Enduring Play Season 2 Episode 9 Cory Tsang

Sat, Apr 25, 2026 2:51PM 1:07:15

SUMMARY KEYWORDS

Game development, JIRA, project management, The Sims, Elder Scrolls Online, object engineering, production tools, agile methodology, task management, work-life balance, collaboration, automation, sprints, team dynamics, scalability.

SPEAKERS

Cory Tsang, Cheryl Platz, Enduring Play Computer

E **Enduring Play Computer** 00:00

Initializing Enduring Play podcast Season Two engine. Decompressing audio. Synchronizing waveforms. Reticulating splines. Launching podcast lobby.

C **Cheryl Platz** 00:13

Welcome back to Enduring Play, the podcast where we explore what it takes to create video games that don't just survive but thrive.

E **Enduring Play Computer** 00:22

Loading Episode Preview.

C **Cory Tsang** 00:24

The way I plan work is how I used to make objects in The Sims. That you can say, well, this thing has ideation and pre production and production. Okay, well, what goes into ideation? Who's involved? What are those tasks? And you can kind of drill down and solve for each of these different pieces.

C Cheryl Platz 00:46

I'm your host. Cheryl Platz, video game designer, director and author of The Game Development Strategy Guide from Rosenfeld Media. Enduring Play Season Two is about scaling our game development ideas beyond the individual. A common thread through most of our conversations is the collective: through communities of game developers, through game education, through games research, and through the events and systems that support the releases and people making our games. Few games are released in a vacuum. So how do we scale together?

C Cheryl Platz 01:17

Have you ever wished your production tools and process were more like the games you make. What if I told you configuring JIRA could be like engineering... Sims objects? Cory Tsang's unique and sometimes mind blowing path through games moved between engineering and production, bridging the gap between disciplines, and in doing so, enabling some of the world's most enduring game experiences behind the scenes. He both pre and post dates my own time on the Sims team having spent 11 years there, during which time his object engineering work drove a significant portion of the gameplay. Mid career, Cory's drive for change sent him to Elder Scrolls Online, where he found himself applying his analytical skills to project management to help his artists manage scale and complexity for one of the world's biggest live Service Games, Cory is now at thatgamecompany, but we won't be discussing that work today. The problem with being great at solving problems is it doesn't always draw as much attention as failures or even wobbly success. It's easy for folks to take talent like Cory's for granted when it is in support of craft done right. There's value in collaboration. There's value in a well crafted JIRA dashboard. We are better together. Now, I know there may be a few technical times here where folks who aren't in project management are tempted to dip out, but part of the point is that thriving teams need non producers, literate in this stuff, to write their own more meaningful starting tickets. So stick around and see what insights you can gain about why it's so important to build systems that leave space for that last minute work without crunch, about how automated handoffs make us all better, and how toilets inspired corey's work on Elder Scrolls. I'm going to get one thing out of the way here. We're going to talk about a tool called JIRA a lot. We'll explain it later, if you've never heard of it. But this is not an endorsement, nor is this a sponsored podcast. You might have a better way, and a better tool could pop up tomorrow, and the whole industry might celebrate. It's not really about the tool. It's what the tool enables us to achieve together. It just happens to be the industry standard. We talk about JIRA here to talk about why and how we use project management software to achieve better outcomes together. And every member of the team should understand that. Let's get into the game.

E Enduring Play Computer 03:14

Player one, Cheryl Platz, pronouns, she her. Player two, Cory Tsang, pronouns, he, him. Podcast level start.

C Cheryl Platz 03:27

Welcome back to Enduring Play, the podcast where we explore what it takes to make games that don't just survive, but thrive. I am your host, Cheryl Platz, and I am here with Cory Tsang, who is also in the Maxis alumni community. Here with me, and our careers have looked different, but we both have something in common, which is that we've both worked in production and we've worked in other disciplines, and that sort of Rosetta Stone experience means that there's something here in this podcast for everyone. Cory, thank you for being here with me today. I'm so excited.

C Cory Tsang 03:58

Yeah, it's my pleasure. My name is Cory, and I've been in in the industry, I think, 30 years this year, starting off as an engineer, moving through like a hybrid of design, and then I became a producer, and then kind of traveling through production, it was odd, ending up as an art producer coming out of engineering. But, yeah, I've seen things.

C Cheryl Platz 04:17

I love that so much, though, like growth mindsets. And, you know, one of the themes of this season is like community, and you being able to bring people together, like take your engineering mindset, bring art communities along. It's just so great. There's so many things to talk about today. So what I'd like to start out with is, usually I have podcast guests start by introducing themselves to folks as they would like at a conference. So how do you usually talk about your background when you meet somebody for the first time? Is it like, is that? Is that basically it or like, What projects are you most proud of from your past?

C Cory Tsang 04:53

Normally, I tell people I was on the Sims, if you played the Sims, I probably built at least 50% as a content in most of those games, and then I led the gameplay engineering team there... we called ourselves object engineers, and then I worked on Elder Scrolls Online for almost 14 years from before we shipped through the evolution from buy to play to having a monetization stream and just the constant updates.

C Cheryl Platz 05:19

Automatically that hits so many people. Like "If you play The Sims." I don't know, it's fine.

E Enduring Play Computer 05:28

Narrative cut scene: The history of The Sims.

C Cheryl Platz 05:32

The Sims is a brainchild of game designer Will Wright, originally released on PCs in the year 2000 from Electronic Arts Studio Maxis. Will was already well known for his runaway success with SimCity. Will's family lost their home to the tragic Oakland Hills fire in in the 90s, and it was reflections on the process of replacing his home and everything in it in search of happiness that led him to create the game. Originally, the Sims was more of a dollhouse, but that didn't provide much of a game loop. It is The Sims that serve as the ultimate barometer of your ability to design a home. Now, whether your goal is to design a good home or a horrifying death trap, that's on you, but even then, it was just a simulation of the people's relationship with their home until two additional designers, Roxy willisenko and Claire Curtin, joined will and started pointing out how compelling the relationship between the people could be. As Roxy told vice:

E Enduring Play Computer 06:22

"He showed me the object placement tool one day. It was cool, but I thought, actually, who are those people? What's the relationship to each other? You immediately begin to anthropomorphize these little creatures that started this whole new line of thinking."

C Cheryl Platz 06:35

Here, we not only see the value of drawing inspiration from lived experience, but broadening that initial inspiration to include the perspective of others. Will's original motivator of play self expression was made more compelling by the layers of companionship and immersion provided by the rich relationships playing out within the dollhouse worlds. To learn more about the motivators of play, check out my book, The Game Development Strategy Guide, which Will was kind enough to call "An amazingly broad and contemporary take on our industry." The Sims went on to change the industry from record breaking sales to equalizing the gender playing field in gaming and becoming one of the first games to depict same sex relationships. The Sims has the Guinness World Record for best selling social simulation series. Try saying that five times fast. With over 200 million copies sold across all formats, the current installment, Sims four was called by EA, the most widely played game in franchise history, with 70 million players. In 2025 the franchise celebrated its 25 year anniversary by re-releasing Sims one and Sims two to steam so you can actually play those games which Cory and I both worked on today.

E **Enduring Play Computer** 07:35

Loading podcast level one, seven generations of toilets.

C **Cheryl Platz** 07:41

I think I want to start there, because I think there's immediately that's going to open up so many questions for listeners for whom that was, that was a core memory. I think you've probably encountered this when you go out into the world and you start talking about that, like the deep emotional connection people have with the work that you've done. Let's talk a little bit about what it meant to be an object engineer on the Sims, because when we were getting ready for this podcast, I joked like that was user generated content before user generated content was cool. But once you make some, give some context for that, because the Sims was a very moddable game, but it was a moddable game because of the way y'all engineered the Edith system and object engineering system and and it was so revolutionary. So let's talk about that. How did we get to the point where people could add to the Sims?

C **Cory Tsang** 08:31

Yeah, the Sims. So Edith, I loved Edith, it was this development environment that was visual, probably before it was cool and it was modular. So I came out of college, kind of beginning where object oriented programming was a thing, and here was a system. It was a visual scripting language, but it was literally objects. You were dealing with toilets and beds and whatnot, and everything was self contained. And so you could think at a very high level, and then you break it down to give it the nuance. So I like to say, like every object starts with "I want to go to the object, and I want to do something, and then I want to leave the object." And then you kind of figure out, what does that mean to go to the object? Because sometimes it's a bed that has two sides, or it has a specific place that you need to go. And then doing things was great, because I could look at the Sim's personality. So the toilet, it's great. It was so much fun making that kind of like, over and over. I mean, I saw like, seven different generations of it. But, you know, having dirty Sims never flush, or married Sims always put the seat down. Or, you know, if you're a neat sim, you wash your hands before and after and, you know, just kind of mimicking and making fun of reality and just putting that kind of character into each of the objects was great. It was so much fun.

C **Cheryl Platz** 09:55

There's so much insight just in that one object type, the toilet. And when you talk about, "hey. The Sims brought their personality to that object." Did you represent that logic in the toilet itself? So it's like checking for the personality of the SIM in the toilet logic?

C Cory Tsang 10:10

Yeah. So it was a state machine, and you could look at the sim that's executing it, or the data that's on the object, and then you could just decide, so when we moved so we actually used the Sims one engine to make Sims three on the Wii, so we've just basically redid it. But Sims three, Sims actually had quirks and so many other personality traits that it really gave a lot of flavor to the behavior that was more than what we had in Sims one. But that was kind of like the magic of building the Sims is because everything was in the object, you could drop in new objects, and they would just fit seamlessly with the systems that exist, as long as, you know, we met certain types of things had interactions that met needs or they served functions. You know, when you prep a recipe, you would have to go to the fridge to get some stuff, go to the prepping surface in order to chop the things go to a cooking object, so you can actually cook it, and it would look for these things. And so you could create a new thing, like a food processor in an expansion pack, and then it just gets incorporated into these action sequences. So you didn't have to hard code anything. It was - the Sims was all about really simple systems that interacted in complex ways.

E Enduring Play Computer 11:26

Loading tutorial level: state machines and Edith.

C Cheryl Platz 11:32

Finite state machines were some of the most useful core concepts I got out of my computer science education. They're not actually that hard once you get the hang of them. It's a way of thinking about the world. An example given on Wikipedia starts with a two state turnstile, state machine where the turnstile itself can be locked or unlocked. Those are the states. When locked, the turnstile does not turn. But if a coin is deposited, the turnstile moves to the unlocked state. You transition between the states, and only then does a push action work while you are in the unlocked state, and pushing from that unlocked state transitions you back to the turnstile locked state. In the context of the toilet example, a toilet might have the states of lid down, seat up, seat down, unflushed, clogged in use, etc. And the types of actions a Sim can do to the object will be different and have different effects based on the current state of the toilet. For example, using an unflushed toilet without flushing it might move it to a clogged state. So what's Edith got to do with all this? Designed and implemented by Jamie Dornboss, Edith, which stands for either edit hierarchies, edit house or Edith bunker, one of the first NPCs, depending on who you talk to, is a visual scripting environment for object development on the Sims using "a box and arrow tree layout where each box describes an instruction and points to one or more other boxes to direct control flow", as in, you could visually see the state machine as you implemented it. Groundbreaking for Its time.

E Enduring Play Computer 12:57

Loading podcast level two, thriving engineering at scale.

C Cheryl Platz 13:03

Which is brilliant, and by building those simple systems that interact in complex ways, whether do you think, when you were building it, that you were, you were... you knew you were building to scale for decades? Like, Were you making these choices with the intention of "We have to build so that this thing can last for decades?" Or were you making those choices which were very smart engineering choices for some other reason?

C Cory Tsang 13:33

So I joined the Sims one on the second expansion pack, which was hot date. So Edith was used, although our team was only like 17 people and was really tiny. It was just three of us engineering and then four artists. It was super small, but the implementation - Edith was set. We still expanded it. So what you had was the scripting system, and once you kind of hardened what you wanted to do, you could take some of the more complex logic and then create what we call the primitive, which is an actual C++ call, and then that just gets incorporated. I know Edith has been documented online. So each node was a state machine, and you can go either, it's a primitive, which was like the go to primitive, to route Sims to a place or to play an animation, and then there were just other things, we called them trees, which could just represent an abstract thing that it's going to do. It had four variables going into it, and then it could access the state values of the actual object. And you'd go into it and step into into these nodes and then execute. Yeah, no, it was, I think we were, by the time we started working on those expansion packs, we knew that we were on a path of continuous additions, and then so we had to build things resilient to change and adding. Because we added into existing systems too hot date was the one where we actually took the simple social system and broke it out into a gamified social system where you're trying to unlock that kiss. In fact, the design of Hot Date centered around Tim LaTourneau saying, "I just want a sim to get mad at me and throw the drink in my face." And we built the extension pack around trying to craft that experience

C Cheryl Platz 15:18

Good to have a core loop! If the core loop is drink in face? Okay, that's the start. It's unfortunate we lost Tim recently, but that is a fun Tim story.

E Enduring Play Computer 15:33

Podcast level, paused begin host commentary.

C

Cheryl Platz 15:37

As one of the few humans that actually got to work on a Sims one expansion pack. I want to emphasize how astonishing this feat of sustainability engineering really was. That team was so small by many standards, 17 people did Cory say. And not only did they ship one of the world's biggest games, but they kept shipping it. The first expansion shipped about six months after the release, and then two a year, each year until the end of 2003 when the final Sims one expansion pack, The Sims Makin' Magic shipped, which was my first shipped video game. And I am deeply aware of how fortunate I was to be welcomed onto that team. And I need you to understand how they shipped the Sims Superstar on May 13, 2003. I went to the ship party in my first week of my internship, and by the time my internship was done, we were done, we had gone Gold Master the game was off to the factory to release fully on October 29. While there was pre production in the mix before I got there, I watched almost the entirety of that expansion pack come into being during my three months as an intern before converting. Sustainable from an engineering perspective, doesn't mean sustainable. From an hours perspective, we put in a lot of hours, and Cory will talk about that. At the time, I joined the team. Tim LeTourneau, credited as a co creator of The Sims by many, had moved on to the Sims two team. Tragically, Tim passed away in the summer of 2025 he was honored by my former manager, Jonathan knight in comments as a servant leader before we called it that Jonathan went on to say that at Maxis, Tim was famous for managing by walking around. And Jonathan remembered that Tim said hi to everyone, called everyone by name, and knew the power of a smile. May we all be remembered like that. I'll leave a moment of silence here to honor Tim and his contributions to the enduring Maxis family.

E

Enduring Play Computer 17:34

Loading Podcast level three, producing with curiosity and context.

C

Cheryl Platz 17:41

How has that ethos of the state machines and the scalability impacted your work as you shifted gears and you kind of moved on to Elder Scrolls and other projects?

C Cory Tsang 17:52

Well, part of the reason why I moved was just to try to make that shift into production being on the Sims as long as I was, it was 10 and a half years across multiple generations. Every time we moved the Sims one engine to another platform, I just got sucked back into engineering, and I lost my roles. And then moving to Elder Scrolls Online, I had to let go some of my instincts of design and engineering, because I'm not the one doing the work. I'm there to support the people doing the work. And it's actually funny, because funny, because I've encountered people who believe that you have to be a discipline expert to be a producer for the team, because they're expecting you to call BS on things like how long it takes them to do it, or what they're choosing to implement. And I think that's such a bad... or it's like the wrong approach, because our job isn't to fact check people. We- we're there to trust the team because they're the ones who are on the hook for delivering. They're the ones who actually have to do the work, and it doesn't matter if, if a lead can do it in half the time. How do you coach that person to get there? But just don't expect that straight out of the box, right? So as a producer, it was more about helping people work better.

C Cheryl Platz 19:08

That's a really good point, and I think we should unpack that a little bit, because I've been on both sides of that. Like I've been at small studios where the producer was also kind of de facto craft lead, and then I've also been situations where we had both craft leadership and production. What you describe is, I think that healthier model, which is, look, production is there to help enable the logistics of getting the game out the door, like making sure the math maths. Making sure that we, like all the tasks, are accounted for, but they're not there to maintain craft excellence. You probably have craft leaders for that, or mentors, or an engineering manager or somebody. They're the ones who are going to call like, that's a terrible idea, or, Why are you doing it that way? But it's kind of bananas to have somebody who's an expert at JIRA and an expert at gate reviews and an expert at those thing AND an expert at Unity.

C Cory Tsang 19:54

It's always somebody who can fill gaps, but it's not the role, it's more that.... Like I find where I can apply my experience is more around asking the questions to get people to think about things that they wouldn't have thought about otherwise. You know, providing context to the work and making sure that people understand that they're working toward a goal, versus just working on tasks. It's funny, I don't care as much about the individual tasks, because they're not for me, they're for the people doing the work, and it needs to mean something to them. And if it's too small for them, then it's too small for me. The importance is really in what are the questions and what are the problems we're trying to solve at a production level? When I was art producer on ESO, the art teams were supporting multiple projects, whether it's the dungeon team or content or marketing monetization pipelines, we had to be able to split our time across multiple efforts, and those efforts had different timing, and they were on different cycles, and so being able to see how they could merge those timelines into a Single schedule that can deliver the most that we can and still keep people working sane hours? I think that's the biggest thing with live services. It has to be sustainable development. Like there's no crunching to hit a deadline because there's deadlines non-stop.

E Enduring Play Computer 21:16

Podcast level, paused. Begin host commentary.

C Cheryl Platz 21:20

The abbreviation DD stands for development director. At Maxis, Ddvelopment directors are cross disciplinary leads who manage budgets, operations and milestones; acting as a liaison between developers and producers. Producers at Maxis were often more heavily involved in the creative side than at some other places I've been. After all, I got my start writing as a production intern on a Sims expansion pack. I also did tasks like asset tracking and research and an environment like that. It makes sense that having a sort of neutral go between to keep the project in check makes sense. I've seen creative teams crunch themselves without firm guidance on scope and success criteria.

E Enduring Play Computer 21:55

Loading podcast level four, the cost of crunch.

C Cory Tsang 22:01

The reason why I got into production is because on the Sims, at least on Sims one, and yeah, mostly Sims one, Sims two, console? We, we were crunching every three months for like, two months at a time, and it was really destructive to my social life. And in fact, when I became a DD, we had this policy, if you identified a problem, you had to be part of the solution. And so I noticed a certain pattern in our decision making that continuously triggered crunch. And so they're like, "Okay, well, now you have to do some DD work too," on top of what I was doing engineering wise. And we actually didn't crunch that project. It was SimAnimals 2. SimAnimals Africa, I think we did one Thursday until 11, and one Saturday, and that was it. And I ended up marrying my girlfriend. I kept on losing them on this cycle of I would disappear for crunch and they, you know, they would leave me, but I was able to hold on to that one long enough that, yeah, we ended up getting married. Three kids.

C Cheryl Platz 23:03

That's a happy story. I don't love the like, the all the crunch that built up, but, like, Right place, right time. But how much agency you had in creating the space for your own... that that that's a lot. But, boy, yeah. EA, in that era was very crunchy. And there is so much you said here that resonates with me, because, you know, I've talked in previous podcasts about the whole ea_spouse thing, right? Like that was, there was a whole vibe, and it wasn't just Maxis, it was all EA. But I didn't know about the sort of shift- "you have to be part of the solution" thing. And that's a very interesting culture, right? Because it almost penalizes the reporter of the problem, which we don't, we don't have time to unpack all of that. But it's awesome that you're talented enough to help solve that problem and get people away from crunch. And one of the things, yeah, very much- One of the reasons I left and went to Amaze Entertainment was because I wanted to run teams and prove that you can run teams without crunch. I was like, "I think I could do this." There was that "Open Seven Days" sign, the neon sign in Redwood City, above a certain person's desk that I'm not going to mention, that was in the same space, and I was like, "I don't think we need to do this." And there were certain techniques, and my teams didn't I mean, every once in a while something happens, right? A build breaks, or a client comes in, and is like, "Everything needs to be purple." You're like, "Okay, well, that I couldn't necessarily have foreseen that." But I love that ethos is like, treat people like people create space, right?

C Cheryl Platz 24:23

The other thing I want to go back to is the importance of leading, like, with curiosity and focusing on solving the right problems and solving goals, because that is really resonates with me as a leader. And it's something like, the longer I go in my career, the more I just like write that down. Like, literally, for the last couple teams, I've been like, literally writing it down, like our norm is leading with curiosity and solving the right problems. And so like, we reward it when it happens and when it doesn't happen, we call it out and like, why did we do those those tasks when they didn't actually solve a goal? And I love that the curiosity thing proves your point. That you don't need - need to have all the answers. You're just asking good questions to get the experts having the right conversation.

C Cory Tsang 25:07

Yeah, that's actually a really great segue to JIRA.

C Cheryl Platz 25:10

Let's do it.

C Cory Tsang 25:14

Okay.

E Enduring Play Computer 25:18

loading tutorial level: JIRA and agile basics.

C Cheryl Platz 25:24

JIRA is one of the most widely used project management tool suites on the market. It is often used as a shortcut for task or bug in the same way that Kleenex is a shortcut for tissue. JIRA started as primarily a bug and ticket workflow tool. Its primary uses remain to monitor the workload of individual team members via items in a backlog upon which work is logged, which generates velocity towards shared sprint goals. JIRA is not the same thing as the agile methodology, a project management approach which divides large projects up into smaller phases and places a high value on feedback driven iteration to determine the priority of what will be worked on next. Prior to agile, most projects were developed using a waterfall approach, a project sequentially moved from one stage to another as a monolithic entity. That's not always efficient, as it can leave people idle when you move between phases, like leaving artists behind. It also means bad strategy early in the process can become a monumentally costly error years later. By rearranging people into cross functional groups, you create pods of folks who can work on features or services in parallel, theoretically reducing your time to market or improving your ability to adapt in real time. Agile offers an information architecture for breaking big problems into smaller chunks. Tasks are general work items. Stories are requirements written from the perspective of an end user. Epics are large bodies of work that break down into stories and tasks and initiatives are collections of epics driving towards a common goal. You'll hear these terms used in a moment. Where most folks struggle is conflating agile with JIRA, or thinking that producers are magical unicorn fairies that will fill everyone's backlogs with fully populated tasks and bugs. Producers don't know your work better than you do. They are there to make sure that things don't get stuck, but they're usually not there to tell you how to do your job. That's on you.

E Enduring Play Computer 27:05

Loading podcast level five, how toilets inspired better JIRA process.

C Cory Tsang 27:11

So you asked about, like, the object oriented nature of the Sims and how it's carried forward. That's actually how I approach JIRA. So I'm a really big proponent for using epics as a functional description of a goal or deliverable, right? It's very common to see teams use epics as kind of like a bucket of their work. So you'll see all animation tasks in an epic, and all audio tasks in another epic, and all 3d tasks and another epic, and it creates a view that is very siloed, and you don't see the connections of your work. And it's very task focused. What we really want to see, what I like to do is create a description in the epic of what's the flow with the pipeline for the work to go through and be complete, where you actually have some sort of definition of Done. That's not to say that you don't need ways of seeing all of the animation tasks together, but there are other ways than grouping it as an epic

E Enduring Play Computer 28:14

Podcast level, paused. Loading, narrative cutscene, the Elder Scrolls Online.

C Cheryl Platz 28:21

As we shift into discussing corey's time as an art producer, it may help to understand the massive scale at which he was working on the Elder Scrolls Online. Notably, Corey joined the ESO team in 2012 a few years before its launch in 2014 and remained there until late 2025. As Corey implied in his introduction, the game experienced a major economic shift early in its lifetime. ESO launched with a subscription model, much like competitor World of Warcraft. However, less than a year after launch, the subscription was made optional. This is exactly why I introduced the Spectrum of Game Monetization in my book The Game Development Strategy Guide. It gives us a visual way to process how a game's model may shift over time. Eso moved from the upper right pay to play quadrant at launch to something more centered in the graph a year later, with Pay to Play, Pay to Express and Pay to Accelerate monetization via subscriptions and micro transactions. Corey worked as a senior producer on dungeons and ESO for 12 years before stepping into a lead art producer role coaching a team of four producers and three art directors to support a 100 person art team, as well as managing outsourced art assets. For context on our Nintendo DS game, Disney friends in 2008 each character had over 100 animations, and we had a team of 15 or so folks at a year on the original schedule, with all the other types of assets, of course, like voiceover, backgrounds, models, textures for about 64 megabytes on disc. The Elder Scrolls Online, over 100 gigabytes. Most of that space isn't code. That's art. Art production might sound like a soft skill, but it's incredibly intensive on a live service, open world, massively multiplayer online role playing game or MMO RPG, like the Elder Scrolls Online.

E Enduring Play Computer 29:53

Resume podcast, level five: How toilets inspired better JIRA process.

C Cory Tsang 30:00

And then, using like automation rules that help push things along, because I can show - I can, I use the blocking link type to let it flow. So as the blocking issues get closed, it checks to see what gets unblocked, make sure that all of its blockers are cleared. And then I set it to a ready status. And then I actually record a date. Because in the past, it's been really useful to understand blockages in the pipeline where things get bottlenecked. Because I had a team that was perceived as a bottleneck. And this is kind of how I became an art producer. I had a team that was perceived as a bottleneck at the end, and what I believed, and I felt was that there were some places along the pipeline that things were stalling, and I had to map out the entire pipeline and start following it along to identify, "Oh, somebody created a duplicate task, and they closed that one, and then the task that we were all looking at was just sitting around. So no one moved it forward, and no one knew that there was a handoff that should have happened." And so we got stuck, or we had a team that was just over overloaded and they weren't able to keep up with the pace. So once I got the JIRA system in, I could actually track with the ready state, the velocity which we were receiving work, not the velocity of how much we were delivering. Well, I could do both, but being able to show that velocity incoming was important to start getting eyes on where the blockage was upstream, and then being able to identify that and solve that problem.

C Cheryl Platz 31:27

That's a really good insight. I'm struggling with that too, and have in the past. It's not unique to any one specific employer, especially in the creative space - where it's not healthy process to just create a feature brief in a vacuum and fling it over to engineers and be like, "have fun."

C Cory Tsang 31:41

Right, yeah.

C Cheryl Platz 31:41

But then there's intermediate steps, these handoffs, back and forth, and it takes a lot of sophistication in JIRA to get those things accurately modeled. Like, where is it? Is it in review? Is it active? Who's doing it? Who's who's logging hours? Is this three tasks? Is it one task? It resonates to hear you talk about things like tracking intake, because sometimes what feels like, "Oh, we're blocking" is actually like, no, actually. Right now we are actually waiting for engineering to spec the work we've already done. So we have gone as far as is responsible to go in this moment.

E Enduring Play Computer 32:16

Podcast level, paused, begin host commentary.

C Cheryl Platz 32:21

A quick note here: IC stands for individual contributor, and it's usually used to distinguish somebody who's hands on in the work, as opposed to maybe somebody who's managing the work, like a producer or a manager.

E Enduring Play Computer 32:37

Loading podcast level six, self motivated meaning for tasks.

C Cheryl Platz 32:44

Let's dig in more on JIRA, because there's a lot here. And one thing I wanted to lean a little bit more on is your take on what an IC's daily best practices should be because I have found, and I don't know if you found this to be true, but to be true, but I you know, circumstances have given me the opportunity to go boom, boom, boom, across several different AAA studios very quickly. And what's interesting is it feels like there's less self accountability in the tools now than there were my first go round. And I almost heard that a little bit in what you were saying a little bit earlier about like, "hey, I want the producer to check my work," and so let's turn it around. Like, what do you think a healthy process looks like?

C Cory Tsang 33:32

In my ideal world, ICs would be making their own tasks because they my philosophy is that a task needs to mean something to the person doing the work. If it doesn't mean anything to them, they're not going to want to engage with it. So if, if they can look at a task and know, "oh, right, this is what we need to do," and it's not a "do this thing," right? Or do it, I've seen "DO IT" tasks before. It's the I made this task, or I had to put it down for a couple weeks, and they go back to it and say, "Oh, what was this?" And then kind of just wondering for me, like, I think ICS need to be able to just jot things down like they would write it down on a notepad. And part of what I've learned over the years is it has to be easy. It has to be easy to just create something that will show up on a list somewhere. And then that at the end of the day, that's pretty much - I just need them to move the task from status, like when they go in progress and when they close it when they're done. That's the other thing I find that ICS often aren't sure about closing something, so they leave it in, like a review status forever, and then it never moves. And so I try to promote a world where there are no mistakes. Like, it's easy to fix things in JIRA, and if it just closed by accident, we'll reopen it, not a big deal. And the reason why I say, like, just need to move it along. Like, if you transition your issues, I can set up an automation rule that tells me when you started it. And. And tells me when it got closed, and then we can do all the math through automation to figure out how much it cost. Now, having an estimate, great. That's great additional information, but if it doesn't serve anybody to have that estimate, then maybe we don't care, right? It all depends on how the team wants to operate, and if they need that predictability, then you probably need some sort of estimate. But I actually like using sprints for that. It's more it's easier to say, can you get this done by the end of the week, or by the end of sprint? And they say, Yes, I'm like, okay, maybe they'll spend five minutes here, five minutes there, or they split their time. I don't need granular tracking, and I don't think we should expect them to, but if they want to, I'm not going to say no more data is always good.

E Enduring Play Computer 35:47

Podcast level paused, begin. Host commentary.

C Cheryl Platz 35:51

So wait a second. What did Cory mean when he said that the object engineering in The Sims inspired his production work in JIRA on the Elder Scrolls Online. Do you see the pattern yet? Cory keeps talking about moving tasks and stories and epics through states. And we talked about finite state machines. Every task in JIRA has states open, in progress, ready for review, and those state machines are defined often by producers who are literate in JIRA. When teams need to build their own process, they need to find folks like Cory who understand how to build state machines so they can build more complex processes with intake and evaluation and passing the torch between different people and different teams. The same skills that helped Cory create problems between Sims by leaving this toilet seat up are the skills that helped Cory at Elder Scrolls Online smooth the gaps between people by automating processes for passing tasks between people when they were ready to hand off work.

E Enduring Play Computer 36:54

Podcast level seven, the sprint solution.

C Cheryl Platz 36:58

So while agile encourages quick iteration, it doesn't actually mandate sprints, which we typically associate with short, several week cycles of development. What role do sprints play in your work?

C Cory Tsang 37:13

Yeah, well, for me, sprints are about narrowing the viewpoint of the work. So in a world where you have a lot of work predefined. It looks overwhelming. I remember this when I was an engineer, and we used to get bugs, and there was always an anxiety about having too many bugs on your list, because we operated on zero out your bugs by the end of the day. And it was crazy. That was one of the reasons why, instance, one, I used to go home like at two in the morning, because QA left at like 11, and so the last batch of bugs kind of arrived and then and you had to zero out. So we were just there waiting for them to finish up and then get our last batch so they can test in the morning. So seeing a ton of things is really stressful, right?

E Enduring Play Computer 38:01

Loading, podcast level eight, slowing down to speed up, a case study.

Cory Tsang 38:09

Actually, this is a this is a story: my team? We had this huge backlog. I joined them. They had this huge backlog of work. And so what they would do is they would do the work, throw it to QA. Do the work, throw it to QA, and then QA would bug it up. Of course, they had a ton of bugs so, but they were trying to get burned through this backlog. But then we'd have to reserve like 50% of the release to just fixing the bugs, leading up to locking it right. And so we took the stance of, "okay, here's how long these things should take. We're going to break it into sprints." We did like two week sprints. And said, "Okay, you only have to get through this amount of work." And so they started taking their time to do it like thoroughly, and maybe test their work a little bit more. And we saw a reduction overall in the number of bugs that we were creating, and that actually reduced it to about 30% of the schedule instead of 50% of the schedule. That actually gave them enough time that we started building more resilient tools and templates to prevent bugs from happening. And in the end, we ended up reducing the amount of time reserved for bugs to about 20% of the schedule, and we were actually able to do more. And so we burned through the backlog, and then all of a sudden, we actually had more capacity than we were the incoming velocity, which was really good, because we had someone leave, and we actually had some tool changes, and the velocity dropped because a lot of that stuff, but because we were so far ahead, like we were able to absorb it, mitigate it, and then carry on.

Cheryl Platz 39:36

But that's an amazing story, and very hopeful, because I think, you know, I've seen it on many teams, where you sort of get into this self fulfilling prophecy, where you feel like you're under this tech debt load that you will never be able to clear. Because that 50% of time for bugs in that log of bugs, like, "how are we ever..." Because the more you develop that, the bigger the list gets. But what you described the slow down to speed up mentality sounds bonkers when you try to bring it to a team. And I've been doing it from a creative perspective. You're talking about doing it from a production engineering perspective, and you bring it to a team, and they're like, "You don't understand, my life is pain, and you want to, you want to take more time?" Like, "Please just take a deep breath." And I have seen the same thing you described happen on teams I've worked on where they do they bring QA up earlier in the process. They do integrated QA earlier. And they're like, "Weird, there's less bugs." Like it's true, like, the longer you go, the more the things can just compound. There's just something about being watchful, being intentional, stopping and checking your work. It's just better. And once you stop spending so much time having to chase everything, and bugs compound bugs. And I think that's part of it. The bugs compound the bugs. Once there's so many of them, you can't - you spend time trying to figure out which which broken is this, right?

Cory Tsang 40:52

Right, yeah

C Cheryl Platz 40:52

I don't know if you've ever seen the idea, but there's an episode where, like, everything's on fire, and he's like, "I'll just go put this with the rest of the fire." I've had at least one experience where I encountered a team where I was like, hey, Reddit is talking about this thing. They're like, Yeah, we just put that in this pile of bugs. Like, "it's not at all that kind of a bug," but they were just so used to - they were so overwhelmed. You can't tell the bugs apart after a while, because it's piled up and piled up and piled up. So I love this story of, like, slow down, pull QA in, get intentional so that you can tell the bugs apart. Takes less time you free up time on your calendar suddenly, yeah, you can start building the game again.

E Enduring Play Computer 41:31

Loading podcast level nine, mastering your production tools.

C Cory Tsang 41:37

Yeah, actually, that's, um, that's kind of like how I've approached production so, like, I am of the production age, where we used to use physical boards, and I'd print out templated cards, and they would fall off the wall, and you spend time putting it back up. And, you know, I spent a large amount of time in spreadsheets. What actually got me into JIRA was I needed to transition the team to somebody who just wasn't quite as spreadsheet savvy, and so I started looking into how we could do this just in JIRA, and I had been looking into it for another team, and I was like, "Oh, I can do this with reporting. It's just- I need to fix how we use JIRA so that it works." But what I try to do is I try to reserve time for myself to learn how to use tools and how to do things a little bit easier, because if I just spend a little that time to learn things become easier, and I make more time right, which then I can use to get better or explore different things, and then it compounds. And so, yeah, I can't recommend enough to just eke out some time to try to figure out the new tools, because it makes life so much easier.

C Cheryl Platz 42:45

The thing with that is often to have a problem you want to solve when going in trying to learn the tool. Because as a user experience designer, human-centered designer, I've often found that, like, when you give someone a tool and you're like, "learn teams!" people are like, "Why? Why would I do that?" But if you're like, "Hello, are you trying to communicate with someone in a low bandwidth scenario?" And only other solution is Skype, then, then suddenly people are like, "Oh, I'll go learn Teams." And when I'm mentoring directs, and they're like, "I just want to want to learn a tool." I'm like, "That's cool, but how do you measure success? But let's look at some of the problems we have on the team, and does this tool you want to learn, maybe potentially solve one of those things?" Because you described this, you had such a clear goal, you're like, well, have this person. They can't use my spreadsheet, so I know what problem I need to solve - that's the thing that that spreadsheet solved, I'm just going to try to do that with JIRA, and that's like a really lovely, clear rabbit to chase,

E Enduring Play Computer 43:37

loading podcast level 10, insight from production tools.

C Cheryl Platz 43:42

So we have all these JIRA tasks. We have all of these bugs. How are you as a very talented producer, helping your team turn this pile of data into insight? I see so many teams struggle with like JIRA dashboards and reporting?

C Cory Tsang 43:56

Yeah, because we definitely had reporting that we needed to provide. And, yeah, having a clear goal again, goal focused work is amazing, in fact. So when I started being a lead and had directs, the way I kind of approached it was, or at least in the beginning, was one come up with a question. What kind of question do you have about your team? And then we're going to spend some time figuring out how to solve that question with a query. And then the next time we come you, go "okay, here's the query. Okay, what questions does this raise now? What's your next level question?"

C Cheryl Platz 44:27

Oooh.

C Cory Tsang 44:28

And then we work on trying to solve that level question. And so we would just keep doing that, and just keep pushing. If I know how much work is coming in, what do you need to ask about, about that work? Right? You know, "when's the delivery? do I know how this fits into everything else? Can I trace how my work connects to an epic, to an initiative?" and then that starts leading into things like, oh, "I need an add on for enhanced searching," because you can't in vanilla jury, you can't find issues based off of the properties of its parent or a linked item. So vanilla JIRA is hard to use. It's really rigid in things, and so by using add ons, you can really make it work for you.

E Enduring Play Computer 45:15

Loading podcast level 11 Sims inspired planning hierarchies.

C Cheryl Platz 45:22

But how do you know how to organize all of the epics and bugs and tasks? that's hard!

C Cory Tsang 45:28

The way I plan work is how I used to make objects in The Sims.

C Cheryl Platz 45:35

Tell me more. Tell me more.

C Cory Tsang 45:36

So, okay, so, so the biggest problem with JIRA is that its hierarchy is really rigid. You have to have epics connected to initiatives. You have to have tasks or stories connected to epics, and then sub tasks up to that, and you're limited to the depth of hierarchy once you start using something I learned, I think was a GDC talk, what they call the soft hierarchy. It's just a parent child relationship link. It's how we used to fake more levels above epic, before they created it in JIRA cloud. It allows you to arbitrarily create a parent relationship using links, and you can have multiple parents, which is really great. Structure is the add on that I use, which allows you to express that hierarchy in a visual way. So the way it goes is, I talk about it in terms of levels of uncertainty. The first level uncertainty is, "I'm going to do a project, when does it start and when does it end?" And so we can create an issue, usually, like, if you think in terms of just containers and work, and epic being a container, you just create the epic. And it says, "We're going to start here and there." The next level of uncertainty is like, "who, what teams do you need? like, what disciplines do you need, or what phases of development are there to achieve this?" Right? However you want to break that down, and then you basically go down and you parent another level of issues to that so that you can say, well, this thing has ideation and pre production and production, and you know, QA passes, and then lock, and then, now we're out. Okay, well, "what goes into ideation? Who's involved? What are those tasks?" And you can kind of drill down and solve for each of these different pieces. And then, because you're no longer limited in the depth of work, like your depth of the hierarchy, because you're using a link type, you no longer have to worry about, oh, is that the right size task? Because if I do that task and this other task needs to be at this level, you forget all of that, right? And then if you keep this idea of that, there's only container issues and work issues, and that container issues always at an epic level, then you can have tasks hanging at any one of these levels of uncertainty.

C Cheryl Platz 47:46

So you might be looking at the top level, like I need to look into whether or not this is the right problem to solve. And you might have a task up there,

C Cory Tsang 47:53

Yeah.

C Cheryl Platz 47:53

Or you might be way down the line, have a task looking into a policy or something, or fixing a brain.

C Cory Tsang 48:00

So, so a good example is you might have an epic for making a monster, right, but that epic needs to have a rig, right? Because the monster needs a rig, and that - "make a rig" is actually an epic of its own, because you need to do, you know, some testing. You have to do the actual technical artwork. You have to get your proxies in, whatever. And then you have the monster. But the monster is actually maybe - every ability could be an epic, because we might get too granular there, but you have to have your animation and effects and audio and testing, because that thing has to have some tuning to it and as a complete unit. So then you have this monster. But then now you say, "Oh, well, that monster is part of this dungeon, and it's going to be a World Boss over there, and it's going to be part of this trial." And now you have to be able to connect this epic to multiple places. So using a parent child linking type, you could link it to all of those things and create visuals so that you know this thing needs to be part of a larger thing. And no matter what level your team's thinking of the organization, can place it anywhere in the hierarchy. That might be way too abstract for it's but it, it's how I think.

E Enduring Play Computer 49:15

podcast level 12, the why of production.

C Cheryl Platz 49:19

You're giving people flexibility to take this, this piece of information, or this set of information, and use it wherever it's needed, yeah, link it wherever it's needed. Because, and I think fact that it's maybe a little abstract or maybe a little difficult to understand, speaks to the true difficulty of making games, which is everything we do is so deeply interconnected, it's like, yeah, it is hard.

C Cory Tsang 49:40

Right.

C Cheryl Platz 49:40

That's - That's why we need producers. Is why we need these tools. Is because, like, make a monster, and it's like, yeah, barely an inconvenience. You just need 40 people and complete narrative alignment. And it needs to work on six different types of consoles and not melt any of them, no problem. So, yeah, it is a little complicated and and it does to link. And I. It makes sense because, like, it makes sense to me that you'd have multiple epics also in the scope of make a monster, which is from a player perspective, from a player impact perspective, what you need to do?

C Cory Tsang 50:13

Yeah, from an IC standpoint, the word epic doesn't actually need to be epic, right? It's just the container in the organization. Now, ICS, I feel just need to worry about the work. And if they're curious, they can look at how it's organized. So we always need to have that represented. You can always follow the breadcrumbs in JIRA to find out how you're connected to everything. And then the containers are where I think leads and producers kind of live. Because we're trying to say, let's say an IC makes a bunch of tasks for themselves, we need to say, "Okay, what is that for? And how does that connect into the work?" We can then take that and then organize it into the structure in a way that makes sense to us, so we know how things are connected, and we can show it to people if they want to, but a lot of people don't want to, right? They just want to see their list of tasks. And so that level of complexity doesn't have to hit the ICS, but I think it's important for leadership, anybody in any level of leadership, to kind of see how it all connects into the big picture. And that's what I feel JIRA helps with.

E Enduring Play Computer 51:12

Podcast level 13: Chain of impact

C Cheryl Platz 51:13

And so to go back to what you were saying earlier, it's not so much on the producer to, like, check an engineer's work to see whether or not it's the right solve. But it is on the producer to say, like, "great. You've chosen to do this work. I'm going to help everybody else understand the impact of this work and make sure they're aware this work is happening, and make sure all the downstream and upstream impacts are communicated, connected, documented, that QA is ready for it, that marketing is ready for it." So you're not checking like, is this the right work to be doing, but you're making sure that the whole ecosystem is ready to receive it.

C Cory Tsang 51:48

Right. Yeah. And, and if things are moving off track, it's easy to see the chain of impact. So you can do an impact analysis to know Who do you need to talk to about that, right? I don't say you missed a deadline bad and say it didn't hit now that we can see that, or we can see it early that we're training, who do we need to talk to to find out if it's bad? Right? Because sometimes, not all things require audio at the end or an effect, right? Sometimes there are variations of things that are just simpler, and you're looking at a kind of like a broad picture of what was going on. And sometimes you can adapt, right? Sometimes the schedule can be flexible. Sometimes people a lot of times, especially when you're using sprints, you can build in the buffer, like, "I'm going to reserve 20% of my sprint for something changed unexpectedly, or we got feedback that is going to take a lot more time." And so people build that flexibility into their time that, "yeah, it could be absorbed," and it's fine. So it's not about perfection, it's not about accuracy. It's just about what do we think is right? How do we trend toward it, and how do we detect if there is potentially a problem? And then how do we evaluate it, but in a way, by doing it, in a way that it's quick and easy to see, and it doesn't take us a week pouring through three or four different spreadsheets to try to match things up and then talking to people to say, "Are you sure this is okay? Are you okay with this? Can you do some extra work?" And it all becomes word of mouth, which is fine. I don't want to. I definitely don't want to sound like I'm telling people not to talk. What I what I tell people often, is that, as a producer, JIRA, helps me see things so I know what questions to ask, not trusting the data over the people. It's about making sure I can see something and if I miss something for some reason, what can we change about the process so that we can detect it again next time?

E Enduring Play Computer 53:38

Loading podcast level 14, Working out loud

C Cheryl Platz 53:44

It's interesting, you mentioned like you're not telling people not to talk, right? Because I've seen teams stumble over that sometimes, where I saw a team once, where accidentally concern about like overhead got misinterpreted is, oh, we shouldn't be talking. And the way I built that into my team norms. It was like, "work out loud", hey, actually, we do want to collaborate, but we want to collaborate in a way that is openly visible to all as much as possible. Like, yes, talk to each other, meet with each other, but be writing it down. Be doing it in visible places. Be doing it a way that we can learn from it later. Be doing it in a persistent way. So like, Sure, okay, have your hallway conversations if you need to, but like, we're going to need that data. So it also needs to go into JIRA, or it needs to go to Slack, or wherever the right place, or confluence, wherever the right place for it is like the conversation the hallway did not remove our need to learn from it, to track it, to do the things. And one recurring theme that I don't know if you get from your executives, but I should get from my executives, is they don't like surprise, right? And so your friendly neighborhood producer is out here trying to be like, Hi, I'm a math wizard. And so sometimes the math can help us avoid the surprises that the executives don't like. So we're here to try to help present those awkward conversations before they happen by saying the math isn't math thing anymore. Maybe we should bring that up before it becomes a nasty surprise. Yeah, that's no one's here trying to make people not talk or like we're just like, this is a really hard thing. And sometimes you can see the train jumping the rails before it actually hurts anyone.

C Cory Tsang 55:11

Yeah, it's, it's kind of like a scaling problem. So when I say is the technology that the data is a backstop for when communication fails. So if you can automate the handoff process, it doesn't mean don't talk to people. It doesn't mean wait until this thing becomes ready to go and do the work. If you have bandwidth to go ask, go ask and see if you can start it. If you want to start it early, start it early. But if someone's out, or they there was some someone else did the task, and they are less familiar, or, you know, we have churn, and you have new employees who are less ingrained in the culture of the team, and they're they're still learning, and they don't know they need to go talk to a person, then it serves as a backstop of the communication so that it doesn't stall. That's the thing, is that, I mean, granted, this is more scaled up. Aaa, like pushing out tons of content, having a stall is horrible because it affects so many teams. But yeah, automation data, it's actually it's just, it is a two edged sword, because if it's really reliable, people get a little bit they lean on it a lot, and then the communication lines start to break down a little bit, and so you have to actively push for it, but on the flip side, like it does provide a little bit extra safety for smaller teams, it's less important because it's more intimate in a small team to talk to people and constantly communicate. When you scale up to like three, 400 people, it's less possible and

E Enduring Play Computer 56:39

podcast level 15 putting the quest for the right work on easy mode.

C Cheryl Platz 56:45

There's some key inflection points when, when studios move up the sort of 50, the 100 the 150 the 300 and you know, one of the coping mechanisms I've built into my creative ideation processes is, like in my feature briefs, literally sort of chunking them out and saying, like, cool. This is your draft. This is the part where you go talk to the other humans. And these are the other humans you're going to go talk to, and you do not get to pass go until you've talked to the other humans. And this is the part they're going to fill out. And it feels so literal, and it feels like Mad Libs, but I just found, like so many stalls, it's, I don't know if it's generational. I don't know if it's just because games are harder than they used to be, but like, people like, yeah, how do I don't know what questions to ask. I'm like, great. I'm just gonna put just gonna help you with that. And when people had that answer key, we had the stalls and we had thriving, right? Like people, people were, they were leaving with more curiosity. Like, oh, here's this problem. Oh, here's how we solve it. But for some reason, it was like people were, like, afraid to collaborate, or they didn't want to interrupt, or they were there's this perception, I think, and I maybe I see it more now because I'm in like, senior leadership. People like, oh, well, you're so busy I didn't want to bother you. I'm like, Well, I'm always, everyone's always busy all the time. I want to be the right level of busy. My calendar is going to fill up with stuff that doesn't matter if you don't come to me with the right like building those interrupts, building the automations, building the reminders, building the things. They're hopefully getting us to the right Busy instead of the wrong busy.

C Cory Tsang 58:09

Yeah. That actually goes to, I know we're maybe low on time, but one of the big principles of what I do now is I want to make it as easy as possible for people to create work. People should always ask for what they need, regardless of whether they think another person has time for it, because usually things that show up late are actually high priority. And so what we need is an easy intake system so that you can then and then, an easy way of looking at that in context of all the other work, to be able to then prioritize it in the big picture. So there's been plenty of instances in the past where, you know, a team is underwater, and they're like, "Oh, don't ask them, because they're too busy," right? But if you ask them, and this thing is really high priority, we can look at our our thing and say it's here and and insert it, right? But if you never ask for it, then it won't get done, and we may have missed an opportunity. And so you always have to allow the asks, yeah, encourage the asks, because any that's the other thing I think I mentioned to you, any friction to the system means people just avoid, avoid it, make it easy to not only ask, but then frame it up so that you can make those decisions. Whoever needs to make the decisions can see it and and feel empowered to make choice.

C Cheryl Platz 59:36

Yeah, change management, it was, yeah, that's, that's a journey. Yeah, we don't have time to unpack all of that, but, like, that's the next piece on the truth. There's like, the the creation, the production process. There's the make game, make game first, and then there's the like, fix game and transform game and change management and all of that. It was just a whole other art, which we probably have a whole other hour long discussion.

E Enduring Play Computer 59:59

Final Boss level, partnering with it to level up your tools,

C Cory Tsang 1:00:06

I'm going to throw something out there that may sound like a pitch. It's not really a pitch. It's not a hit on IT. I actually did, like I spent my first seven and a half years before we got in games, as IT. It is really important that somebody on the game team knows how to use, like how to configure Jira, because going back to goals and problems, being in touch with the actual pain points and friction that the team's experience is going to inform how you design your site to work for the team. And it's really difficult for someone in, like, a central services role to know and predict and kind of work ahead for that they can execute requests. But really it's about designing the system together, and so being in it like I had an opportunity to just be a juror administrator for the studio, but that meant leaving production, and I'm like, I couldn't do it, because we're continually evolving how we use the tools to solve problems. Because problems, once you solve one problem, usually you find another, right? Once you answer one question, there's more questions to answer. And so yeah, it's just, it's very important.

C Cheryl Platz 1:01:21

Yeah, that makes a ton of sense to me. And it's even harder than it sounds, because a lot of times, if you're in a company that has multiple teams or multiple units you, like, you get into, like, instance or environment management, and like, the change you make affects other teams. And we don't have time to unpack all Yeah, it's so it's that's why, at least, at the very least, a good relationship with it, and, like, open conversation about what your needs are is important. But even better, like having somebody who can do these things and is empowered to do these things, and can change the shape of your JIRA, because it's, it's user experience, yeah, and it's not just the user experience of JIRA, it's the user experience of, like, the daily operation of your team. Because whether it's JIRA or some other task management tool, that is how your team is moving your team's accomplishments through time and space. So yes, plus 1000 loading,

E Enduring Play Computer 1:02:14

bonus level: finding meaning in the journey.

C Cheryl Platz 1:02:19

But I like to sometimes wrap things up with like, what are you excited about now? Like, what as you look towards your future, and I know we're not going to talk about your specific work, but like, as you look into the industry, I know it's on fire, but like, when you look at trends, or you look at tools, is there anything that you're like, This is okay, or this is good, or this genre is good, or this this tool is good, or this automation is good? Is there? Is there anything that leaves you feeling like I'm glad we're here and not where we were.

C Cory Tsang 1:02:43

I mean, there's, there's always the I'm glad I have a job.

C Cheryl Platz 1:02:46

That's fair. I mean, it's a real thing you've been through that, I've been through that.

C Cory Tsang 1:02:51

Well, yeah, I got so last July was, it was first time I got laid off in 29 years. But I was really lucky. I I got picked up really quickly. But I think what keeps me kind of excited is that despite the general negativity around Jira, definitely find that it's maturing, and I can help people. That's the thing is, it's all about balance. I can try to provide the tools that help people have a better experience working. Yeah, that's, that's the thing that drives me most, is work life balance.

C Cheryl Platz 1:03:30

I love that, Especially with our shared experience in EA back in the early aughts like that is a very powerful driver.

C Cory Tsang 1:03:36

Yeah.

C Cheryl Platz 1:03:36

It's- your teams are lucky to have you, because you've been forged in the fire of from designing toilets at 2am in the morning with those crazy bugs through the Elder Scrolls and now this, and I just can't wait to see what great as your career continues to unfold, and I'm so grateful for your time today. This was a really fun conversation. I know you were like, I don't know if you're like, No, no. Like, we need this is, the thing is, the industry needs more of these conversations about the thing. Like, it's, we have so many shiny things to talk about. Like, oh, let's, let's talk about that, the game design. But like, also, we need to get nerdy about the JIRA and the people management and stuff, because that's one of the things I've been thinking about. Is it's easy to just think about the game design, but they take so long to make these games. And if we don't talk about being sustainable, that's why we keep falling down. So like this hard work that you're doing as a producer, it needs to be celebrated, it needs to be studied and lifted up. And so thank you genuinely

E Enduring Play Computer 1:04:40

Loading final podcast level links and follow up.

C Cheryl Platz 1:04:44

Well, thank you so much. If people want to follow your work, what's the best way for them to like follow you on the vast internets?

C Cory Tsang 1:04:52

Probably LinkedIn, but I'm notoriously quiet on the internets.

C Cheryl Platz 1:04:58

Well, that's okay. That's okay. The. Your work is still out there in the world too, in the show notes, I'll link to the games you've worked on and so they can go appreciate that. Thank you so much. I really, genuinely appreciate the chance to nerd out about these topics with you. And

C Cory Tsang 1:05:12

Yeah, my pleasure. Thanks for having me.

C Cheryl Platz 1:05:14

Yeah. And I hope our paths will cross again sooner rather than later. And congratulations. It was on the new role in this this market. It's no small feat to do that, and can't wait to see how things transform for you.

C Cory Tsang 1:05:28

Thank you so much.

C Cheryl Platz 1:05:30

Continue exploring the craft of game development with my book that inspired this podcast, The Game Development Strategy Guide available worldwide from your favorite online bookseller or from my publisher, rosenfeldmedia.com. Enduring play listeners can get 15% off@rosenfeldmedia.com through July, 31 2026, using the code enduringplayS2 check out season one of this podcast for the interviews that helped inspire the book or tune in next time for interviews that help expand our understanding of what it takes to create games that don't just survive but thrive. I'm your host. Cheryl Platz, until next time, keep thriving.

E Enduring Play Computer 1:06:16

Initiating final credit sequence. Podcast, enduring play. Created and produced by Cheryl Platz. Copyright IdeaPlatz LLC, All Rights Reserved. Explore the podcast enduringplay.com. Podcast theme composed by Cheryl Platz. To learn more about Cheryl's books, visit Rosenfeldmedia.com. Follow ideaPlatz on Instagram, Youtube or blue sky, I, D, E, A, P, L, A, T, Z. To learn more About ideaPlatz, the design education consultancy, visit ideaplatz.com. Thank you to David Demma for episode quality assurance. For book and podcast merch, go to idea platz.dashery.com.