JARED: What is the correct number of hit points for the player character in a roguelike to have?

DARREN: Absolutely zero or one, if you want to think about it that way. Because hit points are just a terrible, terrible system. Who thought of hit points? I blame Dungeons & Dragons, personally. But you should do something more clever than hit points because there are so many more fun things you can do to make the player feel properly engaged with whatever's going on with the game rather than counting numbers all the time.

JARED: Welcome to Dead Code. I'm Jared. And today we're talking about something a little different. Well, I kind of feel like all of our episodes are something a little different, but it's something a little different that we have not talked about. And that is game development.

So, at the time of this recording, next week is the Seven-Day Roguelike Challenge. If you're listening to this the day it comes out, that will be the week of the Seven-Day Roguelike Challenge. And I've decided to compete, well, compete, participate is probably a better word there.

I have not done a game jam in quite a long time, but I've wanted to do one of these for ages. And I thought, what better way to get ready for the Seven-Day Roguelike Challenge than have some of the folks that host the Roguelike Radio Podcast on Dead Code and chat with them about Roguelikes, and where the genre came from, why it's so appealing to software developers, (because lots of people at least try to make Roguelike games, certainly not a majority, but I think it's probably overrepresented in the kinds of games that developers who try their hand at making games attempt.) and the Seven-Day Roguelike Challenge, and various other just sort of fun stuff about the genre. So, without further ado, let's get into it.

Picture this, you're staring at a screen filled with ASCII characters. What looks like a mess of random symbols to the uninitiated is actually a sprawling dungeon. The at symbol, that's you, moves cautiously through the corridors made of hash signs, past dot markings on the floor, avoiding deadly letters like H for Mind Flayer and V for Vampire.

You've descended to dungeon level 42. Your backpack is filled with unidentified scrolls and mysterious potions represented by various punctuation marks. Suddenly, your terminal flickers. The screen glitches momentarily, and the ASCII characters start rearranging themselves, forming a new pattern.

Two figures materialize on your screen: the first as an M, radiating an aura of procedural generation knowledge so powerful it would crash my 486. The second manifests as a D, surrounded by a cloud of seven-day game jams and design philosophy discussions. These aren't just ordinary characters in your roguelike adventure. They're Mark R Johnson and Darren Grey, two of the masterminds behind Roguelike Radio and veterans of countless permadeaths.

They've braved the dungeons of game development, wielded the plus-five sword of procedural content, and somehow managed to fit complete gaming experiences into just one week of

coding. Listeners, prepare for permadeath and don't forget to identify your potions before drinking them.

Welcome to Dead Code, Mark, and Darren.

MARK: Hi, thanks so much for bringing us on.

DARREN: Hey, thanks for having us.

JARED: So, could you guys introduce yourselves to our audience?

DARREN: Sure. I'm Darren Grey. I am one of the hosts of Roguelike Radio, a Roguelike-focused podcast, one of the organizers of the Seven-Day Roguelike Challenge, a game jam focused on making roguelike in seven days.

MARK: I'm Mark Johnson. I'm mostly the creator of a roguelike game called Ultima Ratio Regum, which has been in dev for 14 years and is an attempt to combine roguelikes with sort of cryptic riddle games, so things like Myst, or Riven, or The Outer Wilds, or Return of the Obra Dinn, and things like that. And I've also been a co-host on Roguelike Radio for quite a while now as well.

JARED: Awesome. So, I think the obvious place to start is with a question that has only one answer and has never been argued about which is, what is a roguelike game?

DARREN: That's very easy because no one has ever argued about it, as you said. Traditionally, a roguelike game has been considered a game that's in the vein of Rogue, which was an early '80s game, Dungeons & Dragons-esque game with random dungeons. You're a little at symbol wandering around the dungeon. And everything's randomly generated every time you play. It's got turn-based interactions with these dungeon environments, and monsters, and items, and everything randomized. So, it's fresh every time you play it. And if you die, everything is gone. Nothing is saved.

So, every time you're playing, you're at risk of losing everything you've built, which makes for a very tense, interesting experience. You never know what's around the next corner. It produces this wonderful feeling of being a real adventure, a real rogue, a dungeon.

Now, these days, that term has evolved a bit. Those core ideas of unpredictability, randomized content, having to restart again they've been applied to more than just sort of turn-based RPGs as they originally were. And we see elements of these mechanics in a wide range of games now. And indeed, I saw some posts on social media earlier today saying maybe we should start just tagging games on Steam as not roguelike to make it easier to sort through because it seems that the term is being used [laughter] very, very broadly these days.

So, elements of the genre are being plucked out and used in other ways and getting games with the sort of roguelike tag, especially in a lot of indie games. But the core ideas were this notion of a randomized run-through with Pyramid F, and traditionally, coupled with sort of RPG turn-based mechanics. So, you kind of...it's a more cerebral game where you're having to think through carefully what you're doing and coming up with clever solutions to randomized problems.

MARK: So, one thing, I think, that I'd add to that is, as you both hint to that, this is a discussion and a debate, and a definition which has seen a lot of discussion over time, especially as we've seen what people refer to as roguelikes shift away from that kind of classic ASCII/ANSI console visual style towards what we often now call roguelite or roguelikelike, which I like more, but most people don't, I think.

Games which began about a decade-plus ago with things like FTL and The Binding of Isaac and things of this sort, which took some of the kind of more abstract or more kind of higher level roguelike design aspects such as permadeath, quite a high level of complexity and also extensive procedural content generation as well, and then moved them outside of these sort of aesthetic associations, which they've been with up until then.

And so, I think for lots of people now, I think, increasingly, there's a lot of people playing these sorts of modern style roguelikes or roguelite games where it's more of those kinds of core design aspects, which people are focused on and trying to explore rather than the aesthetic aspects with which they were associated with for quite a long time.

JARED: So, Darren, you mentioned that the genre, or at least the label roguelike, is so popular that people were joking about tagging things not roguelike on Steam. What is it that makes this sort of punishing design philosophy so engaging for players?

DARREN: I think it's really crept in as something engaging for players over the last 10, 20 years as a counter-cultural element to...a reaction to the more narrative-driven, single-player, scripted, you cannot lose, press X to win style games. AAA has become so dumbed down in much of what it delivers that people wanted things that treated them like adults, that were hard, were punishing, which, you know, whack them about a bit and make them want to come back for more. That feeling in roguelikes of you lose and you know it's your own damn fault you lost, right, that really gets people to come back.

And it's such a change from just, here's a bunch of cutscenes interspersed with some quick time events, which a lot of games have become, especially in the 2000s and the 2010s. There's such a flood of AAA games that I think people found ultimately quite boring when it came to the gameplay side of things.

So, yeah, I think that element of, here's something that's giving me a real challenge, that is making me engage with the systems, that, you know, I have to pay attention otherwise I will die, that really grips people in a way that I think a lot of other games don't.

JARED: Yeah, I mean, in the context of, you know, thinking about the popularity of games like Call of Duty and their single-player experiences, you can put a lot of huge set pieces and explosions around a quick time event. But at the end of the day, you're just making sure you hit X before the timer runs out. And there's not a lot compelling there. You head back to the previous checkpoint if you die, and then you have to press the button on time this time, and there's not a lot there. These games are really the opposite of that in almost every possible way.

DARREN: They're mastery games at the end of the day. The best games, the most well-designed games that have...they might have randomness in the systems, but they don't rely on randomness to win. People get, you know, incredible win streaks in the likes of FTL, where they have, you know, 95%-plus success rates. But to get there, you have to master the system so well, similar to things like chess, say, where you get absolute masters of those games that know those systems so well. Or, in the video game space, similar to sort of multiplayer where, you know, at the highest reaches of multiplayer in highly competitive games, you get people who are insanely skilled due to how well they've practiced that game.

So, roguelikes offer a single-player version of that, where you get to engage with a kind of mastery streak where you know that the more you play, you may be dying and dying, but you're getting better and better at it, too. And that, you know, you get a real proper sense of accomplishment for that. It's not a fake achievement sticker that the game gives you. That's you knowing you've gotten better.

MARK: Absolutely, and those sorts of things were very much, for instance, how I found my way towards rogue. This was, as Darren said, in the kind of late 2000s/early 2010s. And I've been doing a lot of online competitive gaming for years. I've been making an income from poker for a while. And I was beginning to feel like I wanted to move on from both those things for various sorts of reasons, but I still wanted that kind of mastery-focused challenge.

And one day, I saw a thread on a big poker forum, saying, "Hey, if you like poker, roguelikes are sort of like poker, but single-player and offline, and you aren't having to deal with people or spend money." I thought, hmm, that sounds intriguing. And I went and checked it out, and found it had that exact same aspect of, yes, there's random aspects, but that doesn't mean that there's luck involved. It means that there's a very high skill ceiling in terms of trying to master how you deal with the set of random pieces the game gives you each time. And that was it, and I was hooked.

And I think Darren's completely right that a lot of this recent explosion of the genre stems from a sort of phase in more kind of "mainstream, in air quotes, gaming," which tended to lean quite strongly away from difficulty, complexity, master experiences and all these sorts of things. And I think that for lots of people, myself included, when they found roguelikes, they found, yes, this is exactly what I've been looking for, but without all that kind of multiplayer gaming toxicity stuff and something that I can play offline and in my time, but still explores those same kinds of ideas and builds those same kinds of skills, and still kind of demands the brain work hard in some of those same kinds of ways.

JARED: So, you mentioned you've been working on your roguelike for 14 years. Programmers seem particularly drawn to creating roguelikes. So, what aspects of this genre make them such appealing and long-term programming projects?

MARK: That's a great question. Well, I think, for me, part of the appeal is that even though I've wound up creating a very kind of graphically novel and graphically kind of complex roguelike, I think part of the appeal is that you can, to some extent, skip past graphics and go straight to gameplay. I think for lots of people, that's a core interest here. It doesn't mean you aren't creating something which is aesthetically pleasing, since I think, and I know lots of people think, that classic roguelikes do have a certain aesthetic console charm to them. But I think for a lot of people, that chance to focus deeply on gameplay, I think, appeals.

And also, I think because you don't have to have a graphics focus, it means that...I think a roguelike becomes a more viable one-person project than some other game styles, I think. And also, I think, in turn, because most of these games, at least classic-style games, are 2D rather than 3D, again, that kind of pairs down and kind of focuses what you're spending your time on, compared to maybe more kind of complex [inaudible 14:19] and things of this sort.

But, for me, I think a big part of the appeal embarking on such a big project was, well, in part, I didn't know it would be such a big project when I began, of course. And if I had, I would not have picked some weird, obscure Latin name to use for the game, that's for sure.

JARED: That's a classic case of, you know, we did it not because it was easy, but because we thought it would be easy.

MARK: Yes, exactly. Exactly. But I think, for me, I saw a genre where I did feel that there were big things which hadn't been explored. And whilst I'd done a lot of modding and things like that before I started to learn Python to make my game, I definitely saw this as being a much more viable route to being a one-person team who creates something genuinely novel. And also, of course, because at the same time I had begun to explore my academic career, it wasn't something that I had to do fast. It wasn't something that my capacity to pay rent and to have food and to not die...those things were not contingent on making the game fast or releasing things fast, and so on.

And so, it felt, A, like a genre where lots of cool stuff hadn't yet been done, B, a very accessible space to have things appear and create things quite fast. And also, for me, at least something where I could take the time to build something big and novel rather than having to kind of release a commercial indie product.

DARREN: I think also you mentioned programmers, and I think one of the things with roguelikes is, of course, the procedural generation aspect. And there's something just incredibly fun with playing with Proc Gen that you create this creator, and you play around with it. And, okay, you are doing it ultimately to make a game or part of a game. But it's just so fun to muck around with

these things just on your own time. And, you know, a lot of us as kids would have played around with things like Conway's life generator, little toy things like that, and had a bit of fun with that. Many of us have played with things like procedural music generators and so on that just are a delight to fiddle with.

And when you're making a roguelike and you make a thing and you click regenerate, and you click regenerate, and you click regenerate, and every time you get a new piece of content, and you're like, wow. And then you tweak it, and you change some variables, and you get a completely new thing. It's addictive in a way because you're always creating a new thing. It's not like, you know, you're drawing a picture and, you know, you draw it 10 times to get exactly the right picture for a piece of art for a game. Instead, you're sort of coding a system and seeing how that system will surprise you itself.

It's very satisfying to make something that will actually surprise you with what it produces. And a lot of the big games, I've often seen the creators of those big games be amazed at what the players can actually do in their games. They're like, "Oh, I didn't know my game could do that," and that's a wonderful feeling.

MARK: Absolutely, I think that's a key appeal there. There is something extremely pleasing, as my cook would say, in making something that makes something. It's very, very pleasing, and it really kind of gives you a sense that something quite novel and quite clever but also quite kind of tight and quite focused has been created there. And when you get into more complex systems, and as Darren says, the game creates a thing that maybe you hadn't completely known it would make, even though you'd been the person who'd made those systems, it's very cool and very pleasing as well.

JARED: You mentioned the systems there. Roguelikes have a tendency to be very, very complex games, and certainly, not all of them are but many are. Is emergent gameplay a big piece of roguelike games?

DARREN: Yeah, I think a lot of the classics have it. I think there's a sort of it works two different ways here. On the one hand, when you're making something with ASCII graphics or with very minimalist graphics, adding new things is always easy. On the other side, if you want to make a maximalist game with intensely complex systems, you've got to go down that route in the first place.

So, on the one hand, there's always the temptation in these systems to make something complex. And, on the other hand, if you want to make something complex, this is the way to go. Dwarf Fortress is the obvious example here, which has now got a graphical release, but it took a very, very long time to get there. But, you know, that game, if they'd started off with a 3D engine or something, it would never have gone anywhere near the complexity it has now. They had to start with a 2D, very limited graphical style to it, to get anywhere near the complexity that has been introduced into that game.

But yeah, I mean, a lot of the classics, I think, especially NetHack and a lot of its variations like SLASH'EM, I think to a certain extent, people at the time were like, "Let's just add this new thing," because these were open source. People were always chucking more on to the point where in some of these games actually, I think they had too much. For a certain period of time, we had a slightly counter-movement within the roguelike scene of making more minimalist games.

So, “Doom, The Roguelike” was a big example of that, of just cutting things down immensely. So, it's just concentrating on the core gameplay rather than on these loads and loads of more stats and more items and more everything maximalist design. So, there's a good variety actually in the roguelike scene now where you've got, on the one end, things like Hoplite and 868-HACK which are very tightly designed, and still the big giant games like Caves of Qud and NetHack and Angband and so on that offer just this hugely rich depth of systems to dive into.

MARK: Absolutely. And there's also some which still have that high complexity and that high master experience but don't have that much in the way of emergence, I think. If we're looking at classic roguelikes, I think something like Dungeon Crawl Stone Soup is a good example. It's still complex. It's still a hard game. There's still a lot of pieces, a lot of layers. But it doesn't have that same emergent systems aspect for the most part as something like NetHack does, where, in NetHack, so many kind of bizarre little edge case events or chains of events can happen to the player.

And part of becoming good at that game is both in learning what these can be and learning to kind of head them off in advance before they happen, but also part of the skill is if something bizarre happens, being able to stop and pause and say, okay, this strange 1 in 10, 000 thing has just happened. How do I respond to this? So, I don't think all roguelikes, be they classic or modern, have the emergence aspect. But I think that the complexity aspect is a really kind of defining feature of the genre.

DARREN: I'll give an example because I think for anyone who doesn't know about some of these sorts of cases in roguelike, it's good to have an example. In NetHack, cockatrices are an enemy type. And a cockatrice corpse, if you touch it, you'll get petrified, right? Which is not a good thing, okay, generally avoided. However, if you have the right sort of gloves, you can pick it up. In fact, if you have the right sort of gloves, you cannot just pick it up, you can wield it as a weapon and hit other enemies with it and petrify them because it's a petrifying item.

However, if you trip down the stairs whilst holding it, then you might accidentally touch the skin, and it petrifies you anyway. And so, all these kinds of different ways the objects will interact with you that can be unexpected and you can abuse in the right way if you are clever enough. But they can come back to bite you if you're not that further stage player right now.

JARED: So, the complexity being a really important part or often an important part of these games, what kind of technical challenges do programmers face when they try to make a roguelike, maybe their first roguelike?

DARREN: There are many different pitfalls, and I think the most common thing I see is people that decide, I'm going to make another NetHack. NetHack was made by teams of people over decades. You can't just jump in there. Or they decide they want the flexibility to do that, so they create something like an engine so generic that it's not focused on producing anything, right? And it takes them a long time to get anywhere close to interesting gameplay.

There's lots of useful game development systems and libraries out there now, which can help you get a head start on a lot of things. But the biggest pitfall is trying to copy the classics and going over the top in scale, which, to be honest, that's a problem across every single new game developer in every genre. They always think, you know, I'm not going to make a really tiny FPS. I'm going to make the new Call of Duty, like, no, that's overambitious. So, not starting small enough is the biggest challenge by far for the vast majority of people that approach it.

In terms of coding, I mean, you can go the other way and just dive in and make a spaghetti mess or, as Mark Johnson can probably talk about, you know, have all of your code in a single file for many years. I don't know if you still do that, Mark --

MARK: It's the only way to roll down.

DARREN: So, you know, not planning enough structurally with your systems and things. The best thing to do is to try and piggyback on some existing work. There's a lot of open-source stuff out there and most of the genre is built on open-source code. There's a lot of great engines that you can lean on, a lot of library sets that you can make use of that can give you some structure to build around and help you not make some of those early mistakes.

MARK: I think the biggest coding challenge in creating roguelike is laying good foundations, and this is something which I learned to my [inaudible 24:50]. Because prior to doing a roguelike game, I'd done a lot of modding and made some big mods as well. And with those, I never had the experience of my early code was bad, therefore, I had to redo it and go back to it because the later stuff built on it couldn't handle the crappy code I'd made at the start. I didn't have that experience at all, which I guess is because when you're making a mod, you're still based on the, one hopes, good code of this kind of commercially released game.

Whereas what I found was because I was creating a game while learning Python, the earliest code for things like world gen, or handling data, or printing the area around the player on screen and that type of stuff, that code I've redone maybe 3, 4, 5 times across 14 years. Because every time I get better coding, I look back and say, "My God, this is godawful, and this just has to be fixed, and this just has to be changed." And, in general, I'm not someone who kind of sees good code as a kind of aesthetic goal in its own right. I think if the code does what it needs to do and does it fast, it doesn't matter how ugly it is in my view.

But those early bits of code were not just aesthetically repugnant but also extremely slow and set me up for lots of issues down the line. And because roguelikes are games where there are

so many systems and each system talks to so many other systems, if you have some of those kind of base core foundational systems which don't work well, or are very slow, or handle data in quite kind of stupid ways, that's a real problem I've found because then all the later systems will struggle, or will slow down, or will have to keep calling back to this rubbish code.

So, I think, for me, trying to learn Python while trying to make a roguelike in Python I think if I'd been learning Python while making another kind of game, I think this issue would have been less pronounced, I think. I'm sure it would still have been there, of course, but I think it would have been less pronounced. But in such a systems-focused game, the fact that my early systems were written when I was still getting to grips with Python, as opposed to C and BASIC, I guess, were the two I'd done prior, because of that, a lot of that code was bad, and then that impacts every other system.

And so, although I'm getting better at not rewriting, as in sort of if you graphed how much time do I spend rewriting code, it kind of peaked in the middle of this 14 years, where I got good enough to know that the old code had to be fixed, but I hadn't yet quite got good enough to always write the best code first time through. But now it's become quite a rare thing because all of the old crap has been fixed. And I hope...I like to think I'm not writing new crap. But for such a systems-based game, the foundations have got to be strong in terms of code, I think, because, without that, everything will just kind of spiral and get worse and worse and more and more problematic the more stuff you add and the more complexity you build in.

JARED: So, you mentioned this idea of putting all of your code in one file. How big is this file?

MARK: Oh, here we go. Yep [laughs].

DARREN: Tell them the horrors of what you've done, Mark.

MARK: Well, that's a very personal question, Jared [laughter]. No. It's well over a million lines now. Now, to be fair, my game does some very novel kind of graphics creation stuff in that every item has PCG graphics, and the pieces for those graphics are stored as strings. And so, a large part of the code is strings, which are then stitched together to create graphics. So, a mil lines doesn't completely kind of tell the truth of, like, how much of that is code, rather than is, like, data, which is then used by code. But yeah, it's well over a million lines.

And it's one of those things where I've had no formal code training. I don't have a background in comp science. It was one of the things where I started like that because no one taught me better, and then once I learned better, I thought, well, it still works, so why change now? And it is a bit strange, I know, but the tradeoff is that I always know where every function is [chuckles] and where every piece of the code is.

At this point, it's become such a sort of infamous/notorious piece of bad behavior that I feel almost reluctant to change it now. I feel like the game would lose a core bit of its identity if I went and changed too many, many files of code. With that said though, I might change it one day. If

there becomes a clear reason to do so, I will change it. But right now, the game works great on this one file model. It's what I'm used to.

And I guess this also...I think it's one of the things where when you're creating a big hobbyist project, to some extent, yes, of course, you should follow what's good practice and learn from those who've gone before, and so on. But there's also a little bit of me which feels, yes, that's all true, but it's your project. It's your thing. And what works for you is not always what works for other people. And I realize this might sound like a kind of desperate defense of my awful code practice, and perhaps that's true, but I stand by it, even so.

JARED: Well, I fully endorse your approach. I do not want to inherit that codebase, or any like it.

MARK: [laughs]

JARED: But if it works for you, then I do not see why you shouldn't keep going with it until it starts crashing your text editor.

MARK: Well, that is always the fear, without a doubt. But like I said, it's got to a mil, and it works fine. So, watch this space. I will let you know when it does crash.

JARED: So, as we're recording, next week will be the Seven-Day Roguelike Challenge. It will be this week if you're listening to it the week that this episode comes out. I will be participating in the said Seven-Day Roguelike Challenge. What is it?

DARREN: So, it's a challenge to make a roguelike in seven days. That sounds pretty obvious, right? This has been going for...I don't know, actually. I should know because [chuckles] I'm one of the organizers. It's been like 15 years or so this has been going. And, hmm, it started off as a little bit of a joke back when most of the development community was on rec dot games dot roguelike dot development Usenet group, and most of us hung out there.

And it was during that kind of counter-cultural phase when people were saying that, you know, games had gotten too big and too bigly scoped and developers were just biting off more than they could chew all the time, and we need to focus down more on the mechanics and be more like the original Rogue, which was more limited in scope than the likes of NetHack and so on. So, it started off, I think, “Doom, the Roguelike” was the first one that was done in very short few days, and then a bunch of others decided, as a joke, let's all try to make a game in seven days. And I think there was probably about a dozen people at the start.

And the rules are pretty simple. You can plan all you want, but you shouldn't be writing any of your code before the week. You can use pre-existing code all you want, you know, it's open-source stuff, anything you've had from other projects, et cetera. But your new game ideas should all be freshly coded in the week. And within that week, you've got to put together a game. You've got 168 hours. See what you can do.

And it has led to just a flurry of inventiveness. And I think it's one of those things that, you know, restriction being the mother of invention, in that when you are told you can only make a small game in a short amount of time, you throw out so many preconceived notions of what should be in the game. And people have made such interesting ideas restricted around different themes, or, you know, there's a lot of games which are sort of, why don't I take this popular AAA title and make a roguelike version of it, Zelda or Metal Gear Solid, or something? And what does that look like in roguelike mechanic form?

And there's been a ton of just really super inventive, amazing stuff made over the years. We now regularly get several hundred people taking part every year. It's hosted on itch.io. There's, you know, it used to be more traditional roguelikes. These days, there's quite a lot of the roguelite, or the real tiny stuff, or stuff that's just sort of adjacent to the genre. That's all fine.

But yeah, every year, we see completely brand-new ideas that are pushing the boundaries of what have been done in games before and especially the different procedural content systems that you see. It's marvelous. It's an absolute marvel every year to see the creativity on display, completely new things done. And, you know, as well as that, a lot of people have just never made a game before deciding for the first time...roguelike is a very approachable genre. If you've never made a game before, you don't need to worry about graphics and sound design, and a lot of the other complexities in modern game development.

You can make something quite simple, and it's still enjoyable. And there's a lot of existing code you can lean on. So, there's people that just make sort of what I call baby's first roguelike, which is, you know, a very simple roguelike game that doesn't really do anything new, but, you know, for them, it was their first ever game, and they're really proud of it. And that's wonderful to see, too.

JARED: What have been some of the most innovative or surprising games that you've seen come out of Seven-Day Roguelike Challenges?

DARREN: Ooh, so many. So, 868-HACK, that was a Seven-Day roguelike originally, 86856527 was its original thing. And that was...it's a sort of cyberpunk-y hacking theme, where you're exploring a...I think it's a 6x4 grid, so it's a really tiny grid of tiles that you're moving turn-based between. But the tiny space is packed full of interesting decisions. Every single move really counts. That was great.

There was one a couple of years ago I really loved where each dungeon is inside a different animal. So, you sort of enter the animal, retrieve a key item, and go in through another animal. And you just got this kind of layered space where you're constantly inside something else. It was quite trippy and quite cool and just conceptually lovely.

I've had quite a bit of fun making games, and I made one once where you can only damage enemies with traps. And I made the main character have double the movement speed than normal enemies. So, you're constantly running around enemies that can one hit kill you. But if

you behave really cleverly, you can lay these traps and skirt around them, and it can make you feel both weak and clever and strong at the same time, which is quite nice.

MARK: One which really stood out to me a few years back, again, I can't think what it was called, was one where the player character takes up many tiles, and a big part of it is kind of shaping your character to fit through gaps and to fight monsters and so on. And I think there have been quite a few which have played with the kind of multi-tile player character idea. But those, I think, have been some very good ones.

And more broadly, I mean, even though small games are not quite my own forte, there's some fantastic stuff which does come out of out of the Seven-Day Roguelike comp. And, like Darren said, I think it puts me in mind of that quote that I think Roger Ebert said about Herzog, I think, that even his failures are spectacular.

And in a sort of same-ish vein, when I look through the Seven-Day Roguelike games, even those which aren't that well-rated or don't have that many players, I still often think, okay, there's still a cool idea here. You still came up with something novel, and something clever, and something worth doing, and maybe it didn't play out, maybe because of time, or because of code experience, or whatever, but there's still something quite clever and quite cool here.

And I think that's something which kind of stands out, to me, is that, yes, all of the kind of well-judged ones each year are guaranteed to be clever, and novel, and worth playing, but also, those which don't score quite as well still often have something quite clever or something distinctive in them, and that's a real boon of the jam, I think. I think that's a real highlight.

JARED: So, for someone participating in the Seven-Day Roguelike Challenge for the first time, some hypothetical person who's definitely not me, what's some advice you might give to this hypothetical person?

DARREN: Stay small. Every time I go into the challenge, I have three different scopes in mind. I have an idea of a game, and I've got, here's my ideal game I could make this week with all the scoped stuff. And then, I have in mind a cutback version of, okay, well, this would still be good if I don't have all that extra stretch stuff. And then, I have a really super cut down I'll be happy if I can just at least get this out. And inevitably, I'm somewhere between those last two, right [laughs]? So, have in mind these are cool things I'd like to do, but don't get your heart set on doing it because seven days can disappear pretty quickly.

The other thing I'd say is that seven days, although it can disappear quickly, it's also enough time to make something with some good polish if you keep your scope small enough. And a lot of people swear by the idea of making a game in 2 or 3 days and spend the rest of the week getting it really polished. And some of the best games that have been released have taken that approach, where they take their time at the end playtesting it, and balancing it, and making sure everything is really in a nice state.

I always intend to do that, and yeah, I inevitably get scope creep and go crazy and have very little time for bug fixing and balancing [chuckles] at the end of the week. But yeah, the biggest thing is keep your scope small. Try and focus it on one individual idea, like, here's a nice little twist I'd like to do, and focus everything around that twist. Pivot your whole design around a single core twist or kernel of an idea and make that one thing really good and interesting.

MARK: And one thing which I learned from the one time I have tried these, the Seven-Day Roguelike, is make sure that it's a week where the chance of being disrupted is very low. The one year which I tried it, I tried making one with Gregory Scott, who makes the Armoured Commander roguelikes, and, basically, both of our weeks just became so disrupted by other stuff that we just couldn't kind of get it done, in essence.

And although it's a sort of very mundane point, I think, especially if one's older, and has work, or has childcare, or things of this sort, seven days, on one level, seems like very little. But we both found that it was surprisingly easy to get knocked off our kind of rhythm. And then, once we got knocked off that, it was very hard to get back on it.

JARED: Perfect, because our hypothetical person has already booked the whole week off work and doesn't have any kids. So easy, got that covered. All I have to do is pick a small enough scope, and I've done some game jams before.

So, I think the one thing that I'm most worried about is something that happened to me in a game jam I did (It's got to be more than 15 years ago now.) where I had an idea, and I got it through to the basic core of the idea working and realized that it just wasn't any fun. And I didn't know what to do at that point, and I couldn't think of a way to make it fun. So, I just kind of [laughter] gave up.

DARREN: The other thing I'd say is that there's a Discord for the Roguelike development community, and people are super friendly there. They'll play your interim builds and things. They'll discuss ideas with you. It's a good place to hang out if you're wanting to engage with other people and if you get a bit more energy from doing it. I don't. I deliberately avoid it because I get sucked into it, and then I get nothing done. So, I just switch off all sort of external communication during the week, but for some people, it's really useful.

JARED: That's a good tip. So, we've seen a lot of success with roguelites over the past, you know, 10 years even. And we've even seen some more traditional roguelikes like Caves of Qud seeing a fair bit of success. Where do you see the future of roguelikes and especially traditional roguelikes going right now?

MARK: Ooh, well, I think things like Caves of Qud and Dwarf Fortress and so on, I think, they do show that classic style roguelikes or kind of roguelike games can have a huge appeal, in fact. But I think that for a lot of players, I think a lot of players coming into those games aren't necessarily roguelike players. I think a lot of people who come into Dwarf Fortress are much

more kind of simulation and management game players. And lots of people who come into Cave of Qud are much more RPG players.

And so, I think part of the success of those games is how well they've been able to combine the roguelikey aspects, which both make for a novel game but also appeal to people from the roguelike world, with aspects that one doesn't, by default, find within roguelikes, i.e. the RPG aspects or the management/sim aspects.

And so, I think for classic-style roguelikes to hit that wider crowd, there's got to be some other hook there into a more well-played genre of games beyond classic roguelikes. And that will be what brings people in, and then it'll be the roguelike-y aspects, which makes Caves of Qud or Dwarf Fortress stand out from your bog-standard RPG or your bog-standard management sim. So, I think if we do see other Classic roguelike-y style games which have that level of success, I think they will likely wind up pursuing a similar-ish path.

And then, outside of that, I think for classic roguelikes, which are, in air quotes, "just classic roguelikes," I don't think those are going anywhere, indeed quite the reverse, I think. But that community is growing because every time someone plays a roguelite game, there's a one in x chance, where x is quite a big number here, but there's a one in x chance that from that, they seek out classic roguelikes, and then they play those, and they find they enjoy those. So, just in terms of numbers, the volume of people enjoying classic roguelikes has climbed, even while the volume of people playing roguelite games has skyrocketed in the last decade-plus.

DARREN: I also have to wonder what we will see emerge from just the general indie development community because, often, indie developers who haven't traditionally taken part in the traditional roguelike community come out with some amazing things that surprise us all. And we've had great games with FTL and Invisible Inc. and so on that have taken roguelike-y formulas and turned them into their own twist, their own thing, but they're marvelous. I'm certainly looking forward to seeing more and more things like that.

Unfortunately, there's a little bit of a trend of just clone games, copy games going on right now where, you know, one formula proves popular, and so you get a hundred games just like it. There's, unfortunately, a lot of that going on. But for every hundred copies, there's the one interesting thing that does something new and starts off its own wave of new things, and doubtless, we'll see a lot more of that, more Slay the Spire, more Hades, more things like that that stand on their own two feet.

JARED: So, if listeners are inspired by our conversation today, where should they go to learn more about roguelikes or connect with the roguelike development community?

DARREN: So, I mentioned the Discord. It's very active. I don't know how to tell people to access the Discord. You just sort of find it, I guess. Search Roguelikes Discord. There's only one major one, really. On Reddit, there's a roguelikes Reddit, and there's a roguelikedev Reddit, which is incredibly supportive as a community, very welcoming, very open to all sorts of people, beginner

or experienced trying to make a roguelike or roguelike-related game. There's the Roguelike Radio Podcast. Some people might have heard of that [laughter].

MARK: [inaudible 46:24]

DARREN: Where we have many episodes discussing different roguelikes from the past and different design themes. So, we're a very development-focused podcast. So, we focus on the ideas that go into making games. There's the Seven-Day Roguelike Challenge on Itch. It's worth having a look at. There'll be a big flood of games in a week or so coming out of Itch. And then, I guess the other thing is major Roguelikes tend to have their own individual forums, or communities, or Discords where you get a lot of broad discussion, too.

MARK: And I think the only thing which I would add there is that there's also quite a few good and quite detailed roguelike dev blogs like mine and Cogmind's and so on. And I think that those are worth a look as well. Although there aren't too many of them at the moment.

JARED: Yeah, no, I've definitely found you guys' podcast as well as the roguelikedev subreddit and it's...and the FAQ Fridays are an absolute treasure of information about every single topic imaginable. Usually, if I have some question about some specific way something might work or I might implement something, I can either find an FAQ Friday that talks about it or at least one episode of you guys' podcast that talks about it, and those are always super helpful.

Well, thanks guys for coming on the podcast. Is there anywhere people should go to follow you guys online?

MARK: Well, I'm still on Twitter, at least for the time being, @mrj_games, and also on Bluesky with the same name, but without the underscore.

DARREN: I'm on Blue Sky as dgrey.

JARED: Awesome. Well, thanks for coming on the podcast, guys.

DARREN: Thank you so much for having us.

MARK: Thanks for having us.

JARED: One thing that I mentioned before we started recording but I don't think it made it into the episode was that I have a very serious number of hours put into, and this seems ridiculous to say at this point, but the Palm Pilot port of the roguelike Moria.

So, I got a Palm Pilot from my best friend when I was...I don't even know how old I was. I think it was sort of a hand-me-down thing. His dad used them for work and then when he'd get new ones, he'd give them to my friend. And then, when my friend got new ones, he gave them to me.

I think he gave me a couple over the years. Also, hi, John and Jacob, if you're listening to this. I think maybe they do.

And there were a bunch of different games that I could get on my Palm Pilot. This would have been in high school, I guess, but there weren't a lot. There was, you know, Breakout clone, things like that. But one of the things I could get was this Moria clone. I didn't know what a roguelike was at the time but this RPG game that was different every time. And I had to explore this dungeon, and I could do all this stuff. And it was really fun.

And I played a ton of it because it was, you know, one of the few games that had any depth that I could play on my Palm Pilot. And I think that's probably where I got into the roguelike genre. Hopefully, as you're listening to this, I am working on my roguelike game for the challenge. I'll probably be posting about it on social media, so head over there and cheer me on if I'm posting screenshots or whatever it is.

The week after the challenge, I will be doing a postmortem on my newsletter. So, if you haven't subscribed to my newsletter, head over to jardo.dev. There's a sign-up in the footer there.

This episode has been produced and edited by Mandy Moore.

Now go delete some…