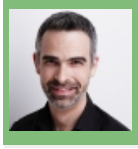
David Lightbown

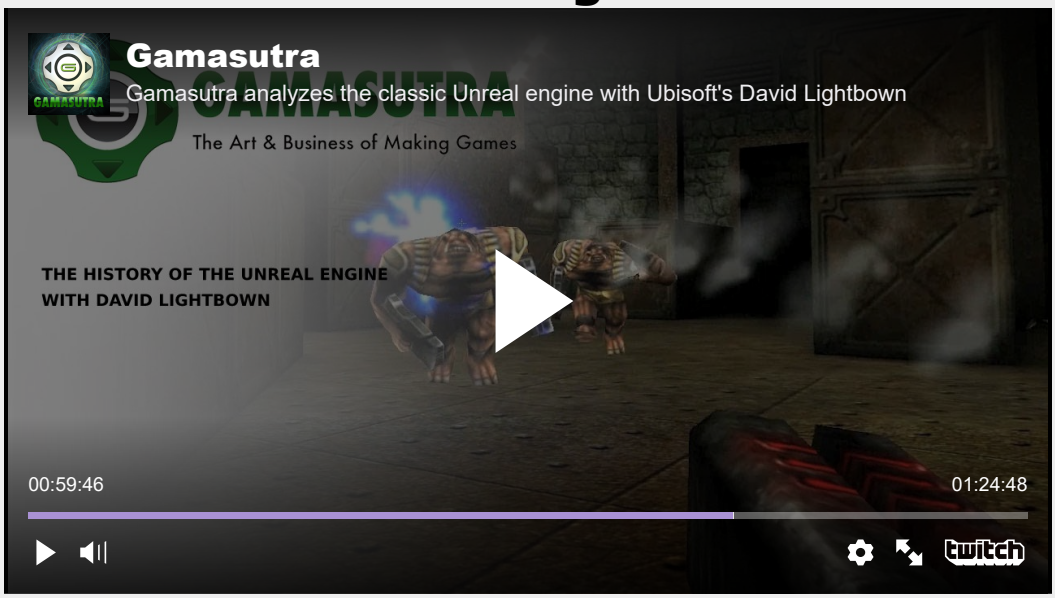


My Account
My Profile
Blog Now
[LOGOUT]

PROGRAMMING
ART
AUDIO
DESIGN
PRODUCTION
BIZ/MARKETING



# Tips for building game dev tools and UX from Ubisoft's David Lightbown




February 13, 2018 | By John Harris

February 13, 2018 | By John Harris

Post A Comment

More: Console/PC, Design, Video

In the metaphorical game development gold rush, there are the devs out hunting for gold, and there are the people making shovels for them.

If you are such a shovel-maker, you know that your task is a sometimes obscure one. Instead of developing software to be used by hundreds of thousands, you're trying to use precision-engineering to keep your developers in the zone and at their best game-making capabilities.

Fortunately for you shovel-makers, there's a champion for you in the halls of Ubisoft (and at GDC). He is, of course, the talented David Lightbown, and lately he's been talking to the people who developed some of the game industry's classic tools on Gamasutra.

Back in December, we were lucky enough to be joined by Lightbown for a conversation about the history of the Unreal Engine and the tools that influenced its creation. During our chat (which you can see up above), he also shared some useful thoughts about the work of tool development that we've transcribed for you down below.

**Stream Participants:**

**Bryant Francis**, Editor at Gamasutra

**David Lightbown**, UX Director at Ubisoft Technology Group

**Why We Study Game UX History**

**Lightbown**: I think that knowing the history of something is super important; if you don't know the history of something you're doomed to repeat it. I think that understanding the history of game development tools is really helpful as a designer, as a tools developer, to help you understand why certain decisions were made. And to also not try to not repeat those same mistakes.

storytelling at its best (Part 2)

## Press Releases

February 14, 2018

Games Press

- CONARIUM TO HAUNT MAC AND LINUX USERS
- THE SEVEN DEADLY SINS: KNIGHTS OF BRITANNIA PRE...
- NEON | Launching on Steam tomorrow
- Merge Games partners with Angry Mob Games to bring...
- Shoppe Keep 2 Character Creator Preview â€"...

View All RSS

## About

- **Editor-In-Chief:** Kris Graft
- **Editor:** Alex Wawro
- **Assignment Editor:** Chris Baker
- **Contributors:** Chris Kerr, Alissa McAloon, Emma Kidwell, Bryant Francis, Katherine Cross
- **Advertising:** Libby Kruse

Contact Gamasutra

Report a Problem

Submit News

Comment Guidelines

Blogging Guidelines

How We Work

Advertise with Gamasutra

## Gama Network

If you enjoy reading this site, you might also want to check out these UBM Tech sites:

Game Career Guide

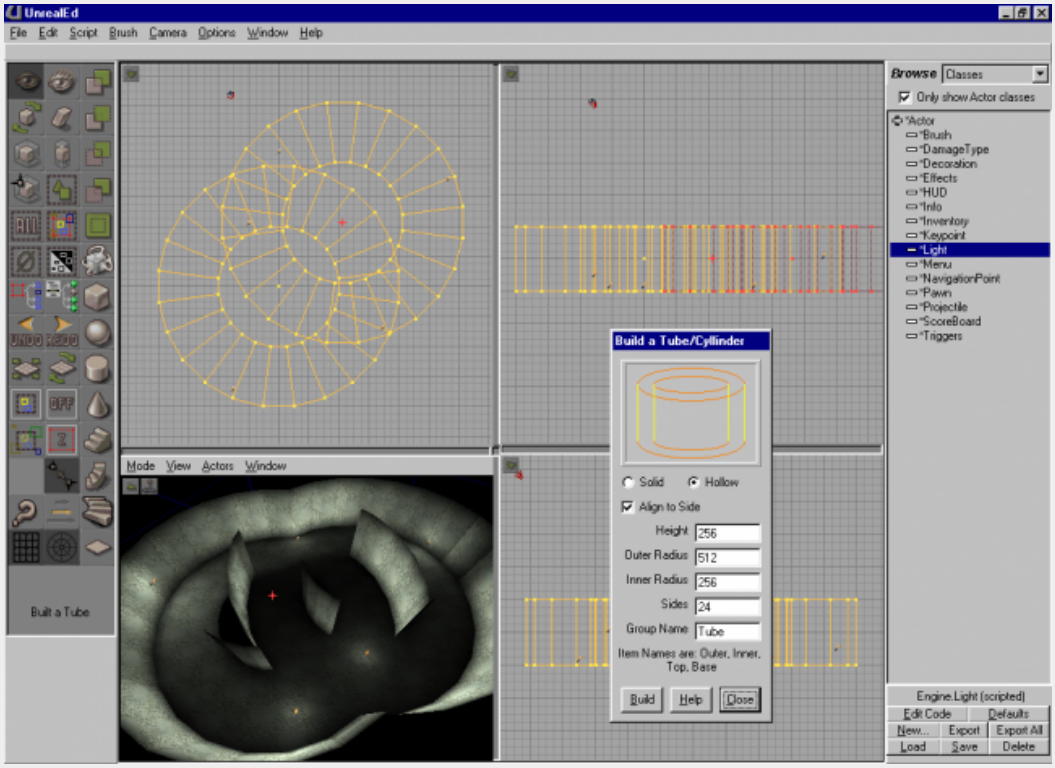
Indie Games

The reason why I got into this actually is that two of my other favorite books, *Blood Sweat, and Pixels* and *Dealers of Lightning*, about the history of Xerox PARC, and again, talking about the history, we're going back to the 1950-60s, when Xerox PARC was founded, and as some of you may know it was the research facility that came up with the GUI, with Ethernet, and networked printers and all this crazy stuff.

The framebuffer, some of the first computer graphics, editing an image was first done at PARC. And this is the GUI that was famously ripped off, so to speak, by Steve Jobs, then by Bill Gates. So knowing the history of this is super interesting and important, and looking at back at some of the GDC postmortems, just during the past two years, look at some of these amazing classic game postmortems. *Deus Ex*, *Oregon Trail*, *Seaman*, *Civilization*, *Pac-Man*, *Ms. Pac-Man*, *Diablo*, *Rez* -- and this year, I'm really looking forward to the *Bard's Tale* postmortem as well -- and I was looking at this and I was saying, "This is really cool, but nobody's doing one of these postmortems on game tools, that's never really been done before."

**"I think that understanding the history of game development tools is really helpful as a designer, as a tools developer, to help you understand why certain decisions were made. And to also not try to not repeat those same mistakes."**

So that was where the idea came from. And I did one just a couple of months ago, the article's released on Gamasutra as an interview I did with John Romero about *TED*, which was the tile editor that he used, which he created, and all the influences that were behind it, and how that led to the editor that was used to do everything from *Dangerous Dave* to *Wolfenstein 3D*, and how it was used and the history behind it. If you look it up on the Gamasutra blogs you can see the article there.




So it's the same thing as trying to retain this history and learn more about it and not making the same mistakes. My next step that I wanted to do was to contact a bunch of other people and see if they wanted to talk about their stuff, and that got me a connection to Tim Sweeney, which is how I was able to sit down with him at GamesCon this year in Cologne, Germany, and talk with him about the origins of *UnrealEd* 1.0.

**Taking Personal Research Into Day Job at Ubisoft**

**Francis**: I'm going to quiz you about your work at Ubisoft for a moment. We were talking earlier about the Ubisoft *Assassin's Creed Origins* cinematic tools. I'm curious, you've spent all this free time diving into the tool's history, we've looked at these four tools today on Unreal Engine... what, in your work on that specific tool, how have you linked your research with your work?

**Lightbown**: You know, it's funny actually, I can't really necessarily talk in too much detail, but I can say, if you are going to build a cinematics tool for a game engine, it is, I think, imperative that instead of sitting down and starting to write code right away, that you familiarize yourself with how these problems are solved in other software as much as possible.

So, in the case of a cinematics editor, go and look at Adobe Premiere, go and look at After Effects, any other non-linear editor. Even audio software. Ableton, Sonar, yes, they're for editing audio files, but the way in which they represent a timeline and how they let you drag and drop your elements and how you manipulate them. There is a consistency in some elements, of how you do that. I think it's so important to figure out what those consistent elements are, and to try to make your tool resemble those consistent interactions as much as possible, because if someone's used Premiere or any other non-linear editor, and they open up your tool, they'll be very familiar with how it works right off the bat.

Especially in my work, one of the things that I spend a lot of time doing is just research, being familiar with the tools that are out there, with their history obviously, but being familiar with how they work, and why they do things the way they do. Just because somebody does something a certain way doesn't mean that you should, you have to question some of those things sometimes.

**"It's imperative that, instead of sitting down and starting to write code right away, that you familiarize yourself with how these problems are solved in other software as much as possible."**

I kind of think about it as like, natural evolution. There are some species that have evolved a certain way, in a certain environment, and they've died off. And the same thing goes with certain types of software and interactions. If many interactions are difficult to use, that software might not have as much success being adopted and used by people, and then it dies off and the ones that have easier interactions are going to survive, and other people are going to look at those and evolve themselves off of those.

So it's sort of like survival-of-the-fittest, to a certain degree. It's not necessarily our job to reinvent the way that we use this software. Go out and look at how other people have solved these same problems, spend some time and I think it really saves you time in the long run, because instead of coming up your own idea, look at how other people do it, implement it that way. Your users will find it more familiar and also you're going to save time, as opposed of designing yourself. You have a great example right there that you can play with an understand how it works.

**Developing Tools For Other Cultures**

**Francis**: You were working under the Ubisoft Technology group. You're working, at least to my understanding, in the middle of a company that has studios around the globe. There are people who speak different languages. I think, as game development gets more global and grows, we're also dealing with the fact that different languages literally have different ways that words are structured. So as a person who makes tools for other professionals' use, if those professions come from another language do you have any thoughts about interactions for other languages and cultures, looking towards the future?

**Lightbown**: That's a great question. It's certainly something that I've thought about, that I've been asked before and I've done some research on this. My understand, again it goes back to what you're familiar with. There was a time when something like Windows was not made to adapt to other cultures. It was made with a specific set of cultures in mind, a specific set of languages. But it was used outside of those cultures, and the people who used it have adapted to it, and it has become what they are familiar with, it has become their "normal," so to speak.

My understanding is, you can have certain cultures where colors have a different meaning as opposed to what they mean here. But based on the research that I have read, my understand is that, in the context of a computer software application, people understand that this is different, that red, for example, might mean error. They understand that it doesn't necessarily mean something that is more applicable to their culture.

They are able to separate the two, and they understand that, in this context, these icons and colors mean this, and in my cultural context it's different. However, there is something to be said about developers understanding those people that is so key. Understand the people using your tools, understand what's natural to them, and try to adapt your tool to make it familiar to them. It'll make it all the easier and comfortable to use, they'll use it more and more, they'll tell their friends about it, and it can have a snowball effect from there.

For more developer interviews, editor roundtables, and gameplay commentary, be sure to follow the Gamasutra Twitch channel.