trends is hard, but doing some research can help narrow your target.

Not all devices are created equal. Profile and test on the least-powerful target early.



Study where the market's going, and consider aligning with a major industry player.

Competition

It may feel like you're throwing yourself to the wolves when you look at the number of studios that seem to have games similar to yours. Just keep in mind, they all started where you did and they all probably felt the same way. You'll have an edge if you can consider:

- A platform may be growing rapidly, but you have to balance growing numbers with the potential for the intense competition that a hot device may attract.
- It may take multiple months/years to complete your game, so you need to try to predict the future. Study trends and try to figure out which platform will be most popular or best suited to your game when it actually launches.
- Your game genre is probably more popular on some platforms than others. Target a platform that has wide distribution but may host less competition for games like yours.

Relationships

Like it or not, success often depends on who you know. Manufacturers and big publishers staff entire organizations and teams that, if you pique their interest, will guide you technically and beyond to come up with a game that will make their platform or device look good.

Apple, Google, Nintendo, Microsoft, and Sony all compete to be a developer's best friend by offering everything from dev kits to co-marketing subsidies. If you do the research and reach out appropriately, chances are at least some will have a featured spot for a game in your genre. And remember that your personality and that of your team may dictate who you'll get along with best – though all are driven by hard numbers, some are more inspired by indie art, some by proven, formulaic builds, and so on.

Mistake 3 —

Wasting internal resources and using difficult workflows

Spinning your wheels on problems that have already been solved can ruin your balance sheet and demoralize your teams. You're going to eventually run into software roadblocks, and you may not have the resources in-house to cope with them. Plus, neglecting the day-to-day workflows for your developers and artists is a constant drain on their value and creates a massive lost-opportunity cost.

Keep your developers happy with simple workflows.

How to avoid it

Outsource where appropriate and make life easier for content creators

Occasionally, game development can be much harder than expected and can eat up disturbing amounts of time – even more so as you get closer to your planned launch dates. You can avoid a lot of anxiety (and costs) by taking advantage of external resources and by streamlining your internal workflows.

Engine-platform resources

Commercial game engines are enormously complex and that complexity has evolved, in some cases, over decades. It's no surprise that developers will come across problems either with their games or with using the engine that are quite baffling. Yet for an engine developer who knows the code intimately, the problem may be quite trivial.

Sometimes it's a lot more efficient to go to experts than to spend precious time hammering out an issue on your own. The most popular game engines recognize this and make their engineering resources available from simple support calls to full white-glove services. Some services cost, but it can be a fraction of what it costs to try to do it yourself.

Online resources

Several sites such as **GitHub** and the **Unity Asset Store** let independent content creators and coders post useful development content. This can range from full-featured tools that solve high-level coding issues to asset packages that enable quickly adding entire game environments and other features users may want. While convenient, there can be additional costs to using these resources.

For example, when pulling these packages off the web, be mindful of exactly what you're adding to your project. Typically, these kinds of packages include a lot of content that won't be used, whether it's a code routine or a graphic asset. After you know what you want, it's important to carefully profile the game and delete the things you're not going to use. Otherwise, irrelevant routines may run in the background or unused animations may load, slowing down other processes. These packages can be quite robust and solve many issues, but you may only need one or two features.

Workflows

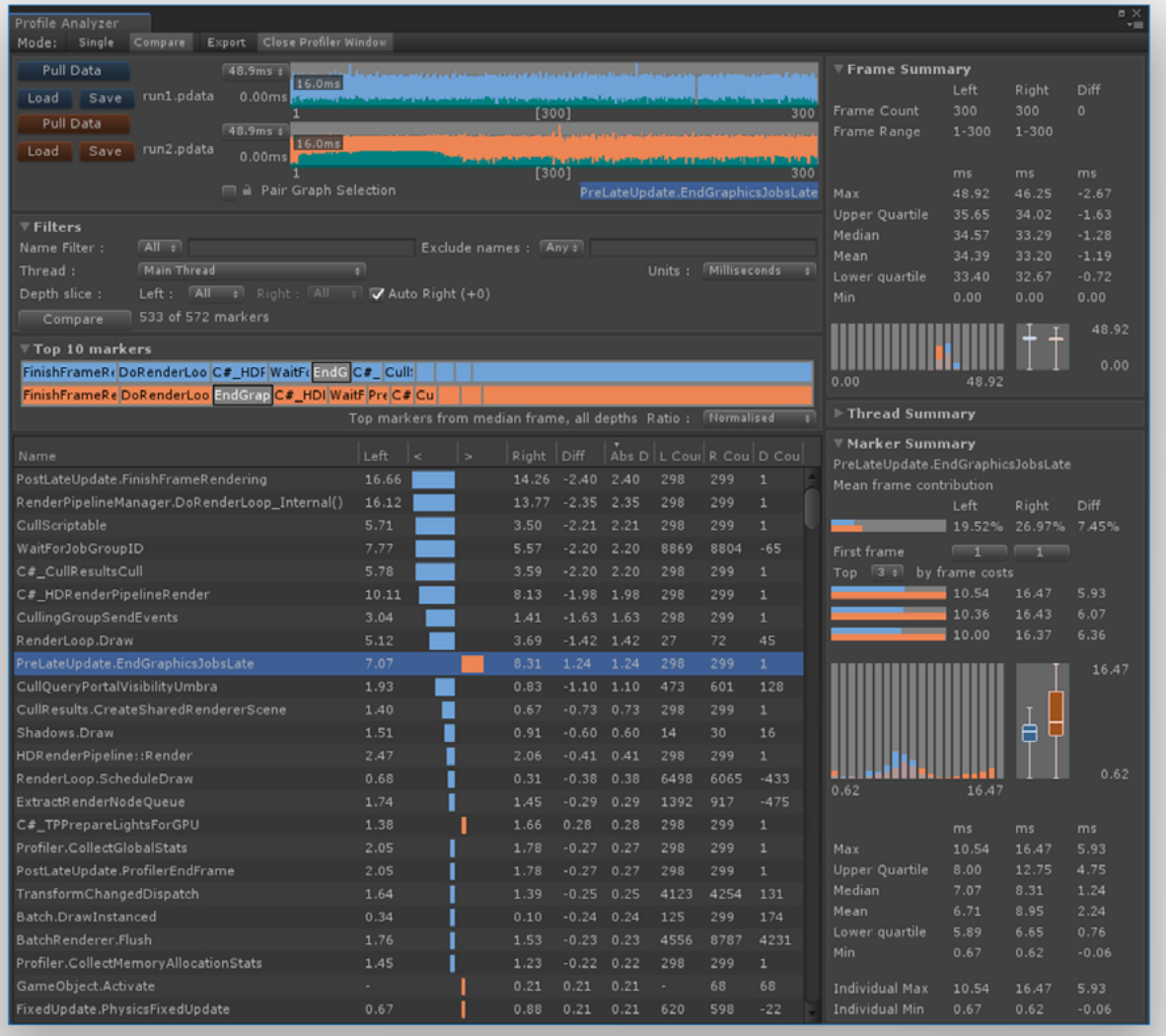
Optimizing workflows for content creators should be a top priority. The value of designers, artists, and animators increases proportionally with the number of iterations they make and how fast they can add them to your game code. It's crucially important to make their day-to-day experience of adding content – their workflows – as easy as possible technically. It will dramatically speed up design processes and development, and minimize delays.

Polygon – Fantasy Kingdom by Synty Studios, from the Unity Asset Store

Mistake 4 —

Inadequate profiling

Not knowing exactly how your code is performing (or misperforming) on your target platform(s) will, somewhere along the line, create immense frustration when you eventually learn you've got a problem. Any launch schedule will be cast adrift, and if you're already guilty of Mistake #2 (not targeting the right platform), you'll be adrift in very deep water.



Profile early and often.

How to avoid it

Profile early and often

To optimize your game's performance on any platform and overcome mobile device fragmentation, profiling is a must. It shows you how the CPU, memory, renderer, and audio components are all getting along, helping you catch performance bottlenecks early in the development process. If you profile correctly and often, you'll be able to spend more time improving features and the game experience. Test with the right devices – and test every day.

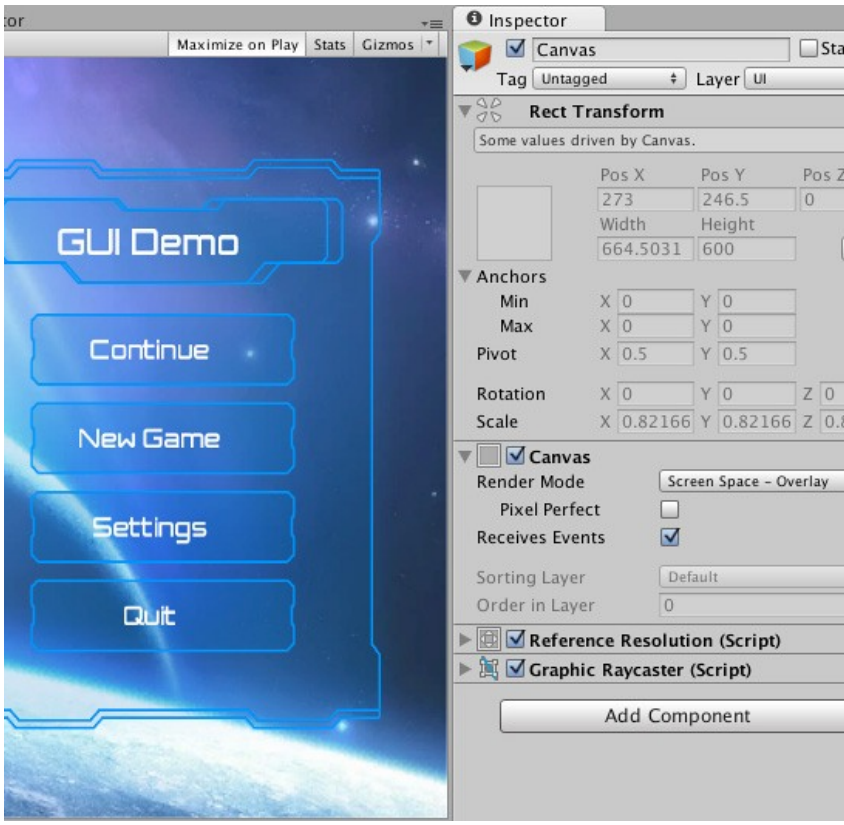
Minspec devices

There is a huge array of device manufacturers, chip sets, and features, and each device has different performance characteristics and capabilities. For any given platform, profile with the oldest, least powerful version that is still likely to run your game in significant numbers. And get a number of devices so developers, QA, and anyone involved can easily profile often.

It's also important to use not only your game engine's profiler but also whatever vendor profiler is available for your target platform(s). You'll see exactly what the device is doing with the memory you think you're using. For mobile devices, you'll want to use your engine profiler as well as Android Profiler in Android Studio 3.0 and higher or Time Profiler in Xcode Instruments for iOS.

Iterative testing

Every day of development typically means new or newly tweaked code and assets. If you don't profile for weeks, you'll have no idea what caused a change or issue. If you're doing it on a daily basis, you can quickly identify exactly which additions are causing an issue. Ideally, every engineer profiles what they're working on every day to check for scripting spikes and overall level quality.

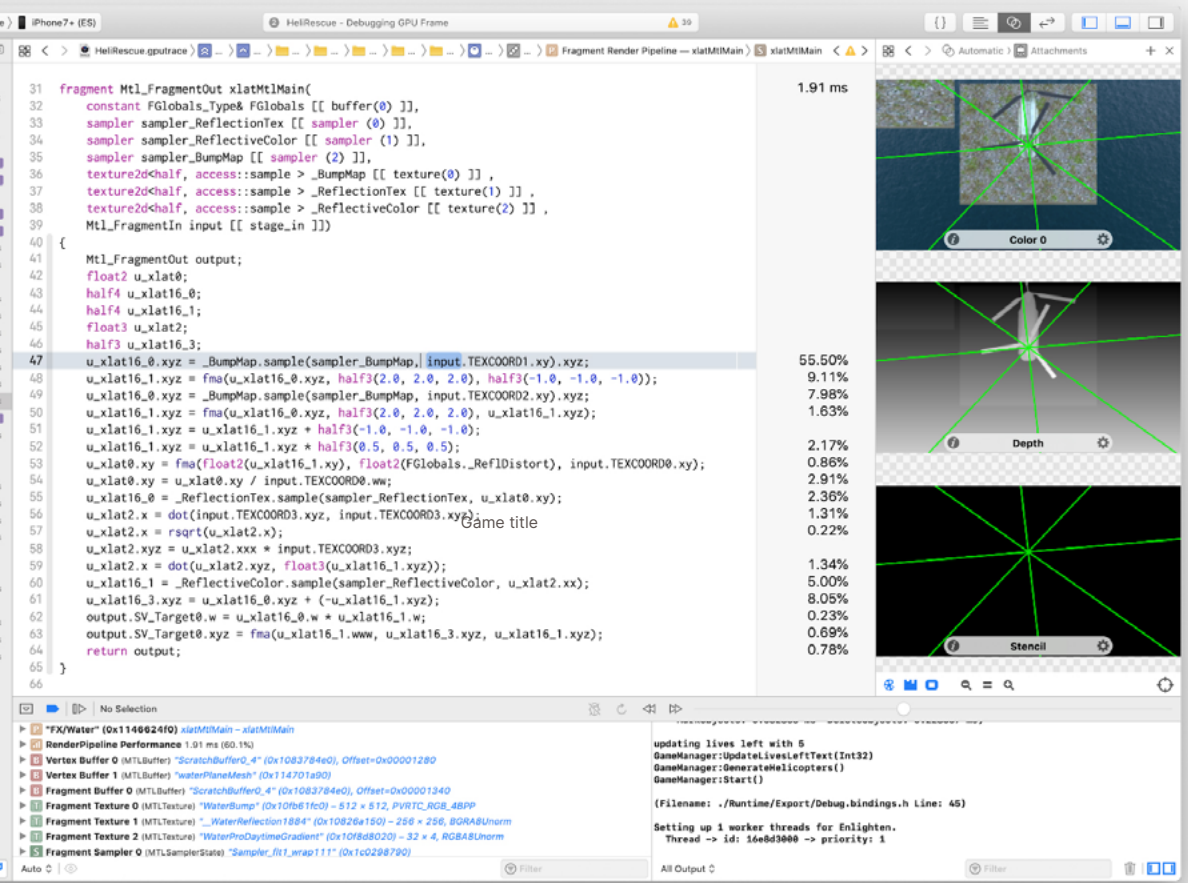


Test frequently, and across a wide range of devices.

Mistake 5 —

Over-engineering

Feature creep, or engineering for problems that your game doesn't have, diminishes the flexibility of your code and your development processes, and is often counterproductive. If a close look at your code shows functions that have little practical impact other than calling other functions, and your work is increasingly making things more complex instead of simpler, you may be over-engineering.



Don't let feature creep sneak into your project.

How to avoid it

Keep it simple

The opportunity window for a hot game only opens for a short time, and shortening the development cycle and getting the game to market quicker can make all the difference. So build the game you've scoped out and planned for, not every future variation you can imagine. There's a difference between leaving room for possible future features and building nearly finished, not-quite baked components.

Small, simple, and specific

If you over-engineer your system/code base, then as your teams scale up not everyone will be able to follow the complicated code base you've created. Keep the architecture as small, simple, and specific as possible so a new coder will be able to keep up. As with all software development, comment your code extensively and make your comments practical and helpful. And when the code changes, change the comments too.

There's a maintenance and performance cost to all code, so if it's no longer needed, delete it. A challenge is that if your code is thickly intertwined, pruning won't be easy, and you may be stuck with unnecessary code that's slowing your game down.

Don't get lazy

Sometimes, convenience is the enemy of performance. As engineers, we want the computer to do as much of the work for us as possible, but that's no reason to forego elegance. C# includes many attractive, powerful functions that will get a job done. However, unless you know the full extent of what those functions are actually doing, you may be extending load/response times with unnecessary logic. Convenience is often a wrapper for complex processes, and unless you fully understand what that logic is doing, it shouldn't be in your game.

Wrap-up

Create your best game and launch it on time

Now that you recognize the top 5 pitfalls of game development, you can focus on the following five maxims that will improve your team's productivity and lead to greater success in the extremely competitive gaming marketplace.

1. Give your plans a backbone by prototyping before you grow your team

Prototyping lets you make the fun in your game tangible, and you'll want to iterate any prototype quickly and often. Plan to add headcount as you actually need it, not while you're early in the prototyping phase.

2. Remember that a platform is more than just an OS

Many development decisions hinge on platform hardware constraints, operating systems, and minimum-specification (minspec) capabilities. Manufacturers and big publishers staff entire organizations to help you create a game that makes their platform look good.

3. Outsource where appropriate and make life easier for content creators

Popular game engines make their engineering resources available, ranging from simple support calls to white-glove services. Optimizing workflows for your content creators should be a top priority.

4. Profile early and often

To optimize your game's performance on any platform and overcome mobile device fragmentation, profiling is a must. Test with the right devices – and test every day. If you don't profile for weeks, you'll have no idea what caused a change or issue.

5. Keep it simple

Build the game you've scoped out and planned for, not every future variation you can imagine. There's a difference between leaving room for possible future features and building nearly finished, not-quite baked components.

Watch our webinar

For an in-depth discussion on the common mistakes that can impede your success developing and launching your great game, be sure to watch our webinar.

And contact Unity experts to learn how we can provide unparalleled support for any or every stage of your game development – from engineering advice and game server hosting to voice comms, ads and much more.