£3
ALL FORMATS

Wireframe

LIFTING THE LID ON VIDEO GAMES

PICO-8 SPECIAL

MAKE YOUR FIRST GAME IN A TINY VIRTUAL CONSOLE

PIXEL PERFECT

MODERN GAMES MADE WITH RETRO CONSTRAINTS

Issue 12 £3
wfmag.cc

9 772631 672000 12

REBEL REBEL

INSIDE THE GENRE-REDEFINING RHYTHM GAME,
NO STRAIGHT ROADS

Subscribe today

13 issues for £20

Visit **wfmag.cc/subscribe** or call **01293 312192** to order

Subscription queries: **wireframe@subscriptionhelpline.co.uk**

There's no such thing as an apolitical game

The Division 2 takes place in Washington D.C. in the aftermath of a smallpox pandemic, and has you liberating the city from a corrupt government. The game's tone-deaf marketing used real-world political tensions, such as sending out a joke email referencing the US government shutdown and a spoof letter declaring Mexico 'approved funding for building a wall along the United States border'.

Despite all this, creative director Terry Spier recently told Polygon, "We're definitely not making any political statements... This is still a work of fiction."

It's a denial that attempts to curb any critique of the game's themes. It's also a ham-fisted attempt to curtail the internet's toxic hate mobs that loudly demand we 'keep politics out of games'. By 'politics', said mobs mostly seem to mean anything they see as pushing a progressive agenda: the inclusion of female soldiers in *Battlefield V*, say.

When developers and publishers say their games aren't political comments, they ignore the simple fact that these themes can't be separated from their real-world contexts. Claiming to be apolitical is itself a political statement: an act of appeasement to a toxic community; a disownment of responsibility. The people who decry the presence of 'politics' in games wilfully ignore that games have always been political. Video games are part of a much longer history of play. While the examples often cited – the board games made by the suffragettes, for example – are rooted in clearly defined political messaging, nearly every game is a product of its society and the politics of its time.

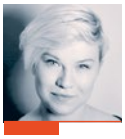
In late 18th- and 19th-century Britain, there was a surge in the publication of educational board games aimed at teaching children geography. Unsurprisingly, a recurring theme was empire. The way in which empire was both depicted and discussed in these games tells us about their political contexts. Some of the games' portrayals seem to critique militarism or mercantilism, but celebrate the actions of missionaries, such as The Noble Game of Elephant and Castle (1822), while others unabashedly celebrate empire in all its forms, like 'Tar of All Weathers, or the Game of British Colonies' (c.1857).

They're *statements*, whether the designer perceived them as such or not. If I came across a source stating that one of the publishers of these games denied its game was a comment on empire, then my reaction wouldn't be 'Well, I guess I can't analyse it because it's not a statement after all'. If anything, it would pique my interest, because the idea that structured games are purely escapist fun is a fairly modern concept. For over two centuries, board games were largely seen as a medium meant to teach and 'improve' their players. While I've found examples in historical documents showing opposition to the political sentiments within a game, I've yet to find any responses that state that games themselves *shouldn't* be a political medium.

This denial of responsibility for used themes is pervasive in modern games. As a journalist, I've sat in two separate preview events, both for very different games, but both with themes of imperialist or colonialist expansion at their core.

Yet at these events, we were told these games were in no way a comment on imperialism or colonialism. If you have to say that your game isn't making a statement, then that ship has sailed. You can't separate subject matter from its context; just like those geography board games, a 'comment' has already been made by its mere existence. You don't get to separate yourself from these themes because you say so, or hide behind the idea that the agency that games give players shields you from criticism.

All games are political in some sense because, like every piece of art or media, they're created within a social and cultural context. *The Division 2* will most likely be used by historians and researchers one day as a product of its time, just as the geographical games of the 19th century are studied now. Instead of trying to state the impossible by attempting to remove themselves from this reality, developers should examine and take responsibility for the narratives and themes they use in their games.



HOLLY NIELSEN

Holly Nielsen is a videogame journalist and historian who researches the history of British board games. She regularly gives talks and lectures on the history of games, and how history is used in games.

#12

Contents

Attract mode

Interface

Toolbox

Rated

WELCOME

There's something oddly comforting about PICO-8, the machine we introduce in our feature on page 18. Created by Joseph 'Zep' White at Lexaloffle, it's purposefully designed to evoke the feeling of programming an 8-bit machine like a BBC Micro or ZX Spectrum. But there's more to PICO-8 than just nostalgia; it's an environment that, with its chunky 128×128 display and tiny 32kB memory, creates a set of confines that positively encourage users to think economically. As others have pointed out, there's nothing like a set of confines to spark creativity, and you'll see all kinds of inventive ideas at work in this issue. It's probably no accident that the acclaimed *Celeste* originally started out as a PICO-8 game, for example – it's a solid place for testing out ideas before expanding them and adding scope. Other developers, like Paul 'Liquidream' Nicholas, have managed to do impressive technical things with PICO-8 – like create a 2D rendition of *No Man's Sky* – within its piffling bit of memory. Best of all, PICO-8 offers a friendly gateway to designing games; it's like programming a ZX Spectrum, but without the faint sense of terror that came with saving all your hard work to an audio cassette. Enjoy the new issue.

Ryan Lambie
Editor

Going UNDERGROUND

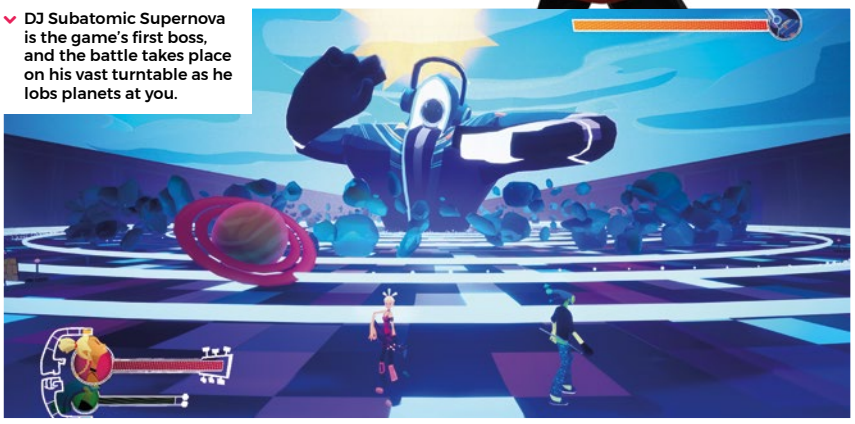
We meet the creators of No Straight Roads, the rhythm-action game with a rebellious streak

One minute I'm fighting a gigantic DJ surrounded by planets and asteroids; the next I'm fighting a colossal, shrieking matriarch who's angry that I've interrupted her daughter's piano concert. This is the world of *No Straight Roads*: part brawler, part rhythm-action game, where enemy attacks and player counter-strikes are synced to the ever-changing soundtrack. You switch control between two members of an underground rock band (literally underground: they live in a sewer), whose aim is to use their music to take down bosses and, eventually, save their city from a despotic, rock-and-roll-hating empire. It's all delightfully eccentric and vivacious, and so too are the people behind it.

Malaysian developer Metronomik was founded in 2017 by Wan Hazmer, a former lead designer on *Final Fantasy XV*, and Daim Dziauddin, who was once an illustrator and concept artist for *Street Fighter IV* and *V*; when Wireframe meets the pair at EGX Rezzed in early April, they talk excitedly about their previous careers in Tokyo, and how it was their dream to form a studio and make a game of their own. They effervesce about gathering their team of programmers, musicians, and artists – some of them are new to the industry, like mural artist turned illustrator Ellie Yong, while others, like composer Falk Au Yeong, are also veterans of the *Final Fantasy* franchise. They enthuse about their plans for *No Straight Roads*: an ambitious riot of colour and music inspired by such things as *Jet Set Radio*, *Brütal Legend,* and *Steven Universe*.

At one stage, Dziauddin interjects, "Oh, by the way, we're cousins" – and then the pair burst out laughing. We have to admit, their excitement over their one-of-a-kind action game is infectious. Here's what Metronomik had to say.

"Everything gels through narrative, gameplay, and design – it's a very nice flow, the growth of the game"



DJ Subatomic Supernova is the game's first boss, and the battle takes place on his vast turntable as he lobs planets at you.

No Straight Roads creative director (and Metronomik CCO) Daim Dziauddin.

As you've taken it around the world, have the reactions to *No Straight Roads* been different in each country?

Wan Hazmer: At the Tokyo Game Show, we had users who really loved indie games. At Paris Games Week, we had more children playing the game, which was really interesting – we were next to *Fortnite*, so all the kids were there, and we attracted them over. At the Taipei Game Show... I guess the game has an Asian flavour in a way, because of the art, and also the voice as well, so they loved those aspects of the game. PAX East was mostly about how awesome the music is, and how humorous it is. So it's interesting how they all perceive the game differently. But it's mostly been all positive.

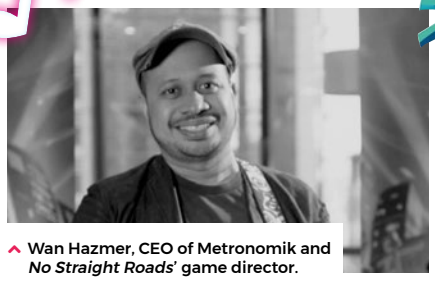
What's the process of creating these boss battles like? Because they have many stages to them. Do you storyboard them? Maybe act them out to each other?

WH: Where do I start? At first, we were talking about how it would be cool to have a game where you use rock music to fight [the villains]. So the idea is that all the bosses follow the music, and then from there, we wanted to have a reason why they're all playing music – so the boss battles come out of their personalities.

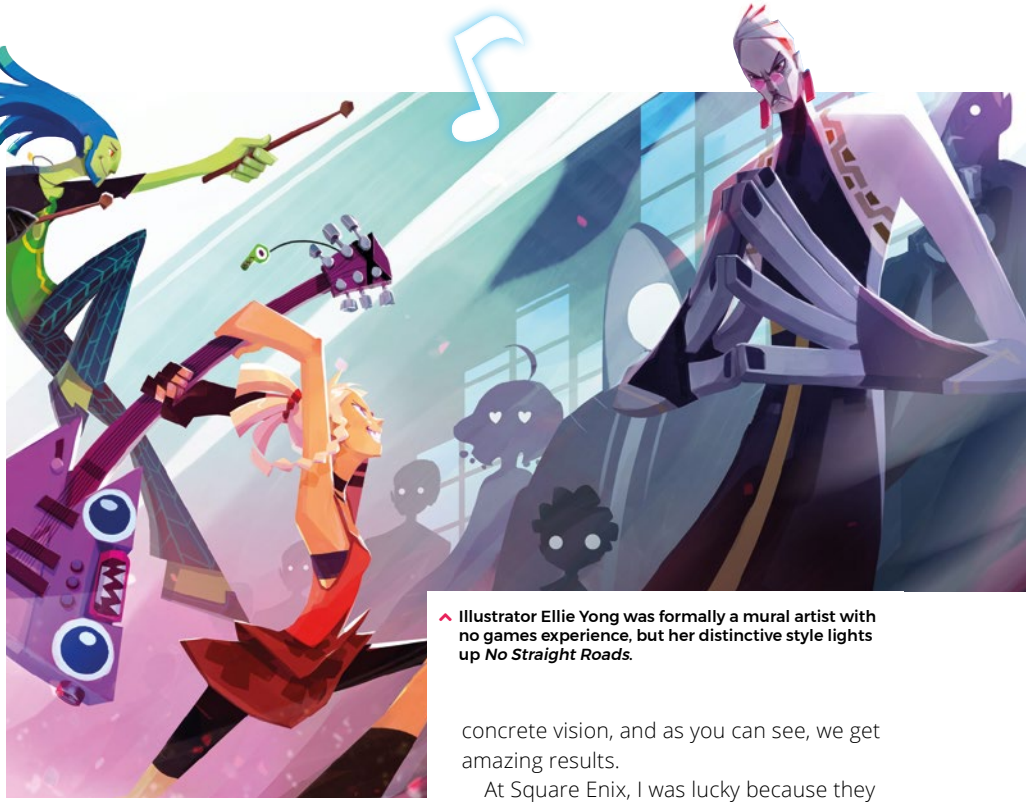
Daim Dziauddin: Story-wise, as well, we love movies so much, but we understand that in gaming you can't tell a movie story through a game. So we had time to really tell the story through psychological experience. We had a vague idea for a boss kind of a style for the music – you'll notice that each boss has their own [visual] theme, so the DJ has a solar system theme, for the mother and child it's more like nature. When we thought about the solar system, we thought, "Oh, maybe the battle can be a circular thing." So everything gels through narrative, gameplay, and design – it's a very nice flow, the growth of the game.

WH: All the different personalities, how they perceive the music, are all fragments of [Daim's] personality!

DD: Ha, yeah! Some of the bosses are a little bit like me. I love music, but I can't follow music theory – like sheet music



Wan Hazmer, CEO of Metronomik and *No Straight Roads*' game director.



Illustrator Ellie Yong was formally a mural artist with no games experience, but her distinctive style lights up *No Straight Roads*.

and stuff. That frustrates me, you know? I listen to music and copy it – I play bass and drums...

WH: We used to be in a band together. Not a major one.

DD: Not even performing live. We were just by ourselves. But Zuke [one of the central characters] is more of the musical theory kind of character. This is the multiple personality approach to music we put in the game.

WH: I think that makes it more relevant to people. Because there's some form of creativity in everyone.

So how quickly did you come up with the visual style? Because obviously, it has such an impact on the game. Even a change to the colour palette can change the tone of the game completely.

WH: You're right – it changes the whole direction of the game. I guess the most important thing is, it has to look fun. I thought that we had to go colourful – let's not take ourselves too seriously. Let's give everybody green, blue, red, yellow skin. We just went from there.

DD: It was funny how we found our artist [Ellie Yong]. We went to Deviantart, and he [Wan] was really raving about this lady. But I saw her portfolio and said, "We really need to have her." I contacted her by email, and coincidentally, she lives in Kuala Lumpur as well – and she was jobless at the time! [Laughs]

WH: It was nice timing, us getting together. I just saw her work on my computer and I was like, "We need to get this girl, now!"

DD: She doesn't know much about game art...

WH: ...but at the same time, that's amazing. It was so lucky for us. She might have easily said, "Oh, games are so stupid." But she was like, "Oh, I want to do this."

We're also working with [composers] Falk Au Yeong, Pejman Roozbeh, Andy Tunstall, and James Landino – the four of them are composing the music for the game. And one of the things I like to boast about the company is that half of them are fresh graduates – there's a lot of talent in Malaysia, and we want to support them.

The studio you've set up sounds like a fun, loose place to work. Is that quite different from your experiences of working in Japan?

WH: [Spontaneous peals of laughter] Super, super different.

DD: I could say a lot of things about the different things between our studio and our past history...

WH: Careful! Actually, I loved working at Square Enix. I worked with some very talented people. The good thing about Square Enix is, if you let go of your very concrete vision, people will come back to you with wonders. I've brought that to our studio as well. I try my best not to give a concrete vision, and as you can see, we get amazing results.

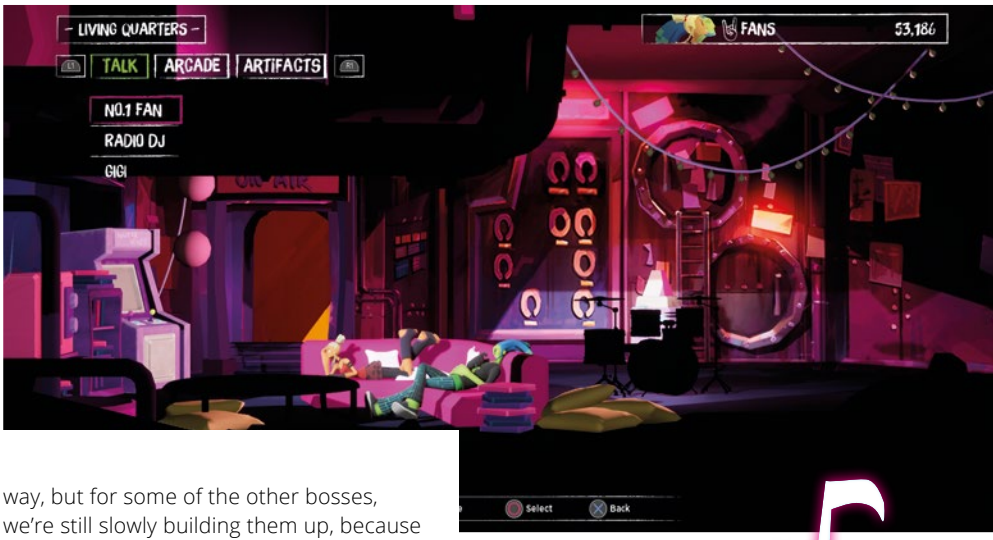
"I don't mind too much if you don't have experience; our 3D modeller was an accountant"

At Square Enix, I was lucky because they care about Western ways of doing things as well. Especially my director, [Hajime] Tabata – he's an amazing director. Also, he loves crazy people. The guy who worked on the cars in the game – he used to be a truck driver, for seven years. Then after, he learned CG for six months, and that was it – he joined Square Enix as a game designer. I wanted that in our company as well. I don't mind too much if you don't have experience; our 3D modeller was an accountant! But he showed us some amazing 3D work he did on the side, and yeah, we hired him.

Our environment artist crossplays – he cosplays as female characters – which is great! It just brings more to the studio. That's my recruitment style – I got that from Square Enix.

DD: Different people have different opinions on studios. For me, I'm not a big fan of big studios, especially when you've just started – it's very difficult to share concepts behind ideas with a big team. The management is so complicated, so when you want to bounce off some creative ideas, you have to jump through a lot of hoops just to get stuff going. There are a lot of procedures and steps you have to take. I'm not a big fan of that. When we created this studio, we have a very simple system – there are no hierarchies in the office.

A later boss, dubbed 'mother and child,' involves fighting a young pianist and her gigantic, very angry parent.



Lead characters Mayday and Zuke live in the sewer, which means they're literally an underground band.

WH: It's flat.
DD: Yeah, it's a flat hierarchy. The two of us are the decision-makers at the top, so everything's a simple yes or no. But other than that, all of us are on an even playing field, so we can sit at a round table, and people can just bounce ideas around.
WH: A programmer can complain about the art.
DD: Exactly. Some of the beautiful effects [in *No Straight Roads*] are done by the programmer.
WH: Like the rockets and all that.
DD: If you are a programmer, you aren't *just* a programmer in our company. Anybody can be anything they want. The programmer can comment on the animation, the animator can comment on the story writing. So that's how we keep things – "Say anything you want". We use Slack, and there's no taboo subject. That's how people are always giving off ideas, and that's how we have a healthy environment, because nobody's caged in a creative way.
WH: The funny thing is, some of the most amazing ideas come from the fresh graduates. We're very lucky to have them. Not only that, but we want to make sure it's fast-paced. So we have a meeting every two weeks, we comment on all the deliverables, anything we want, and then we move onto the next thing. That's why the DJ [boss fight] is only one year and three months old, and the mother and child stage only took eight months to make from scratch. That's something I learnt from my earlier career as well, to keep the production fast – not too much focus on promotion rather than the game itself.

So you said you have 35 percent left to complete, roughly. What's left to do?

WH: We still have some boss fights. For the DJ and mother and child, they're a vertical slice of sorts, so we pumped it up all the way, but for some of the other bosses, we're still slowly building them up, because the final deadline's also the deadline for all the bosses. We're trying to make sure we've left enough time for the city, but the hub world's almost finished.

What's the game development scene like in Malaysia right now?

WH: We have a lot of game companies in Malaysia. We're known as an outsourcing country, but we have some companies that make amazing games. Some of the monsters in *Dark Souls III* were actually created by a Malaysian company, Passion Republic. *Final Fantasy XV* was [partly] Streamline Studios. We have a company working on the new *Command & Conquer* and *StarCraft: Remastered* and all that.

There's less original IP. I think the problem is positioning the game in the market. We Malaysians have two problems: we're either not proud of our culture, or we're too proud of our culture! So when we're not proud of our culture, we copy the Japanese anime style, or copy a Western style, or try to make a battle royale game, for example. Then we have people who are too proud of our culture, so they make Malaysian games that nobody other than Malaysians would want to play. So because of that, I've tried to bring back my know-how from *Final Fantasy* – that's the reason I came back to Malaysia, actually: to bring back that know-how – and make sure we all understand that once we make an action game, it'll sit beside *God of War* and *Spider-Man*.

***No Straight Roads* releases later in 2019 for PC and PlayStation 4.**

FINAL FANTASIES

"The thing I learned from *Final Fantasy XV* is relevance," Hazmer says, thinking back to his time at Square Enix. "We sat down, a bunch of us, with director Tabata-san, and we discussed it. "I think *Final Fantasy* sucks now; how do we make it more relevant?" So for us, relevance is very important. *Final Fantasy XV* had more people talking about it compared to earlier *Final Fantasies*. And we wanted to put that relevance into [*No Straight Roads*] as well."

The Embr corporation forces its employees to buy their own equipment, from fire hoses to axes. "Only 36 payments of $29.99."

Burning Ambition

Muse Games tell us about their gig economy firefighting sim, Embr

Info

GENRE
Simulation

FORMAT
PC

DEVELOPER
Muse Games

PUBLISHER
Muse Games

RELEASE
Early Access
Q4 2019

If you're aged 18 to 85, then you too can be a firefighter.

There's something worryingly believable about the premise behind *Embr*: that some blue-sky thinking Silicon Valley tech firm might come up with an app that turns ordinary civilians into firefighters. From its debut trailer, which manages to couch potentially horrifying possibilities in sunny terms ("Are you between the ages of 18 to 85? Then you're ready, hero!"), to its trendy minimalist title, *Embr* perfectly apes the Uber vision of the app-driven gig economy currently sweeping the globe.

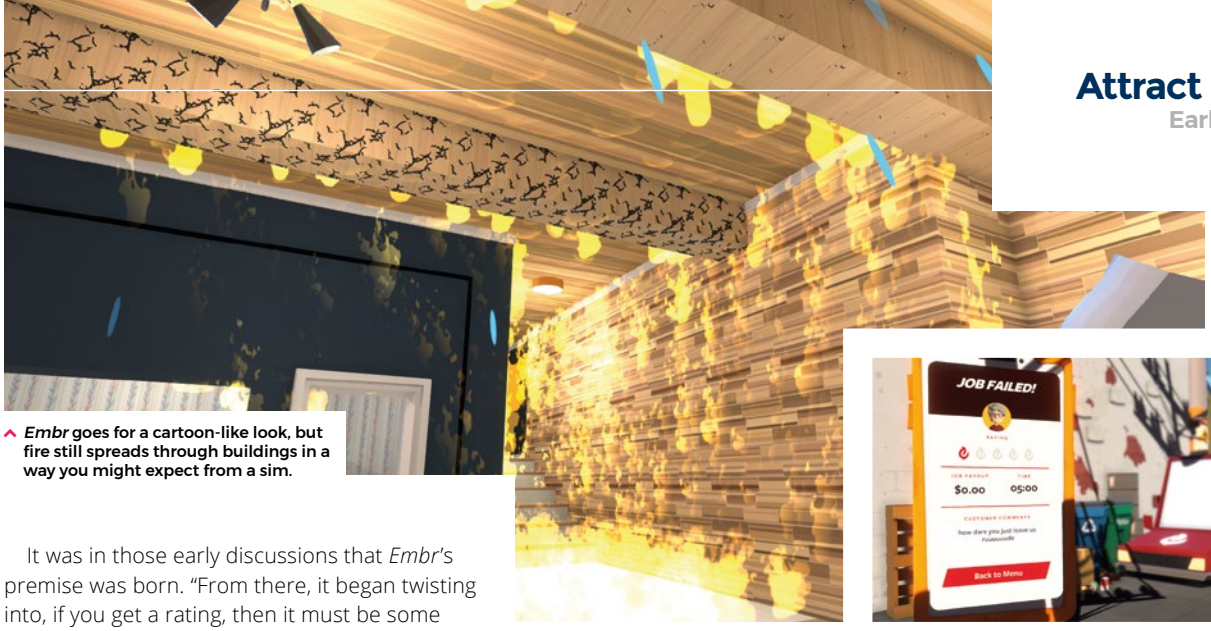
And yes, there's a compelling game underneath all this, too. *Embr* is a systems-driven co-op sim about life as an amateur firefighter: armed with a hose, axe, or trampoline, you respond to emergency services through the Embr app, rescuing civilians from assorted house fires and other minor disasters. *Embr* can be played solo, but it'll really come to life in its online co-op mode, where parties of four can team up to put out fires and earn income through the app. Before you know it, you'll be smashing down doors with fire axes and rescuing old ladies by throwing them out of first-floor windows while a friend tries to break their fall with a trampoline. It's all zany, hectic stuff, but according to Cameron Bajus, one of *Embr*'s lead developers, the game originally started out as a more serious firefighting sim.

Rescuing panic-stricken people stuck in bathrooms is par for the course in *Embr*.

"We really wanted to make a game that didn't exist," Bajus tells us. "A firefighting simulator game hasn't really been done well, so we wanted to take a shot at that. It started out as something of a more serious take on it, and there are still a lot of the systems in the game that reflect that: there's a very dynamic heat distribution system that's pretty realistic, and throughout the house, there are different materials that hold different levels of wetness. There's a high level of sim underneath everything, because that's how it started – as a very high-level sim."

While *Embr* was still in its early stages of development, however, Bajus and his team started playing around with its mechanics, and they quickly realised how comical the situations they'd created actually were.

"We hadn't added our art assets yet, so all of our rescue objects were just these tofu block rectangle things," Bajus recalls. "As we were playtesting around the office, the thing to do was just to throw them down stairs, throw them out the windows – all sorts of stuff. We were like, 'We want to lean into this funny part of the game, but we've got to make sure you don't feel like a really bad person for doing it.' So how can we make a firefighting game where, if someone dies, you don't just lose instantly? Is there a rating, somehow?"



Embr goes for a cartoon-like look, but fire still spreads through buildings in a way you might expect from a sim.

Mess up a firefighting job, and you'll get a stinging review on the Embr app – and, worst of all, no pay.

It was in those early discussions that *Embr*'s premise was born. "From there, it began twisting into, if you get a rating, then it must be some kind of [firefighting] service. We thought, 'Oh, you'd just call a firefighter from your phone.'"

As Embr firefighters, players will therefore receive alerts via the app, and depending on how promptly they rescue potential victims from a blaze, will be rewarded with a rating and a cash payment – which is just as well, because Embr firefighters also have to pay for their own firefighting equipment. "So basically you get to pay us for working for us!" beams Muse Games' marketing manager, Wendy Fritscher.

"It certainly references the gig economy that's currently happening in America," Fritscher tells us. "Then people buy that across the world because we're all very America-influenced. We definitely wanted to have a very self-aware and cynical approach to the way we presented the game to players."

"At Embr, we believe anyone can be a firefighter," Bajus confirms, adopting the soothing voice of a corporate advertisement. Playful satire aside, though, Bajus adds that he wants the game to give players a formidable opponent to pit their skills against.

"It certainly references the gig economy that's happening in America"

"The main enemy in the game is the systems, and your control over those systems," Bajus says. "So managing all the electricity sources, and the fires, the water sources, the gas, the physics system, and all those sorts of things. You interact with and use those systems, and navigate your way through the building to get to our, uh, valued customers inside the building... So interacting with all those things, and staying on top of it... it's impossible to control all of it at the same time."

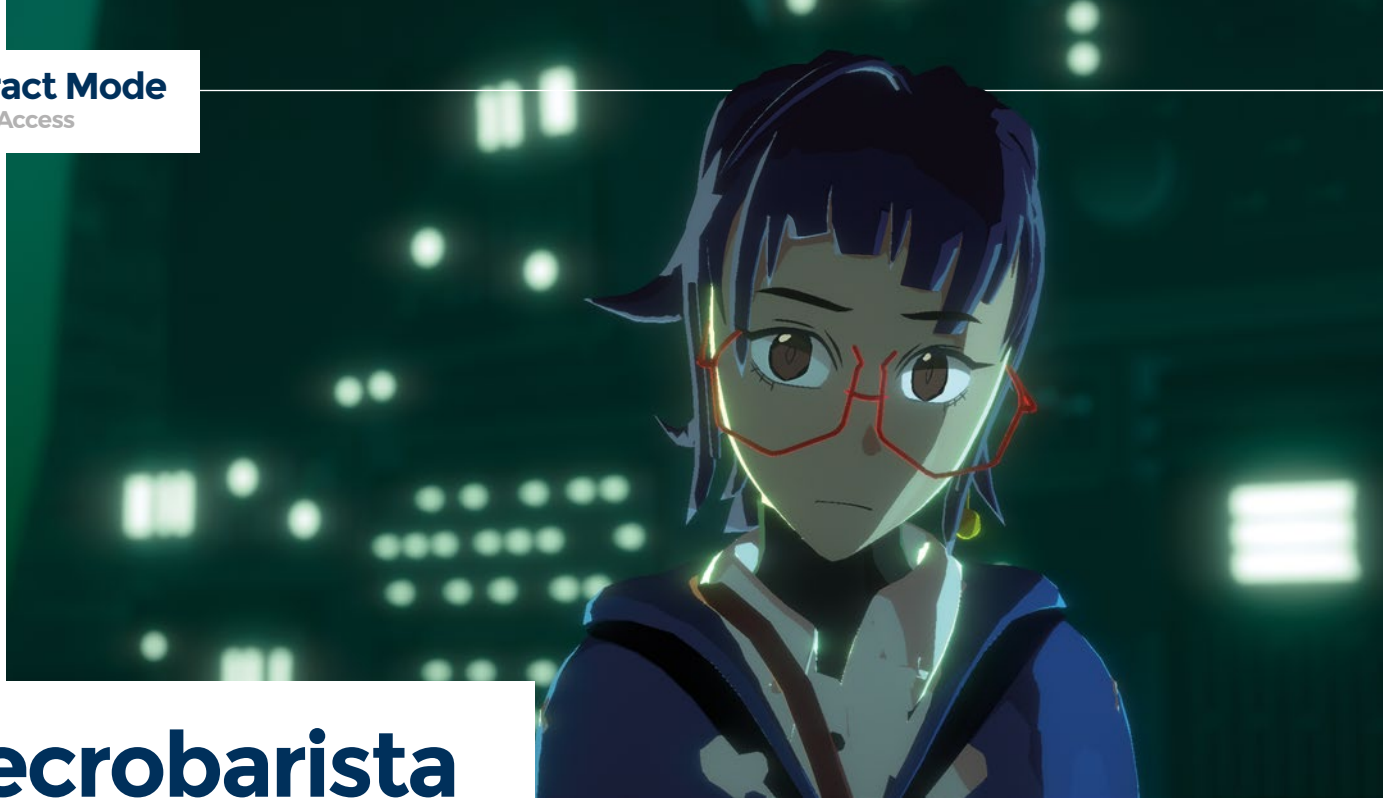
The challenge will be there if players want to find it – houses can apparently be cleared of occupants within seconds, if a team's efficient enough – but *Embr* is also being pitched as a game you could enjoy at parties, like *Overcooked* with fire hoses.

"There are different kinds of success you can have," Fritscher says. "You can save everyone in the house, because you have people in the game doing excellent teamwork. And then you can play this with friends at a party, with a beer in one hand and cigarette in the other, not really paying attention but still having fun. You're still gonna feel like you've succeeded – because if people get out, they pay you! They might leave you a bad review, though."

To add to the challenge, *Embr*'s "valued customers" behave in their own unpredictable ways.

CREATIVE MUSE

For New York-based Muse Games, *Embr* was born in the aftermath of *Guns of Icarus Alliance*, its multiplayer shooter that, six years after the PC version launched, got a PlayStation 4 port in spring 2018. Once that game shipped, the 14-strong team at Muse had a chance to take stock and look again at the other game ideas they'd previously set aside. Says Wendy Fritscher: "We're actually working on a few projects right now, with a core team of two to three people each, and then overarching animators and UX artists that we share between all of them to give us an overarching Muse Games unity." One of those games, *Project Fire*, soon became *Embr*; according to Fritscher, two other titles – *Project Oasis* and *Project Hora* – are still in development and, for now, under wraps. Muse's rodent-based brawler *Hamsterdam* (which we previewed in issue nine) is out this spring.



Necrobarista

Designing the perfect coffee cup isn't an easy task

Info

GENRE
Visual novel

FORMAT
PC / Mac / Switch / PS4

DEVELOPER
Route 59

PUBLISHER
Coconut Island Games

RELEASE
2019

Necrobarista is a take on visual novels direct from Melbourne, Australia. Fittingly for the city's culture, the game takes place in a coffee shop – oh, and it also serves as a purgatory for the departed, with the dead having 24 hours in the café before going to the 'other side'. The story carries a mix of dark themes and a distinctly Australian sense of humour, focusing on smart and modern dialogue.

But right off the bat, it's easy to be confused about whether or not *Necrobarista* is a conventional visual novel. Developed in full 3D and making use of dozens of camera angles for each scene, it's more similar to what you might see in a feature film. But then, developer Route 59 always thought of its project as the "next big leap for visual novels," according to Kevin (Hsun-Yu) Chen, director and lead developer on the game.

A concept that started as an homage to anime ended up finding its own unique aesthetic.

Anime was also a big inspiration for the art style. With Australia's proximity to Southeast Asia, the region's influence plays a major role, and everyone on the team is a big fan of Japanese media. But since working in animation requires a lot more manpower, Route 59 found the alternative path of visual novels to be the perfect choice.

"They're in a middle ground between something that is achievable – like a game, because we do come from a game design background – but also something that is very anime, in terms of aesthetics," Chen says of his game's visuals. "As time went on, we started departing from that anime aesthetic to start forming our own, which I think is a natural result of any creative endeavour."

Along with the detailed 3D environments, the way the story cuts and transitions from scene to scene intrigues from the get-go. Every scene, whether it includes text or not, has to be manually progressed with a click or button press. In the opening sequence, for example, one of the characters asks if the light can be turned on, and the game then switches the camera to focus on a lightbulb, which is activated with the player's input. Similarly, a clock is ticking and the animation doesn't stop until it's progressed by the player.

But why is that important? Well, Chen admits this is one of the studio's favourite aspects of visual novels and comic books, calling it 'player-directed pacing'. While the genre doesn't allow for much player agency, following the narrative and engaging in specific

The departed aren't the only ones allowed in the cafe, and it can be hard to distinguish the dead from the living.

tasks to progress means you still exhibit more control than when you're, say, watching a film.

"By giving players that single mouse click to progress, we allow them to dictate the pace at which they personally enjoy the shots," Chen says. "Having little actions like turning the lights on, or a character posing before they say something, or a transition between shots – those are all ways in which we give players a chance to pace the visual novel."

Route 59's Brent Arnold holds the role of director of photography, and during the course of *Necrobarista*'s development, he adopted a pipeline structure similar to one commonly seen in the film industry – composition, lighting, and staging are all considered, making this a more cinematic experience than we might usually expect from a visual novel.

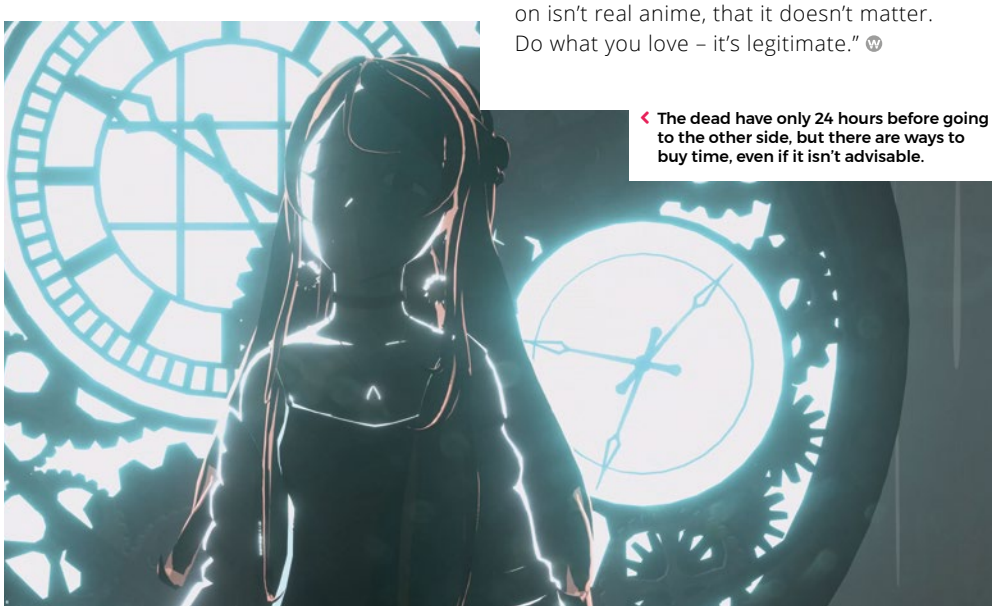
Necrobarista also uses a simple strategy to immerse the player in its considerable lore. Every now and then, players will see key words highlighted in the dialogue; checking out that word results in an additional pop-up bit of prose explaining a specific term, or maybe providing a brief bit of background information on the character that is currently speaking, for example.

"*Necrobarista* adopted a pipeline structure similar to that of the film industry"

This is similar to Supergiant Games' approach with its 2017 RPG, *Pyre*, with these highlighted words and phrases working almost like Wikipedia entries to explain the game's backstory in short bursts. In *Necrobarista*, the concept is taken a bit further: they're also used as in-game currency to unlock optional short stories at the end of each chapter. These range from being able to read The Council's rulebook to checking out longer, more text-heavy stories that fill in those missing gaps in the game's lore.

This mechanic is seen as the perfect means to introduce more about the world without suffocating the player with long, experience-disrupting reads. Chen laments the fact that it's common for visual novels to be extremely exposition heavy, where players can spend a lot of time just having lore explained to them. *Necrobarista*, meanwhile, sees a main story understandable on its own, but with more to discover for those interested in uncovering it.

Really, it all goes back to that love for Asian visual novels and the anime style – even if *Necrobarista* isn't 'pure' anime as such. "There are a lot of die-hard fans out there that would [defend] anime as exclusively from Japan," Chen says. "But I think anime is an aesthetic, in a sense. I really want to show to the few out there who maybe think what they're working on isn't real anime, that it doesn't matter. Do what you love – it's legitimate."



The dead have only 24 hours before going to the other side, but there are ways to buy time, even if it isn't advisable.

THE LAND OF COFFEE

A large Greek and Italian population means coffee culture didn't take long to arrive in Australia, and the people in Melbourne take coffee very seriously. The city even hosts the Melbourne International Coffee Expo every year, and is home to the highest density per person of cafés (and restaurants) in the world. Since reflecting this was really important for Route 59, the team put a lot of work into its own virtual café. There are at least four types of coffee in the main story alone, and Chen says t took a while to design the perfect coffee cup: "You can't make it look too fancy because it's unrealistic," he says. "But if you make it look too plain then it's just a cup. You have to strike that perfect balance, while at the same time making sure it looks very Melbourne."

Headlines
from the virtual front

01

02

03

01. Gaymes for all

A new gaming magazine with a specific focus on the LGBTQ+ community is set to launch soon, with Gayming Magazine aiming to look at the world of gaming 'with a queer twist'.

As well as news, interviews, and features with a focus on LGBTQ+ companies and designers, Gayming will also aim to bring together members of the community through both the magazine and local groups. The online magazine will launch June 1.

Elsewhere, a few members of the old CVG team announced they would be reforming to launch the news-focused Video Games Chronicle, under the Gamer Network banner. It's all go in the games media [please subscribe to Wireframe].

02. The widening gap

The UK games industry has seen its gender pay gap widen in the last year, with figures for 2018 showing an increase of 3.5% in pay disparity. While the national median wage gap stands at 9.6%, the median wage gap in the games industry specifically – for the 19 companies reporting their figures – stands at a massive 18.8%.

Rockstar North stood atop the pile with a median wage gap of 34.4%, followed by Codemasters' 33.3% and Sumo Digital's 32.6% – though it is worth pointing out all those companies saw their median gaps fall since 2017's report.

On the better end of the scale, Namco reported a 4% median wage gap, while retailer GAME had the smallest difference, with a median wage gap of 0.3%. No company reported parity or women earning a higher median wage than men.

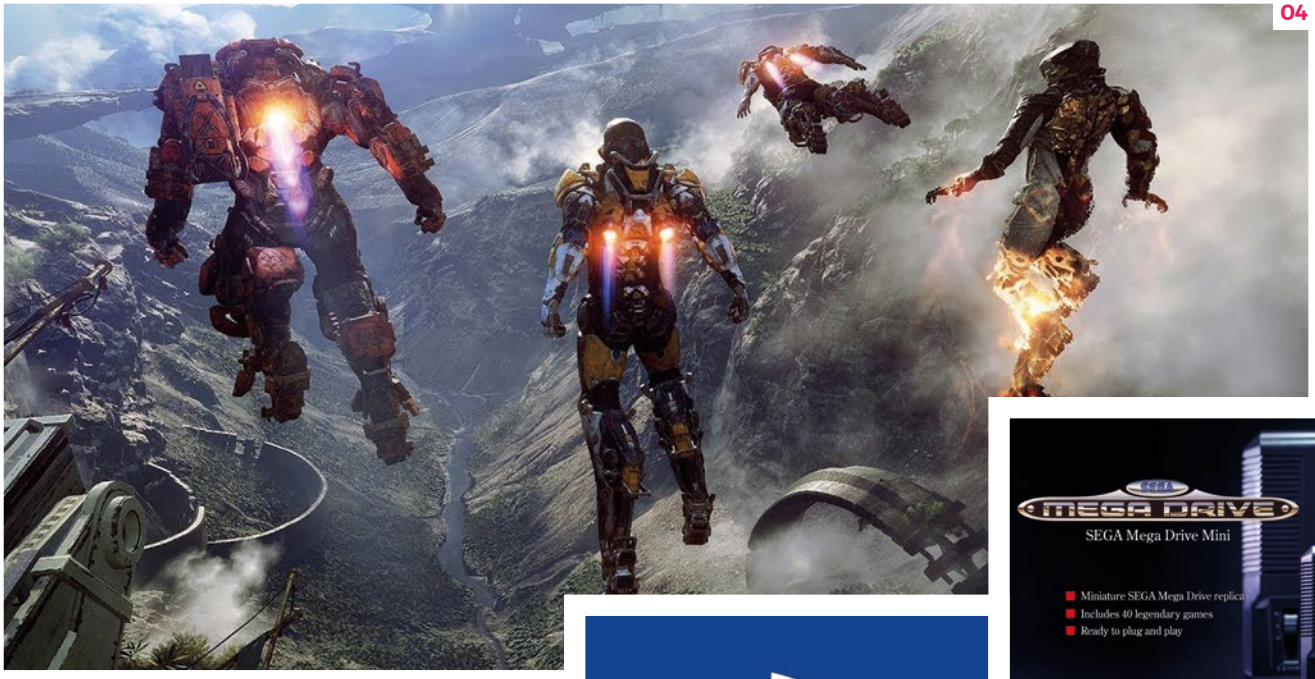
03. EC vs Valve

The European Commission has turned its eye towards Valve and its geo-blocking practices, stating: "In a true Digital Single Market, European consumers should have the right to buy and play video games of their choice regardless of where they live in the EU. Consumers should not be prevented from shopping around between Member States to find the best available deal."

This Statement of Objections didn't just focus on Valve, with publishers Bandai Namco, Capcom, Focus Home Interactive, Koch Media, and Bethesda parent ZeniMax also named. The EC sees it as Valve and the publishers entering a bilateral agreement to engage in geo-blocking, in short. An investigation is ongoing.

'Skyrim granny' Shirley Curry will be a character in the *Elder Scrolls VI*

Real-life lobbyist banned from *EVE Online* for corruption; virtual world proves sense of justice



04

06

04. Anthemic exposé

An in-depth Kotaku report (**wfmag.cc/Anthem**) delved into what went wrong with BioWare's lacking MMO *Anthem*, highlighting confused management, a lack of focus, and a desire to appease EA bosses over maintaining a specific vision of what the game could and should be. Which makes sense, having played the game, and goes some way to explaining how a survey on Reddit showed the game has lost around half of its player base since launch.

Said report resulted in much feedback online, as well as an admission from BioWare GM Casey Hudson that 'these problems are real' in an internal memo leaked after the article's publication. Changes are said to be in the works at the studio, though time will tell – and the proof will be in the as-yet-unannounced-but-definitely-exists *Dragon Age 4*, which is sure to serve as a litmus test for the state of BioWare when it releases.

05

05. PsstS5

Details of the as-yet-unnamed PlayStation 5, which won't be launching in 2019, have been revealed. System architect Mark Cerny, speaking to Wired, says the new console will feature an eight-core CPU based on AMD's third gen Ryzen processors, as well as a custom GPU based on the forthcoming AMD Navi line. It will be capable of real-time ray tracing, so we're going to see some stunning puddles in the next console generation.

The PS5 will also feature an SSD as standard, a custom sound chip to support 3D sound, and VR support as standard. It will also play games from disc.

06. Where's AtGames at?

Sega is bringing a Mega Drive Mini (or Genesis in the States) to the world, with the scaled-down console arriving with two controllers and a sweet 40 preloaded games on 19 September, priced at £70 or thereabouts. We've known it was coming for a while, and expectations are high following Nintendo's mini-output (as well as Sega's abandonment of AtGames' lacklustre previous efforts), so... well, we're excited about it.

Sonic, *Ecco*, *ToeJam & Earl*, *Altered Beast*, *Space Harrier*, *Gunstar Heroes* – the mix of titles announced so far (though not games will be the same in every region) is enough to make us salivate. Just add *Contra: Hard Corps* to the list and we'll pre-order three. Each.

Tim Sweeney: 'Epic Store will keep getting exclusive games', mainly because capitalism is a thing

Gaben's face used to market underpants in China, naturally

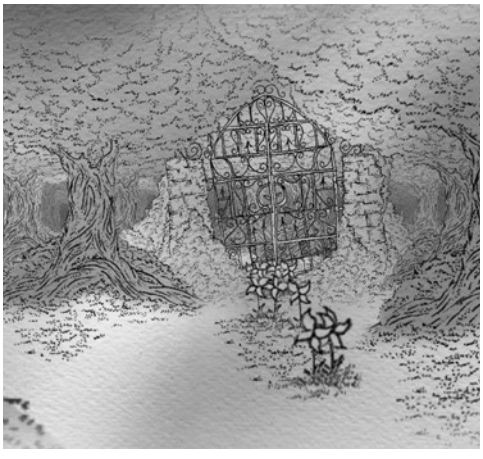
incoming

R-Type Final 2

Given the timing of its announcement on the 1st April, we assumed *R-Type Final 2* was a joke at first. After all, original developer Irem swore that *R-Type Final*, released back in 2003, was as definite an ending to its horizontal shooter series as its name suggested. But thanks to Japanese studio Granzella, the 32-year-old name is indeed being resurrected, and is currently planned for release on the PlayStation 4. And while details are few right now, we do know that *R-Type Final 2* is being headed up by Kazuma Kujo, who directed the last game, and that it'll likely be crowdfunded. An accompanying teaser trailer suggests that Granzella has at least built some kind of prototype for the sequel, though, and it all looks reassuringly familiar: there's a lone R-9 spaceship, joined by its indestructible Force satellite, charging into battle against a plodding army of biomechanical enemies. It remains to be seen whether *Final 2* can find a way of updating the 32-year-old formula for a new generation, but if the sequel can at least match the best moments of the earlier entries, then its huddle of fans will probably be happy.

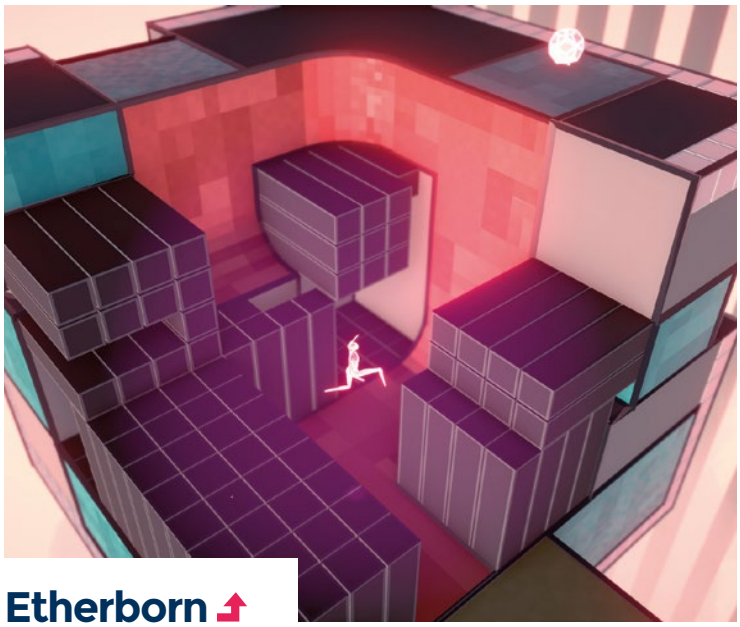
Collage Atlas

Illustrator and developer John Evelyn's delicate pen-and-paper drawings spring to life in the first-person adventure, *Collage Atlas*. Evelyn says the game takes in such universal themes as memory and hope, while its world design is inspired by such artists as Naohisa Inoue and *Where The Wild Things Are*'s Maurice Sendak. We're looking forward to bathing our withered souls in Evelyn's monochrome fantasy world: with hand-drawn textures applied to 3D models, it creates the illusion of diving headlong into a fantastical pop-up book.



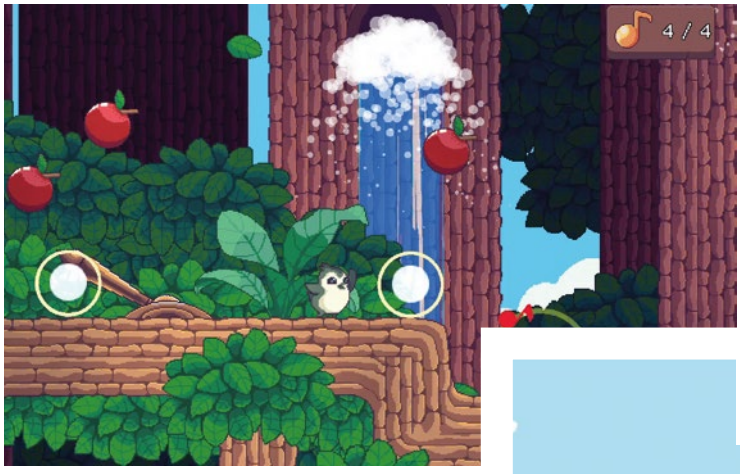
Pacer

Formerly known to its friends as *Formula Fusion*, gravity-defying future racer *Pacer* is in the process of being heavily retooled ahead of its debut later in 2019. The debt to *Wipeout* is clear, but *Pacer* handles nicely so far, while developer R8 Games has come up with a wealth of newfangled modes – including its own take on the trendy battle royale genre.



Etherborn

From gravity-defying racers to a soothing puzzler that also plays with the laws of physics. Developed by Altered Matter, *Etherborn* has players roaming a series of 3D locations where gravity is constantly changing, and your path to the next waypoint is littered with strange objects and disembodied voices. It's all a bit like shopping in Ikea, then.



Songbird Symphony

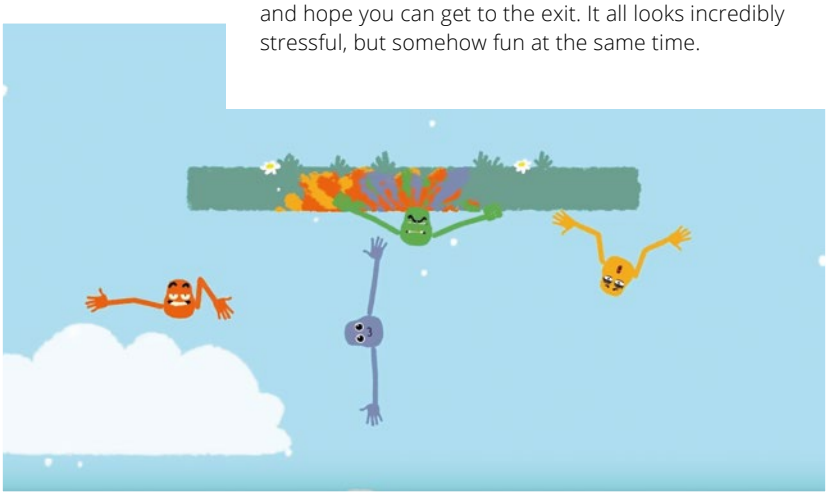
A tiny bird with an identity crisis heads off on an adventure in this likeable hybrid of platformer and rhythm-action game. There are items to collect and a gallery of characters to meet, but the main draw is the musical set-pieces, where battles are won through carefully timed chirps (or button presses, if you're not a tiny bird). Expect cute pixel art and catchy tunes in abundance.

Omno

Tucked away in the Leftfield section at this spring's EGX Rezzed was this little gem from Germany's Studio Inkyfox. Infused with a dash of *Journey*'s zen-like calm, *Omno*'s an adventure game that takes in exploration, puzzle-solving, and uncovering new powers. What most caught our eye is its character and environment design; from its spindly-legged hero to its lush fantasy landscapes, *Omno* is a stunning achievement – particularly given it's the product of just one developer.

Heave Ho

Here's the perfect game to play in your living room with friends – assuming your friends aren't too bad-tempered, that is. It's a little like *LocoRoco* crossed with the constant deaths of *Super Meat Boy*: the aim is to swing and flip your fragile characters across hazard-filled landscapes and hope you can get to the exit. It all looks incredibly stressful, but somehow fun at the same time.



PICO-8

your new favourite console

Virtual console PICO-8 proves that less is more when it comes to making games. We find out what it is and how to get started

WRITTEN BY
DAN LAMBTON-HOWARD

PICO-8's palette of 16 colours is perfectly suited to making big, bold, and beautiful 8-bit sprites.

A million miles away from the complexities of developing with Unity and Unreal Engine 4 lies PICO-8. Conceived as a fantasy console for making, sharing, and playing tiny games and other computer programs, PICO-8 has no hardware, yet contains everything you need to make complete games. It includes editors and tools for writing code, drawing sprites, building maps, and composing sound and SFX, all within a 128×128-pixel window.

When you boot it up, you get a splash of coloured pixels, a cute digital chirrup, then a blinking cursor. Created as an homage to the 8-bit computers and consoles of the eighties, PICO-8 is a stripped-back game development environment that seeks to capture the look and feel of retro bedroom coding. It does this through big, chunky sprites, beeps and bleeps, and by imposing strict limitations on code and memory.

Since its release in 2014, PICO-8 has become a mainstay of game jams thanks to its striking aesthetics, approachability, and ease of use. This has led to the growth of an active and supportive community of indie game developers, code-tinkerers, hobbyists, and retro gamers who are constantly adding to the ever-expanding library of cartridges for the console, all of which are accessible by using the console's in-built cartridge browser, SPLORE.

REMAKES AND DEMAKES

For a certain group of fearless developers, PICO-8's limitations offer an even greater challenge: squeezing big names onto a cartridge. The task of remaking, or 'demaking', popular games is not one to be undertaken lightly, and involves trimming a game down to its essence. And giving it a name that would make any tribute band blush with pride. Notable games that have undergone the PICO-8 demake treatment include *No Man's Sky* (*Low Mem Sky* by Paul 'Liquidream' Nicholas, *Spelunky* (*Delunky* by Johan Peitz), and *Nuclear Throne* (*Nuklear Klone* by Frederic Sochu).

"The first time you run PICO-8, there are enough clues to install, load, and run some demo carts," White says. "It's a little adventure itself."

FROM PICO TO GOTY

The indie smash hit *Celeste* – recent winner of Indie Game of the Year – began life on PICO-8. Created by Matt Thorson and Noel Berry in just four days for a game jam, *Celeste Classic* has only 30 levels, but shares many of the same elements as the final release, such as tricky platforming, the red-haired protagonist with her signature mid-air dash, the eponymous mountain, and those ever-elusive strawberries. Such was Matt and Noel's love for the PICO-8, that in the final release you can actually find a 'physical' console hidden within one of the game's levels, complete with a monitor and joystick and loaded with a copy of the classic game. You can also play it for free online: **wfmag.cc/DzoCpY**

LESS IS MORE

Joseph 'Zep' White created PICO-8 in response to his early memories of writing code on his father's BBC Micro, plotting large colourful pixels on a clunky keyboard and "trying to construct anything even remotely resembling a playable video game." In order to rekindle this feeling, White started to build tools which would be just as exciting to use as those early computers. He explored and experimented with different specifications to find a sweet spot, finally settling on a 128×128 resolution, fixed 16-colour palette, 32kB cartridges, plus a D-pad and two buttons for player input.

Far from arbitrary, these constraints help create a consistent look and feel across the console's games, and were chosen in order to encourage a certain design culture and development experience. "You can tell at a glance that a game is a C64 game because of the palette," White says. "The particular combinations of hardware and data size restrictions for each system gave rise to separate pockets of game design trends. Each system felt like a material, and working with the grain would produce something that carried the character of that material with it. So that became a guiding principle of how to design PICO-8: each design decision and choice of limitation should contribute to the overall character of the machine in a meaningful way."

> "The constraints of PICO-8 guide you towards a solution"

Importantly, they were also chosen so that PICO-8 would be a place for games that are "fun to design but not laborious to implement." This is why PICO-8 is worthy of your attention: it offers one of the most enjoyable and accessible game dev experiences around.

PIX APPEAL

Part of the fun of PICO-8 is the speed at which an idea can go from existing in your head to being fully implemented (for proof, flick ahead to the 'how to get started' section). This short and satisfying feedback loop is a refreshing antidote to working with modern engines, and makes prototyping ideas easy and effective. This is partly the reason why it's been so readily adopted as an engine of choice for game jams – events where small development teams or solo creators compete against the clock to build games, encouraging a focus on small games with limited scope. As such, they play perfectly to the strengths of PICO-8.

Ayla Myers, an independent game developer from New York, has made many games for PICO-8, most notably the addictive action title *Just One Boss*, and given talks on the joys of PICO-8 token crunching. Referring to the process of streamlining code in order to squeeze in features, she argues that the console's artificial limitations actually focus her design process. Says Myers: "You end up making these really weird decisions like, 'Either the player can have a hat or the boss can have this extra attack, but not both'."

Far from being a frustrating experience, these limitations actually allow a creator to focus more on the game itself, and how each element contributes to the whole. "It's amazingly freeing," Myers says. "There's no time spent figuring out how you're going to render anything or what shaders you need to build or anything like that. You have 16 colours. You have 128×128 pixels. Make something beautiful."

THE MEDIUM IS THE MESSAGE

The limitations of early computers and consoles fundamentally informed game design choices: tiling and reusing sprites to save memory are decisions that emerge from dealing with restrictions on memory and processing power.

Quick prototyping on PICO-8 laid the foundations for the multi-award-winning *Celeste*.

Delunky is a surprisingly faithful PICO-8 recreation of the excellent roguelike *Spelunky*, but in 32kB of memory. Witchcraft.

"I love the constraints of PICO-8 because they guide you towards a solution," Myers says of working on *Just One Boss*. "I wanted to make a big bad boss, but knew that I wouldn't get more than ten frames of animation to fit into my sprite sheet. So I opted instead for a boss with floaty hands that could imply a body without having to actually draw one. And the game itself is largely a product of those types of decisions, to the point where I would not at all have made the same game if I'd built it on something other than PICO-8. And I think that's a really beautiful thing."

These restrictions force a creator to evaluate every element in their game and how they work together. Does this enemy type work well with the player's abilities? How can I make the most of this animation cycle? Can I reuse these explosions elsewhere? Not only is thinking like this generally good practice in game design, but it also forces creators to limit their vision to a manageable scale, helps avoid feature creep, and focuses thinking on one or two core mechanics instead of complex systems. This also means that developing on PICO-8 is a very beginner-friendly experience.

The PNG files of a PICO-8 cartridge actually contain the entire game encoded within them. And they're cute, to boot.

Rémy 'Trasevol Dog' Devaux is a regular contributor and organiser of game jams with PICO-8. His games, like the superb *Gar's Den*, are full of lively dancing characters, swirling vortexes of colour, and punchy game feel.

For him, part of the console's appeal is the speed at which you can prototype and test an idea. "Game feel is a huge focus when I work on a game," Devaux says. "PICO-8 does help with this because you can just press **CTRL+R** and that will instantly recompile the game and show the changes you just made. It's great for iterating over details."

Like Myers, Devaux's aesthetic comes from working with PICO-8. "The art-style used in *Gar's Den* is very much one I came up with and developed in PICO-8 over the two years I'd been using it with my different games. The 8×8 dancing character, the use of the palette to create texture, the hazy background... All these things were born in PICO-8, from its constraints."

THE PEOPLE MAKE IT

Myers and Devaux are part of a community of makers and sharers that has built up around PICO-8. This community, often cited as one of the best reasons to work on the console, is voraciously active in the creation of new carts and in the sharing of helpful tools and catalogues of resources. They're also welcoming to newcomers. What is it about PICO-8 that creates such a positive community? For White, it's the console's immediacy. ➧

Joseph 'Zep' White launched PICO-8 in 2014. As well as creating fantasy consoles, he co-runs Pico Pico Cafe, an event and meeting space for artists and developers in Tokyo.

Ayla Myers works with PICO-8's limitations to create delightful experiences, packed with polished pixel art and slick game feel.

CART-CEPTION

PICO-8 can export as standalone executables for Windows, Mac, and Linux, and as HTML for embedding on websites like **itch.io**. However, in possibly the coolest way to export a game ever, you can export as a PNG.

This produces a cute little picture of an actual game cartridge with your own choice of cover art, while the entire game is encoded within the file itself, making it shareable, super cool, and likely to prompt onlookers to gasp, "What do you mean the game's *in* the picture?"

> "You have to build everything up from scratch"

"PICO-8 doesn't hold your hand at all," White says. "There's no built-in collision detection or animation. You have to build everything up from scratch, or grab snippets from the forums to do simple things. But I think there's value in this approach; of having nothing magic happening underneath. You can see how everything works, right down to the handful of primitive functions that PICO-8 supplies, like drawing a pixel on the screen. I think it's very satisfying and empowering to know that you built something from scratch and understand it down to first principles. And when it comes down to it, it doesn't actually take much code to create something that is exciting for a beginner – or for anyone else, for that matter."

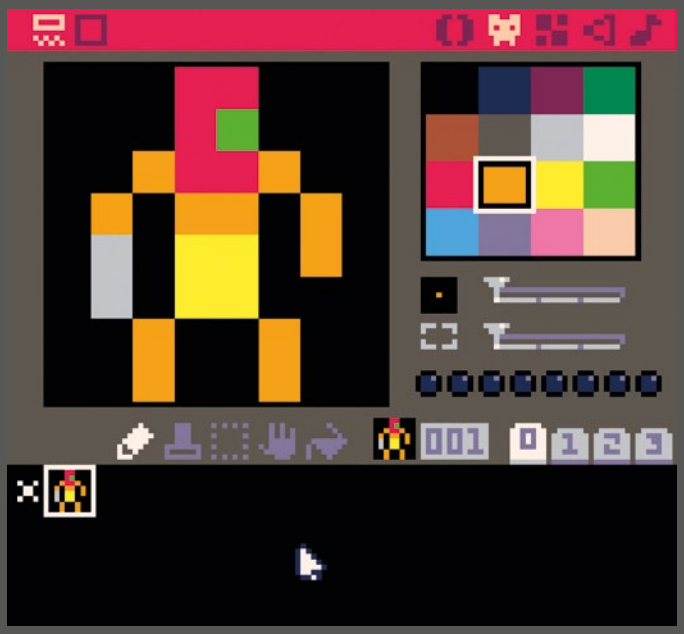
HOW TO GET STARTED

By now, you may be wondering how you get your hands on PICO-8. The first step is to buy a licence from the Lexaloffle website (**wfmag.cc/pico-8**) which will allow you to download PICO-8 on Windows, Mac, and Linux. At the time of writing, this will set you back $15, or about £12. Once it's installed, boot it up, and type 'SPLORE' into the command prompt to open the cartridge browser, which will give you access to the entire library of games and software available.

Once you've found one you like, press **ESC** a couple of times to check out the code. **ESC** will also let you move between the command prompt and the editors. The tabs at the top-right of the editor will let you move between screens so you can view the sprite, code, map, sound, and music editors. Let's look at each screen in turn, and get something moving.

#TWEETCART

A testament to the ingenuity of the PICO-8 community, the Twitter hashtag #TweetCart is chock-full of GIFs of procedurally generated art, tessellating vortexes of colour, and little playable games. All of these are created under the condition that the code must fit entirely within a single tweet. New ones are made almost daily, and creators always tweet the source code so you can see exactly how the sorcery is performed.



SPRITE EDITOR

If you've ever used MS Paint, you'll be comfortable with the sprite editor. On the left, you have your sprite, on the right, the colour picker, and in the middle, various paint tools. At the bottom is your sprite sheet (split over four tabs). Each 8×8 tile is indexed. Choose index 001 and draw a character in the sprite window. You can see I've gone for a tiny Samus Aran. Isn't she cute?

CODE EDITOR

PICO-8 uses Lua as its scripting language. It's a fast and easy language to learn and write. At the top-left of the code editor are tabs for different pages to help keep your code organised. PICO-8 doesn't have lower case letters and uses its own font, which makes things simple, if a little hard to read at times (you can also use an external text editor if you prefer). In this example, we've written code in PICO-8's three special functions `_init()`, `_update()` and `_draw()`. The code declares a player, then moves that player if direction buttons are pressed, plays sound effects, and draws everything to screen. If you've never coded in Lua before, then check out the 'further resources' section for additional help.

```
function _init()
 --everything in here happens once as
soon as your cart starts

 --create a new table named player,
containing x and y value
 player = {x=64,y=64}
end

function _update()
 --everything in here happens 30
times per second

 --update the player's location if a
dir button is pressed
 if btn(0) then player.x-=1 end
 if btn(1) then player.x+=1 end
 if btn(2) then player.y-=1 end
 if btn(3) then player.y+=1 end

 --play SFX 0 if player hits the 'z'
key or controller button
 if btn(4) then sfx(0) end

end

function _draw()
 --everything in here is drawn 30
times per second

 --clear the screen
 cls()

 --draw a 16x16 tile section of the
map from index 0,0 (top left)
 --at point 0,0 on the screen (also
top left)
 map(0,0,0,0,16,16)

 --draw sprite 1 at player x and y
position
 spr(1,player.x,player.y)
end
```

MAP EDITOR

The map editor is like the sprite editor. Here you use the sprite tiles you've created to paint levels for your game. Each tile on the map screen contains a reference to a sprite in the sprite sheet. Create a new terrain sprite in the sprite editor in index 002 and then select this sprite in the map editor to draw a 16×16 room in the very top left of the map screen. You can see I've gone for some suitably alien-looking rocks and built a spooky cavern. What mystery awaits within?

GETTING THINGS RUNNING

Now we've drawn some sprites, written some code, and made a sound effect, let's see it all working together. Press **ESC** to exit the editors and return to the command prompt. Enter 'RUN' to see your hard work come together. You should have a little sprite that moves around the screen when you press the arrow keys. Well done! Press **ESC**, then type 'SAVE MYFIRSTGAME' to save a copy of the game cart.

SFX EDITOR

The SFX editor is split into two views. Pitch mode (pictured) presents an area in the middle of the screen where you can drag and drop the frequencies of notes. Use the volume sliders beneath to control the volume of each note, and the instrument selectors to choose between eight synthetic instruments, each with their own character and colour. Controls at the top let you navigate between effects, and control playback speed and looping. Tracker mode (toggled in the top left-hand side of the screen) is more suited to music, and presents each note in detail – perfect for precise editing. For now, make sure you have SFX 00 selected in the top left and play around with drawing notes in pitch mode with different instruments. Press **SPACE** to hear your creation. I've made a random retro squeak noise – ideal for alien impacts.

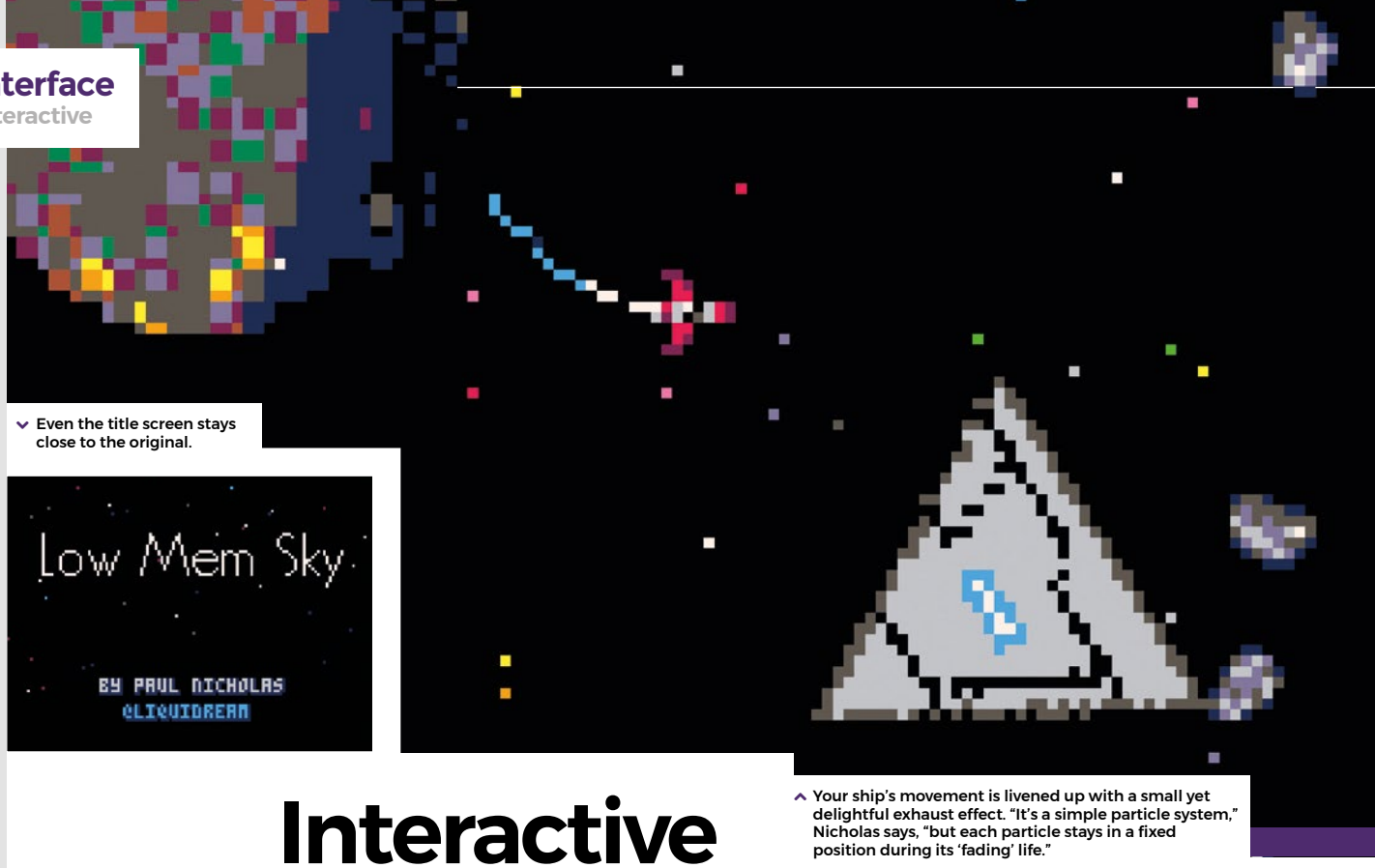
MUSIC EDITOR

Music in PICO-8 is controlled by a sequence of patterns. Each pattern consists of sound effects playing on some or all of PICO-8's four audio channels. These patterns can be looped and started and stopped in code – for example, at the start of a level or menu screen. At the top of the editor, you'll find the pattern selector. To the right are the controls for looping and stopping patterns. Below this are the four audio channels, each can be set either off, or to a SFX. In the example pictured, you can see a music track made up of SFX numbers 10–13. For more help with music and sound, look at Gruber's excellent video series in the 'further resources' section.

FURTHER RESOURCES

This basic example shows how easy it is to get something moving in PICO-8. If you want to learn more, there's a wealth of advice available to you, as well as a supportive community willing to help. Here are some good places to start, and you'll find more resources on page 42.

- The Lexaloffle website has a great selection of resources and community links. **wfmag.cc/lexaloffle**
- Lazy Devs Academy has a YouTube series that follows the making of games from start to finish – perfect for beginners. **wfmag.cc/lazydevs**
- Gruber, PICO-8's resident composer has a YouTube series about making music. **wfmag.cc/gruber**
- PICO-ATE is an encyclopaedic compendium of helpful links. **wfmag.cc/pico-ate**
- See who's posting on **#PICO8** on Twitter, or visit the Lexaloffle forum above. You're sure to find someone willing to help.



Even the title screen stays close to the original.

Your ship's movement is livened up with a small yet delightful exhaust effect. "It's a simple particle system," Nicholas says, "but each particle stays in a fixed position during its 'fading' life."

Interactive

Low Mem Sky

Meet Paul Nicholas, who managed to squeeze a 2D demake of No Man's Sky onto PICO-8

Are you a solo developer working on a game you want to share with Wireframe? If you'd like to have your project featured in these pages, get in touch with us at wfmag.cc/hello

Low Mem Sky creator Paul Nicholas, in appropriately 8-bit form.

No Man's Sky captured imaginations from the moment it was announced almost six years ago, and while its 2016 launch wasn't exactly smooth, subsequent updates have seen it flourish into an absorbing and technically stunning space sim. In its own way, pixel tribute *Low Mem Sky* is another impressive feat of programming.

Created by software developer Paul Nicholas, it's a 2D, top-down rendering of the *No Man's Sky* concept – a procedurally generated galaxy to cross in a little ship, planets to land on and explore – all crammed into the fantasy console, PICO-8. As you'll discover in our feature on page 18, PICO-8 is a development environment that's defined by its strict limitations; with its 128×128 pixel display and 32kB cartridges, it's perfect for the rapid creation of tiny, focused games. But as *Low Mem Sky* proves, PICO-8 also provides another worthwhile challenge: finding out just how far its limitations can be pushed.

"At first, I started creating a few small games and demos, but the projects steadily grew in scope and complexity," Nicholas says of his early brushes with PICO-8. "My first big project was *SCUMM-8* – my demake of the classic point and click adventure game engine, which was also my first open-source project on GitHub. *Low Mem Sky* was my first proper venture into procedurally generated content. I've always been a fan of such games, ever since *River Raid* on the Atari, with its seemingly never-ending levels. My admiration for procedural generation grew when I used to play *Frontier: Elite II* for hours on my Amiga 600 back in the day. It was fascinating to see that [David Braben] managed to fit an entire galaxy on a single floppy disk."

"I used to play *Frontier: Elite II* for hours on my Amiga 600 back in the day"

The idea of making a lo-fi interpretation of *No Man's Sky* first came about in the summer of 2018, when Nicholas took part in a game jam dedicated to making unofficial, 8-bit tributes to famous games – click on the #DemakeJam hashtag on Twitter, and you'll find blocky renditions of everything from *Ocarina of Time* to *Overcooked*. One of the main rules that entrants were asked to abide by was that their games be completed within eleven days; a tall order for Nicholas, particularly given the number of elements he needed to cram into just 32kB of memory.

Low Mem Sky's planets teem with alien life.

Just like *No Man's Sky*, you can explore planets and collect resources.

TINY SPACE

"I think the biggest challenge was mostly the act of squeezing the various aspects – galactic map, space flight system, planet surface exploration, trading and inventory system – all into a single PICO-8 cart. It needed to have all of those core features in, otherwise, it wasn't really a proper demake. It would've been much easier to split the game across multiple PICO-8 carts, but for me, having it all within a single cart was an important aspect of the whole challenge."

Fortunately, PICO-8's open community of developers regularly make their work freely available for other users, meaning Nicholas could use existing routines to get *Low Mem Sky* up and running.

"This jam was very short – we only had eleven days – which was just about enough time to create the initial version in my spare time. I had a bit of a head start though, as I'd recently combined a couple of existing routines – PICO-8 ports of OpenSimplex noise generation by Felice Enellen, and Marching Squares by Frederic Souchu – to create a proof-of-concept of an infinite smooth-scrolling 2D terrain in PICO-8. In fact, *Low Mem Sky* wouldn't have been possible (at least within the jam period) were it not for other members of the PICO-8 community making their code snippets available."

The result is a charming and surprisingly atmospheric space exploration sim, as you hop from planet to planet to the echoing strains of Chris Donnelly's superb chiptune soundtrack ("He's even had requests for digital copies of his 8-bit renditions of the original *No Man's Sky* music," Nicholas tells us). And then there's the scale to consider: *Low Mem Sky* can't quite match the 18 quintillion planets Hello Games managed in their opus, but Nicholas' tribute still has something in the order of 180 billion worlds, each with their own procedurally generated terrain and creatures milling about thereon. Nicholas continued to add to *Low Mem Sky* after the game jam ended too, with subsequent updates adding such things as new ship classes, space station docking, and other improvements.

Squeezing so many space sim elements – such as the galactic map – proved to be one of the big challenges of *Low Mem Sky*.

With work on *Low Mem Sky* now complete ("I'd reached the capacity of a single PICO-8 cart ages ago, and was battling to squeeze that last release into it," he says of the game's aptly named L.A.S.T. update), he's moved onto another, similarly ambitious demake.

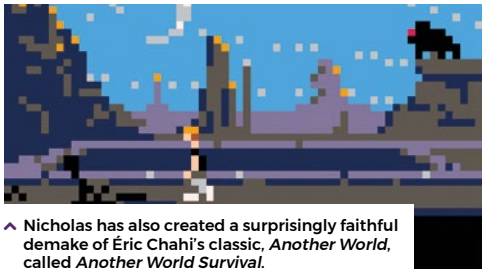
"This time I'm trying to recreate the RTS classic *Dune II* in PICO-8," Nicholas reveals. "It's currently titled *Pico-Dune*, but I'll let my Patreon community decide the final title. I'm very excited about it, as I loved this game back in the day. I don't yet know if it's really possible to squeeze it all into PICO-8 – but I think it's gonna be fun to find out!"

SQUEEZING PICO-8

"There are several tricks that I tend to rely on, primarily to make the most of the limited available cart space," Nicholas says, when asked about the techniques he uses to squeeze the most out of PICO-8. "The first is using the minifying feature of picotool, created by Dan Sanderson. Depending on your code, this can help reduce your character count – one of PICO-8's program limitations. Unfortunately, this also results in the code being unreadable, but you can always include a link to the original source for the benefit of the rest of the community."

Nicholas also recommends using PX8, a data compression library written by PICO-8 creator Joseph White. "This allowed me to fit the graphics for about ten completely different screens and locations, spread across two carts."

For more tips on how to push PICO-8 beyond its limits, check out Nick Walton's *Toolbox* guide on page 40.



Nicholas has also created a surprisingly faithful demake of Éric Chahi's classic, *Another World*, called *Another World Survival*.

Here's Nicholas' latest work in progress: a PICO-8 demake of *Dune II*.

Extra Credit

STEVE MCNEIL
Steve contributes nothing to society, but occasionally assuages his guilt via charities.

"I hope you'll find out more about these great charities, and maybe even bung them some cash"

As a comedian who's drifted into the world of making silly things about games, the best description I've come up with when asked what my job is, is 'video game-playing idiot'. Obviously, this is not a job. It certainly isn't a job the planet needs anyone to do. Getting paid to play games whilst being stupid is such a waste of a life that it should arguably be an imprisonable offence but, until I can atone for my sins via state-sanctioned incarceration, I have to find other ways to try to contribute something to the planet, rather than just gobbling up valuable oxygen.

I first met the gang at War Child Gaming a couple of years ago, when myself, Dara and Ellie from Go 8 Bit (not Sam, he's *actually* useless) took part in a campaign around Armistice Day where gamers attempted pacifist (no-kill) runs of traditionally violent games. Like many of my age, I first became aware of War Child when they released the 1995 charity album, *Help*, featuring huge artists of the time such as Radiohead, Oasis, and Blur. The charity exists to help children in areas affected by conflict and, unsurprisingly, given the growing amount of content which exists online for gamers, they're keen to access this audience in the hopes of bringing their message to a new demographic and, of course, raise funds.

Live-streamed events are great for raising awareness but increasingly War Child, and other charities like them, focus on revenue generation not through donations but by convincing developers and publishers to offer up time-limited bundles of games/DLC, co-ordinated with their traditional campaigns. It seems gamers are more likely to help financially if, in doing so, they get a game; a truth so unsurprising that it's almost patronising pointing it out.

Of course, live streams are a useful way of promoting these deals, and earlier this year I was lucky enough to be involved in another,

Special Effect, a great charity that helps disabled gamers.

ESL's 'Battle of the Brands', which sought to raise money for SpecialEffect, a charity which adapts technology to allow gamers with physical disabilities to control games which might otherwise be inaccessible to them via traditional controllers. A third charity, GamesAid, also focuses on fundraising within the gaming sector, but as a way of funding more traditional charities that seek to support children in areas such as housing, education, and welfare.

I've not yet had the chance to work with GamesAid, despite the comedian Imran Yusuf, who I know, regularly organising comedy gigs to raise funds. He must hate me. But, and it pains me to say this, some things aren't about me. I hope in reading this you'll take time to find out more about these great charities, and maybe even bung them some cash. If you do, tell them Steve McNeil sent you. Every penny you give to these charities can help offset my selfishness. One day, I hope to be able to look at myself in the mirror without spitting. Unlikely, but I can dream.

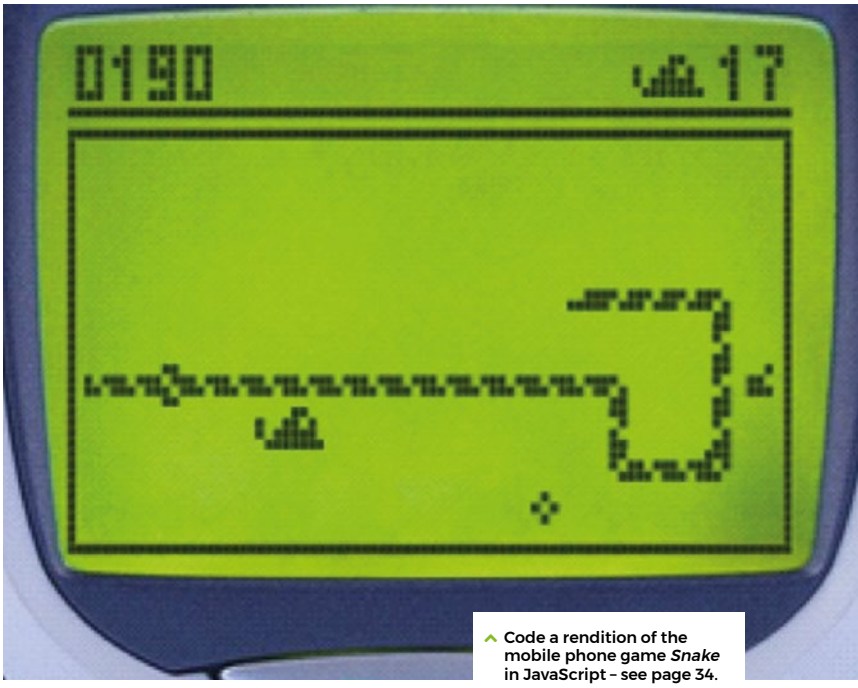
Donate to SpecialEffect here:
wfmag.cc/SEdonate

Get involved with War Child's Game On here:
wfmag.cc/WCGameOn

Donate to GamesAid here:
wfmag.cc/GAdonate

Toolbox

The art, theory, and production of video games



Code a rendition of the mobile phone game *Snake* in JavaScript – see page 34.

Find out more about the role of infrastructure in CityCraft on page 28.

CityCraft: Foundations of a video game city

An introduction to the role of infrastructure, and the fundamentals of urban engineering

AUTHOR
KONSTANTINOS DIMOPOULOS

Konstantinos Dimopoulos is a game urbanist and designer, currently working on the *Virtual Cities* atlas, and consulting on several games. **game-cities.com**

The crucial networks and facilities that have allowed civic life to exist are collectively described as urban infrastructure. These are the utilitarian roads, sewers, telephone lines, pipes, and bridges that are, not unexpectedly, often treated by games as a decorative afterthought. Despite this, landmarks like bridges, monorails, and well-designed sewers can actually define the character and functionality of a city; most of *Half-Life 2*'s City 17 was, after all, experienced through its sewers and canals.

Infrastructure keeps cities running, and considering such matters when designing digital cities is crucial if our aim is to create an immersive game environment. Of course, as infrastructure enables the core functions of a settlement, it's bound to evolve and change as societies, technologies, and life change around it. From today's complex telecom networks, to medieval wells, and from Roman roads to electrical substations, infrastructure is the skeleton on which any urban tissue – imaginary or real – must be based.

A Matter of Scale

Large infrastructure, such as bridges, international airports or harbours, besides offering interesting locations and gameplay opportunities, can also provide a sense of scale. We tend to find such imposing and expensive to construct structures in or near urban centres; sometimes, a glimpse of a great bridge in a game's background is enough to suggest the size and character of a city.

Managing transportation, power, and water networks is a crucial part of games such as *Cities: Skylines*.

COMMUNICATION AND TRANSPORT

Communication, and the transportation of people and goods, have always been central to urbanism; historians and geographers have argued that civic sizes and typologies are determined by the effectiveness and reach of communication and transportation networks. The modern suburbia, for example, couldn't exist without the motor car; sensible walking distances defined the limits of ancient Athens, and the audible range of a bell delineated many medieval districts. Interestingly, and as we humans have a tendency towards face-to-face interactions, communication and transportation networks overlap – or at least have done so for the greater part of history.

Allowing movement enables communication, and much of social life happened (and continues to happen) on roads and pavements – two of the oldest civic elements. The first permanent dirt roads were constructed over 5000 years ago, and the use of stones to pave them began during the Roman Empire. And whereas millennia-old plank roads have been discovered in southwest England, the widespread use of tarmac only began in the 20th century. Separating vehicle (or animal) traffic from pedestrians is another ancient practice, with the Roman pavement being the more famous example. Interestingly, the modern European pavement only became fashionable in urban planning during the 19th century.

Contemporary roads aimed at cars are at least three metres wide per lane, whereas a comfortable pavement should be at least two

The humble manhole can add believability and even character to virtual cities. This one is from real-world Oslo.

The bleak and atmospheric power station forms the backdrop for *S.T.A.L.K.E.R.: Shadow of Chernobyl*.

metres wide; a transportation system based on donkeys, camels, or horse-drawn carriages would obviously have to be designed to different specifications.

Then there's mass transport to consider. Horse-drawn trams and steam-powered underground trains moved urban populations during the 19th century, and eventually led to light rail systems, cable cars, and electrical metro networks. As for interurban connections, these can, according to setting, be handled by harbours, airports, long-distance trains, highways, and even teleportation gates.

Communication, for centuries based on town criers and messengers, has in the past 200 years vastly developed its specific infrastructure, and its wired and wireless networks are dense in today's urban spaces. The first mail delivery systems and pneumatic tube concepts gave way to miles of underground cables, mobile phone towers, telephone booths, and hundreds of satellites, as telecom networks grow in importance and efficiency by the day. It's difficult to imagine a contemporary city without a connection to the internet.

WATER, WASTE, POWER

Access to drinking water is, of course, vital for humans. The first settlements were thus located next to bodies of (relatively) clean drinking water, such as rivers and lakes. The technology of wells allowed those settlements to expand farther inland, and aqueducts transported fresh water over great distances, geographically liberating cities. Today, networks of underground pipes distribute water, and semi-private water towers still define the skyline of New York City. Before drinking water enters the supply pipes to individual properties, it passes through the larger mains pipes, but only after having been made safe to drink at a water treatment facility.

Ironically, it's water and erosion that are the great enemies of engineering. To preserve our built environments, we have to make sure that excess water is led out of our cities and into seas, lakes, or rivers. Sewers for managing floods, an idea as ancient as the Minoan civilisation, are often combined with networks that collect and dispose of waste that will eventually be processed in large sewage treatment plants. Whereas sewers of all types are commonly found underneath all of the city, sewage plants tend to be located far from residential areas due to their distinctly unpleasant odour.

"*Half-Life 2*'s City 17 was experienced via its sewers and canals"

As *SimCity* taught generations of players, water can not only be used to produce energy (the Three Gorges Dam hydroelectric facility in China can produce 18,200 megawatts, compared to the 4000 megawatt capacity of Chernobyl), but power is also required by all modern settlements. These vast pieces of infrastructure are intriguing from a game design perspective, and often need to be carefully protected in strategy games. The power from these complex networks moves machinery, and gives cities light, warmth, and energy. Wind farms, nuclear plants, gasworks, and fossil fuel power stations allow contemporary cities to exist, though in many cases, at a huge cost to our collective health and the environment.

Concept art from *Metro 2033* showing how a subway tunnel could be turned into an evocative, unique residential space.

Defensive Engineering

Depending on the type of world a game is set in, engineering, designing, and maintaining defensive infrastructure may be an important aspect of urban planning. Town walls, ramparts, castles, moats, along with barracks and gates, essentially defined the shape of medieval urbanism, whereas Moscow's metro was constructed to also function as a vast bunker in case of nuclear war. The space city in *Mass Effect*, the Citadel, is flanked by five arms – known as Wards – which can close up and transform it into an armoured cylinder.

The best game dev advice I ever received

The making of Batman Begins for EA taught Reid that there's no time for ego when your team's up against a tight deadline

AUTHOR
REID SCHNEIDER

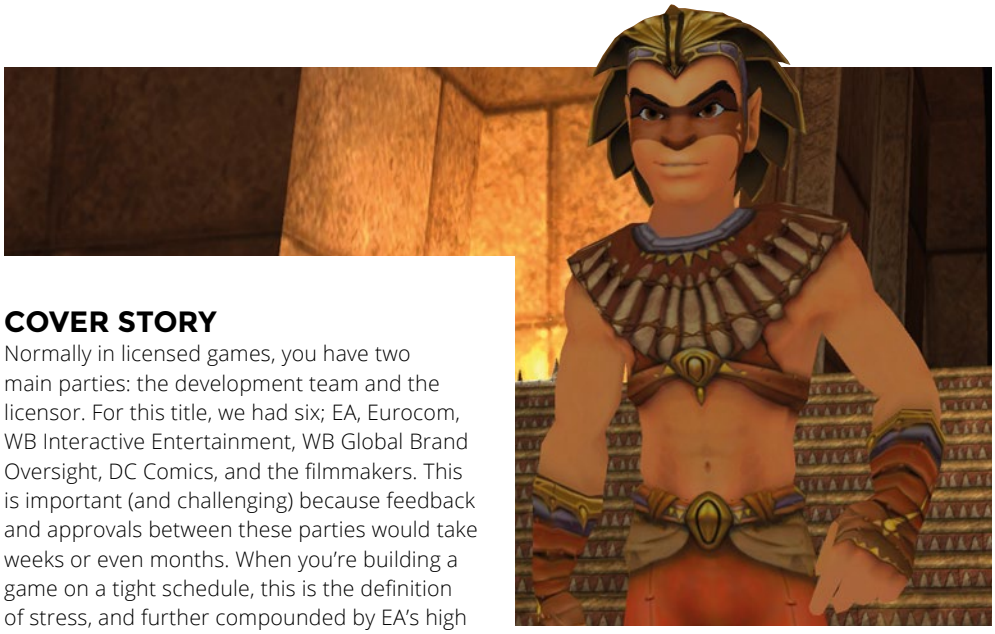
Reid is the producer of *Splinter Cell*, *Battlefield Vietnam*, *Army of Two*, *Batman: Arkham Origins*, and *Batman: Arkham Knight*. Follow him on Twitter: **@rws360**

In 2004, I was working for a division of EA called EA Partners, and had just finished working on *Battlefield Vietnam*. While I was closing that game, EA had obtained the *Batman* licence. That deal was, however, signed as a major console transition was coming, but there was still a tonne of life in the PS2. The goal, then, was to get a current-gen *Batman Begins* game made through an external developer, while an EA internal studio, Pandemic, would work on the next-gen version for the PS3 and Xbox 360 to coincide with the second film, *The Dark Knight*.

But there was a problem. By the time the contract was finalised, the time to make a game that launched alongside the movie was very short – roughly 12 to 14 months. With that in mind, we signed up Eurocom, a development studio known for its solid work on licensed games in condensed timeframes. The general manager of EA Partners asked me if I would help them get *Batman Begins* done, since I had some experience with Warner Brothers from my time at Ubisoft. I said yes, and before I knew it, we were on a 14-month schedule with no margin for error. In hindsight, I should have known this was a suicide mission with limited upside, but I was young, hopeful, and more than a bit naïve.

To get the game up and running, we repurposed much of the tech used for Eurocom's 2003 game, *Sphinx and the Cursed Mummy*. In fact, many of the original prototypes even had the Sphinx character running around the streets of Gotham. It looked pretty weird, but we made it work and got things going. As we ventured further into development, we realised that while the schedule was tight, we faced an even greater challenge on the licensing side.

Before Eurocom took over, DICE had briefly considered making *Batman Begins* using tech from their *Rally* series, but eventually passed after building a few prototypes.



Sphinx would slowly morph into the game that would become *Batman Begins*.

COVER STORY

Normally in licensed games, you have two main parties: the development team and the licensor. For this title, we had six; EA, Eurocom, WB Interactive Entertainment, WB Global Brand Oversight, DC Comics, and the filmmakers. This is important (and challenging) because feedback and approvals between these parties would take weeks or even months. When you're building a game on a tight schedule, this is the definition of stress, and further compounded by EA's high expectations for the title – the publisher had the goal of building a long-term franchise, since the *Lord of the Rings* games were drawing to an end.

In the second half of development, we began to do early press and previews for the game. The EA marketing and PR teams were given the mandate to go out and secure as many 'covers' for the game as possible. In the early 2000s – when there were still a sizeable number of gaming magazines – our PR manager was able to secure a cover feature with Official Xbox Magazine. EA, along with Eurocom, saw the potential of this opportunity, so we all jumped on it; the magazine was going to do a big spread along with developer interviews. Time was tight, so we needed to pull together a great preview version of the game, along with screenshots, and cover art for the magazine.

Making cover art for a magazine is a long process as it needs to serve many masters (developer, publisher, and so on), but most importantly, sell magazines. Given our tight timeframes, we chose to use an approved piece of art from the film's press kit. The magazine loved it, EA and Eurocom were on board – even WB Interactive and DC liked it, and we had the go-ahead. To say that we called this one wrong would be an understatement.

Though the image was from an approved press kit, and pretty much all the parties agreed to it, there was an executive in the Warner Brothers Brand Team who decided that it was definitely not OK to use. The executive then ripped into all of us like never before. (In fairness, we should have expected this to happen, as people close to the executive sometimes referred to her as "She who must not be named" – a reference to Voldemort from *Harry Potter*.)

We were bummed out, as we had all moved mountains to get this done. I took it particularly hard, as my goal was to support the team. Eventually, EA decided to go forward with the magazine story anyhow; we couldn't pull out at the last minute, as it would have destroyed our relationship with that publication and potentially others. And that's when my boss, Scot Bayless, hit me with some of the best advice I have ever received.

"Do you want to win, or do you want to be right?"

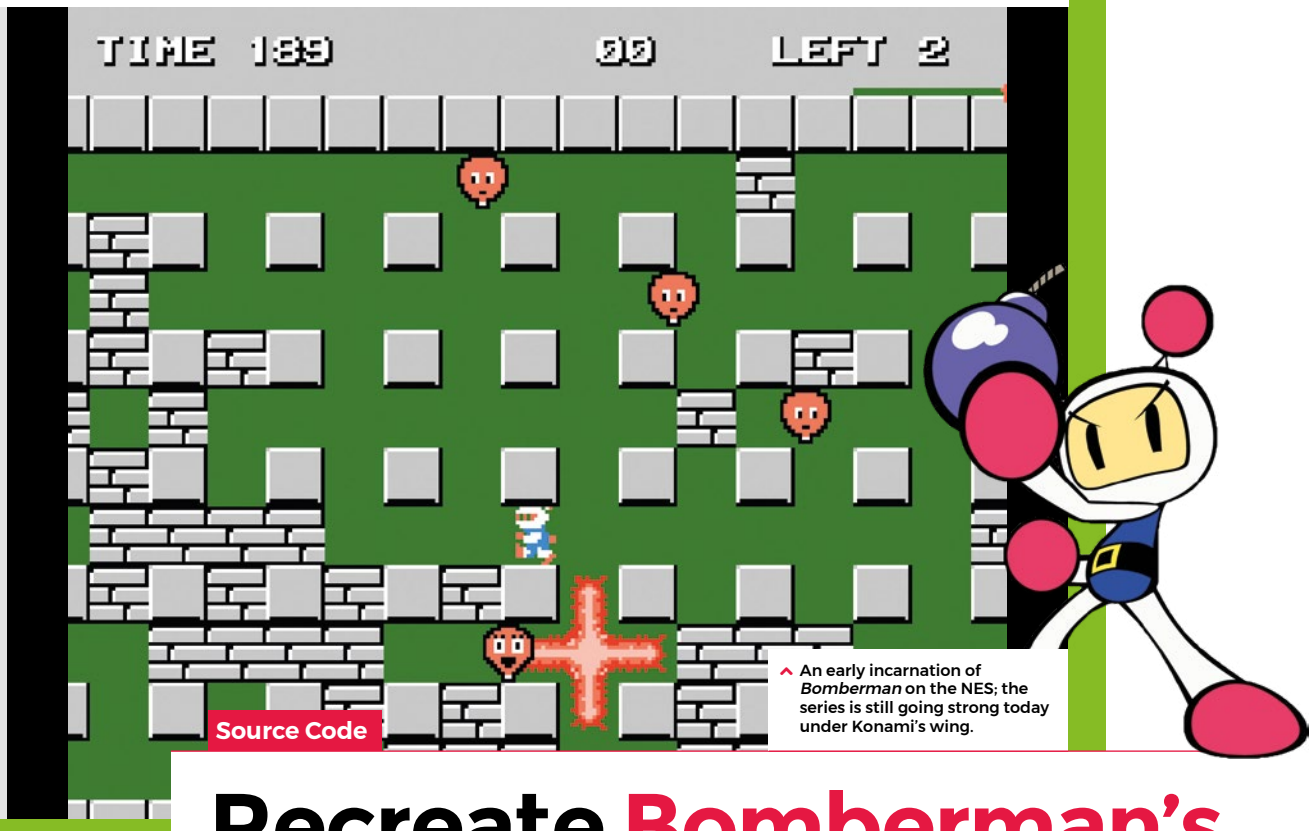
He could sense the sombre feeling in the room, and that it was affecting our work. He took me aside and said, "Do you want to win, or do you want to be right?". I realised, in that moment, that our win was making the best game we could in the time we had. We knew we'd done everything correctly, and were right in our convictions, but it didn't matter. We needed to refocus, get our heads back in the game, forget about being right, and just focus on the win.

Arguments happen all the time inside teams: tempers flare, and people can get cranky. When these things happen, I'll often ask the people involved, "Do you want to win, or do you want to be right? You generally can't have both."

When we refocus on what really matters – the game, obviously – and take our egos out of it, we can almost always see more clearly and get things done.

Knightfall

After the *Batman Begins* dust settled, EA announced that a game based on *The Dark Knight* was planned to release alongside the film. Our goal was to set that team – Pandemic – up for success, so we were happy about that. Unfortunately, Pandemic's team in Australia was hitting a wall in terms of the technology required for PS3 and Xbox 360, and the move from SD to HD. They had some prototypes running, but EA's belief in Pandemic as an effective studio was waning. The team eventually announced that they couldn't hit the *Dark Knight* movie launch date, or even the release of the DVD. Ultimately, EA shut the doors on Pandemic, and dropped the Batman licence to refocus on their internal IPs.



An early incarnation of *Bomberman* on the NES; the series is still going strong today under Konami's wing.

Source Code

Recreate Bomberman's iconic explosives

Learn how to recreate the exploding bombs found in the classic Bomberman games

AUTHOR
RIK CROSS

Bomberman was first released in the early 1980s as a tech demo for a BASIC compiler, but soon became a popular series that's still going today. *Bomberman* sees players use bombs to destroy enemies and uncover doors behind destructible tiles. In this article, I'll show you how to recreate the bombs that explode in four directions, destroying parts of the level as well as any players in their path!

The game level is a tilemap stored as a two-dimensional array. Each tile in the map is a `Tile` object, which contains the tile type, and corresponding image. For simplicity, a tile can be set to one of five types; `GROUND`, `WALL`, `BRICK`, `BOMB`, or `EXPLOSION`. In this example code, `BRICK` and `GROUND` can be exploded with bombs, but `WALL` cannot, but of course, this behaviour can be changed.

Each `Tile` object also has a timer, which is decremented each frame of the game. When a tile's timer reaches 0, an action is carried out, which is dependent on the tile type. `BOMB` tiles (and surrounding tiles) turn into `EXPLOSION` tiles after a short delay, and `EXPLOSION` tiles eventually turn back into `GROUND`. At the start of the game, the tilemap for the level is generated, in this case consisting of mostly `GROUND`, with some `WALL` and a couple of `BRICK` tiles. The player starts off in the top-left tile, and moves by using the arrow keys. Pressing the **SPACE** key will place a bomb in the player's current tile, which is achieved by setting the `Tile` at the player's position to `BOMB`. The tile's timer is also set to a small number, and once this timer is decremented to 0, the bomb tile and the tiles around it are set to `EXPLOSION`. The bomb explodes outwards in four directions, with a range determined by the `RANGE`, which in our code is 3. As the bomb explodes out to the right, for example, the tile to the right of the bomb is checked. If such a tile exists (i.e. the position isn't out of the level bounds) and can be exploded, then the tile's type is set to `EXPLOSION` and the next tile to the right is checked. If the explosion moves out of the level bounds, or hits a `WALL` tile, then the explosion will stop radiating in that direction. This process is then repeated for the other directions.

"I'll show you how to recreate the bombs that explode in four directions"

There's a nice trick for exploding the bomb without repeating the code four times, and it relies on the sine and cosine values for the four direction angles. The angles are 0° (up), 90° (right), 180° (down) and 270° (left). When exploding to the right (at an angle of 90°), sin(90) is 1 and cos(90) is 0, which corresponds to the offset direction on the x- and y-axis respectively. These values can be multiplied by the tile offset, to explode the bomb in all four directions.

Bomberman bombs in Python

Here's Rik's example code, which recreates *Bomberman*'s bombs and four-way explosions in Python. To get it running on your system, you'll first need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**


```
from math import cos, sin, radians

SIZE = 9
WIDTH  = SIZE*45 - 5
HEIGHT = SIZE*45 - 5

# bomb range
RANGE = 3

GROUND    = 0
WALL      = 1
BRICK     = 2
BOMB      = 3
EXPLOSION = 4
# images for tile types
images = ['ground','wall','brick','bomb','explosion']

player = Actor('player')
player.mapx = 0
player.mapy = 0

# each position in tilemap is a 'Tile' with type, image, timer
class Tile():
  def __init__(self, type):
    self.set(type)
  def set(self,type):
    self.timer = 0
    self.t=type
    self.i=images[type]

tilemap = [[Tile(WALL) if x%2==1 and y%2==1 else Tile(GROUND)
for y in range(10)] for x in range(10)]
tilemap[3][2].set(BRICK)
tilemap[4][7].set(BRICK)

def on_key_down():

  newx = player.mapx
  newy = player.mapy

  if keyboard.left and player.mapx > 0:
    newx -= 1
  elif keyboard.right and player.mapx < SIZE-1:
    newx += 1
  elif keyboard.up and player.mapy > 0:
    newy -= 1
  elif keyboard.down and player.mapy < SIZE-1:
    newy += 1

  if tilemap[newx][newy].t in [GROUND,EXPLOSION]:
    player.mapx = newx
    player.mapy = newy

  # space key to place bomb
  if keyboard.space:
    tilemap[player.mapx][player.mapy].set(BOMB)
    tilemap[player.mapx][player.mapy].timer = 150

def update():

  for x in range(SIZE):
    for y in range(SIZE):

      tile = tilemap[x][y]

      # decrement timer
      if tile.timer > 0:
        tile.timer -= 1

      # process tile types on timer finish
      if tile.timer <= 0:

        # explosions eventually become ground
        if tile.t == EXPLOSION:
          tile.set(GROUND)

        # bombs eventually create explosions
        if tile.t == BOMB:
          # bombs radiate out in all 4 directions
          for angle in range(0,360,90):
            cosa = int(cos(radians(angle)))
            sina = int(sin(radians(angle)))
            # RANGE determines bomb reach
            for ran in range(1,RANGE):
              xoffset = ran*cosa
              yoffset = ran*sina
              if x+xoffset in range(0,SIZE) and \
              y+yoffset in range(0,SIZE) and \
              tilemap[x+xoffset][y+yoffset].t in
[GROUND,BRICK]:
                tilemap[x+xoffset][y+yoffset].set(EXPLOSION)
                tilemap[x+xoffset][y+yoffset].timer = 50
              else:
                break

          # remove bomb
          tile.set(EXPLOSION)
          tile.timer = 50

def draw():
  for x in range(SIZE):
    for y in range(SIZE):
      screen.blit( tilemap[x][y].i,(x*45,y*45) )
  # draw the player
  screen.blit( player.image, (player.mapx*45,player.mapy*45)
```

Making Snake in JavaScript

Download the code from GitHub: **wfmag.cc/wfmag12**

Rhett shows you how to make a simple version of the classic mobile game in JavaScript

Rhett's version of *Snake* doesn't have an intro screen, but if it did, it'd look like this.

AUTHOR
RHETT THOMPSON

Rhett Thompson is an indie developer with an interest in game development tools. **keybored.app**

Snake is a classic game you might remember playing on the Nokia 3310, or in its more modern form of *Slither.io*. Either way, *Snake* is a masterpiece of simple yet challenging design: the aim is to move the snake around the screen, eating food while avoiding walls or its own growing body.

Before we jump into the game's code, however, we need to decide what engine to use. Unity and GameMaker are both overkill for a simple game like *Snake*; besides, coding the engine ourselves will provide an insight that can't be learned if we let someone else do all the work for us. That's why I'm using nothing but pure JavaScript (JS) here – this way, the game will run on any device with a browser.

The likeably chunky Nokia 3310, host to one of the most famous incarnations of *Snake*.

GETTING STARTED

Before any JS will run, we need to make a place for it to live. Let's make that home now by creating a new folder named **Snake**. It can be anywhere on your PC; just make sure you can get there easily. Inside that folder, make a new file called **index.html**. This is the webpage that will house our game.

Open this file and write the following code:

```
<!DOCTYPE html> 1
<html> 2
  <head>
    <meta charset="utf-8">
    <title>Snake</title>
    <link rel="stylesheet" type="text/css"
href=”css/snake.css”> 3
  </head>

  <body>
    <canvas id="bg"></canvas> 4

    <script src="js/snake.js"></script> 5
  </body>
</html>
```

1. This line lets the web browser know that this is an HTML file.
2. This `<html>` tag holds everything in the webpage.
3. This links to a style sheet (or instructions) for how to layout the webpage.
4. This canvas is the HTML element that will be used to draw the game's pixels.
5. This tells the web page to load the JavaScript file that contains the game code.

Next, we write a simple CSS file to style the webpage. It's best practice to make a subdirectory (or subfolder) to hold each type of file used for a website. Inside the **Snake** folder, make another folder called **css**. Inside the new **css** folder, make a new file called **snake.css**. Open **snake.css** and write the following code:

```
canvas { 1
  position: fixed; 2
```

```
  top: 0; 3
  left: 0;
}
```

1. This tells the following style code to apply to all 'canvas' elements found on the webpage.
2. This makes the canvas have a static position that won't change when the user scrolls the mouse wheel.
3. These two values set the position of the canvas to start at zero pixels away from the top of the webpage and zero pixels away from the leftmost corner of the webpage.

Finally, it's time for the JS. Before we write the code, we need a file to put it in. Make another new subfolder inside the **Snake** directory called **js**. Inside that, make a new file called **snake.js**. See **Figure 1** for a complete picture of the directory. Next, open the newly made **snake.js** and let's start by initialising the canvas for drawing pixels:

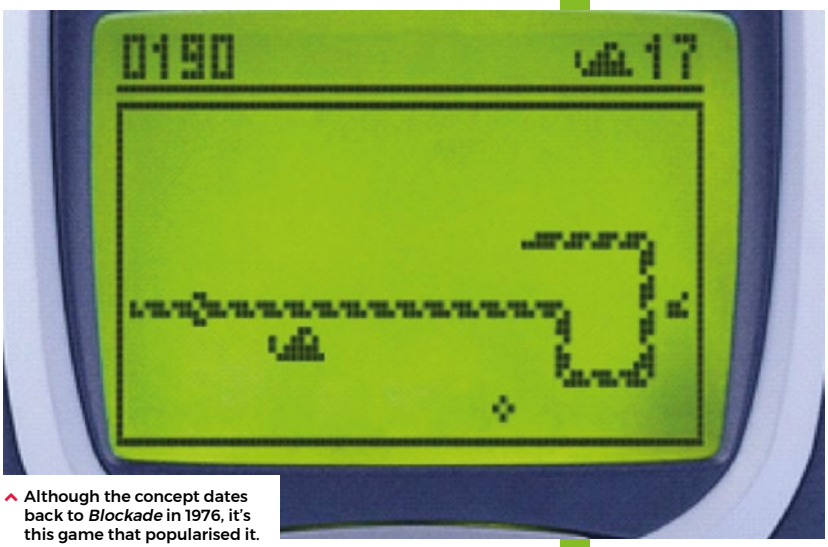
```
var canvas = document.getElementById('bg'); 1
var draw2D = canvas.getContext('2d'); 2

canvas.width = window.innerWidth; 3
canvas.height = window.innerHeight;
```

1. This grabs a reference to the HTML canvas via its ID.
2. Next, we initialise the canvas' context to be 2D so we can use it to draw pixels.
3. Resize the canvas so it fills the whole webpage.

The game needs to draw lots of pixels, so let's make a function to do this easily:

```
function drawBox(x, y, width, height, color)
{ 1
  draw2D.beginPath(); 2
  draw2D.rect(x, y, width, height); 3
  draw2D.fillStyle = color; 4
  draw2D.fill(); 5
  draw2D.closePath(); 6
}
```

1. The function takes the following parameters: X start position, Y start position, width, height, and colour.
2. To draw on the canvas, we define a path that will be filled in with a specified colour. This line opens the draw path.
3. Define a rectangle to be filled in. This will be the main part of the draw path.
4. The `fillStyle` is the colour of pixels that fill the draw path. It's a string that can be an RGB value with a number range of 0–255, a hexadecimal value like '#ffffff', or a colour keyword like 'white'.
5. Draw the path.
6. Close the path when you've finished drawing.



Although the concept dates back to *Blockade* in 1976, it's this game that popularised it.

Every video game requires a fair amount of maths to make it run smoothly, and *Snake* is no exception:

```
function roundToGrid(numb) { 1
  return Math.round(numb/10) * 10; 2
}

function randomRange(min, max){ 3
  return Math.random() * (max - min) + min;
}
```

1. We want everything in our game to fit on a nice grid, with each step being a 10px square.
2. This simple function keeps everything neat. See **Figure 2** for an explanation.
3. JS lacks a built-in `randomRange` function, but this is easily fixed. The function takes a `min` and `max` value as its parameters. It will return a random number. This number's range will be inclusive of `min` and exclusive of `max`.

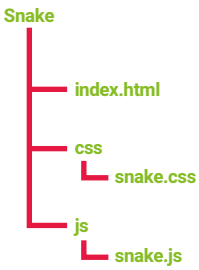
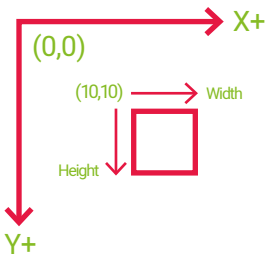


Figure 1: The complete file directory should look like this. At the top level is the Snake folder. Inside that is index.html and the css and js subfolders. Inside those are the relevant files.

87 ÷ 10 = 8.7
Round (8.7) = 9
9 x 10 = 90

Figure 2: The `roundToGrid` function takes any number (in this case 87), divides it by ten, rounds the answer, and finally multiplies that by ten.

Slither.io is a modern take on *Snake*. Players still eat food to grow, but they're on a huge map filled with other players.



In JavaScript, canvases draw pixels starting at the top left-hand corner of the screen with an inverted Y-axis. That means our `drawBox` function will start drawing a box with the XY positions as its top left-hand corner and extend from there the width and height.

Every game needs stuff to fill its world. Below, we define the variables that we'll use to run our game:

```
var snake = []; 1
snake.push( {x:canvas.width/2, y:canvas.
height/2} ); 2
snake.push( {x:-10, y:-10} ); 3
snake.push( {x:-10, y:-10} );

var xDir = 0; 4
var yDir = -1;

var food = {}; 5
food.x = randomRange(0, canvas.width-10); 6
food.y = randomRange(0, canvas.height-10);

var moveTimer = 0; 7
var moveTimerMax = 100;
```

1. An array to hold the snake's entire body.
2. We fill the array with `.push`, which adds an item to the end of the array. We add the snake's head first. We want it to start in the middle of the screen.
3. Push two tail segments to the array. These start off the screen, but we'll move them soon.
4. These two variables will be used to control what direction the snake moves.
5. Set up the food just like the tail segments – an object with X and Y properties.
6. Pick the food's position randomly between 0 and the canvas' width, but minus 10 so the food doesn't spawn off-screen.
7. `moveTimer` will count time. `moveTimerMax` will hold the length of the timer (100 milliseconds). These will control when the snake moves.

With our engine's math and draw functions complete, the next thing it needs is an update loop:

```
var lastFrameTime = Date.now(); 1

window.requestAnimationFrame(update); 2

function update() { 3
  var deltaTime =  Date.now() - lastFrameTime; 4
  lastFrameTime = Date.now(); 5

  //Movement timer here…

  render(); 6
  checkCollision();

  window.requestAnimationFrame(update); 7
}
```

1. To keep the game running on track, we need to store what time each frame starts in this variable. `Date.now()` gets the current time in milliseconds.
2. `requestAnimationFrame` requests that the browser call a function before the next repaint. The animation frame is a place to update the game before it's drawn.
3. Define the `update` function. This function will be called every frame of the game.
4. Delta time is the time between frames, or the amount of time that passed from the last frame to this frame. Delta time, mathematically, is the difference between the time when this frame happened and when the last frame did. This is where `lastFrameTime` comes in. (NB: `deltaTime` is in milliseconds here.)
5. Resets the `lastFrameTime` as this frame's start time so that it's ready for the next frame.
6. We'll write these functions soon.
7. At the end of every update frame, call the next frame recursively so the game loop never stops.

Next, we create a timer to control the Snake's movement:

```
moveTimer += deltaTime; 1
if (moveTimer > moveTimerMax) { 2
  moveTimer = 0; 3
```

```
  //Movement code here...
}
```

1. Every frame, add the **deltaTime** to the **moveTimer**.
2. If the timer has reached more than the **moveTimerMax** (100 milliseconds), it's time to move.
3. It's a good rule of thumb to reset the timer first so that it's ready to start counting again right away.

First, inside the movement timer, every tail segment moves:

```
for(var i=snake.length-1; i > 0; i--) { 1
  snake[i].x = snake[i-1].x; 2
  snake[i].y = snake[i-1].y;
}
```

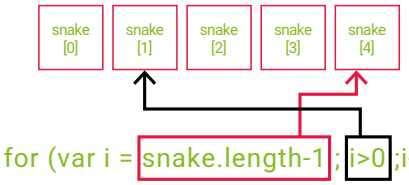
1. This is the really cool part of *Snake*. It's the main animation loop where we generate the effect of the tail following the head. Every time the snake moves, each tail segment takes the place of the previous one. This is accomplished via a reverse **for** loop. Instead of starting at index 0 and going until the end, the loop is starting at the last index and stopping at index 1. Don't update index 0 here because it's the head. It'll be updated later separately.
2. Each tail segment is given the X and Y position of the index before it.

Following that, we move the head (index 0):

```
snake[0].x += xDir*10; 1

snake[0].y += yDir*10; 2
```

1. To move the head, we add 10 pixels (because that's our grid size) multiplied by the current **xDir**. This variable will always be 1,-1, or 0. **xDir** is 1 when moving right, -1 when moving left, and 0 when moving up or down.
2. **yDir** is 1 when moving down (the Y-axis is flipped), -1 when moving up, and 0 when moving left or right.

If you run the game now, all you'll see is a blank screen. That's because we have outlined what data makes up the game, but now need to render a visual representation of it:



The loop starts at `snake.length-1` (the last index) and continues all the way to `index 1` (which means every index greater than zero).

```
function render() { 1
  drawBox(0,0, canvas.width, canvas.height,
'#000000'); 2

  food.x = roundToGrid(food.x); 3
  food.y = roundToGrid(food.y);
  drawBox(food.x, food.y, 10, 10, '#ffffff');
4

  for(var i=0; i < snake.length; i++) { 5
    snake[i].x = roundToGrid(snake[i].x); 6
    snake[i].y = roundToGrid(snake[i].y);
    drawBox(snake[i].x, snake[i].y, 10, 10,
'#ffffff'); 7
  }
}
```

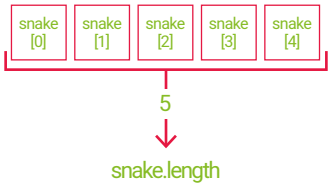
1. **render** will be called each frame, so what's drawn on the screen will reflect the variables behind the game.
2. First, clear the last frame so we can start drawing on a clean slate. Do this by drawing one big black box that starts at the canvas' top left-hand corner (0,0) and go its full width and height. **#000000** is a hexadecimal colour value for black.
3. Round the food's **XY** position to the grid before it's drawn.
4. Draw the food at its **XY** position and make its width and height 10 pixels. **#ffffff** is a hexadecimal colour value for white.
5. Loop through the whole snake, including its head.
6. Round each snake-segment's **XY** position to the grid before it is drawn.
7. Draw the snake segment.

When a game ends, the code needs to pick itself back up again – which is where the **restart** function reinitialises all the variables:

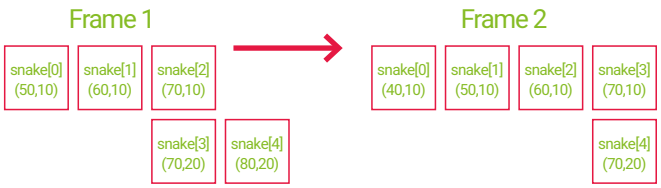
```
function restart()  {
  snake = []; 1
  snake.push( {x:canvas.width/2, y:canvas.
height/2} ); 2
  snake.push( {x:-10, y:-10} );
```

QUICK TIP

You can use `1/(deltaTime/1000)` to get a rough fps count. This works in any game engine, including Unity. (Just be sure `deltaTime` is in milliseconds – in Unity, `deltaTime` is measured in seconds, so the correct fps code would be: `1/Time.deltaTime`)



`.length` returns a count of how many items are in an array. This means, even though all arrays start at index 0, `snake.length` will count all indices (0,1,2,3,4) as five items.



Each time the snake moves, its tail is updated starting at the last segment, which is then moved to the position of the segment before it.

```
  snake.push( {x:-10, y:-10} );

  xDir = 0; 3
  yDir = -1;
  moveTimer = 0; 4

  food.x = randomRange(0, canvas.width-10); 5
  food.y = randomRange(0, canvas.height-10);
}
```

1. Empty the snake's body-array.
2. Refill the array with the default: head and two tail segments.
3. Reset the snake movement direction to up.
4. Reset the movement timer.
5. Respawn the food in a random position.

COLLISION DETECTION

Next up is the **checkCollision** function. This one's a bit long, so note that the next three blocks of code need to be written in the marked area below:

```
function checkCollision() {
  //Code goes here…
}
```

This code checks the snake-tail's collision:

```
for(var i=1; i < snake.length; i++) { 1
  if(snake[0].x == snake[i].x && snake[0].y ==
snake[i].y){ 2
    restart(); 3
    break; 4
  }

  if(food.x == snake[i].x && food.y ==
snake[i].y){ 5
    food.x = randomRange(0, canvas.width-10);
    food.y = randomRange(0, canvas.height-10);
  }
}
```

1. Start after the head (index 0), and loop through every tail segment.
2. 2D collision is tested by checking if the pixels of two objects overlap. Programmatically, this means checking if their X and Y positions are the same. Thus, the snake-head's position is checked against every tail segment's.
3. If the snake's head is touching part of the tail, it's game over! Time to restart!
4. The game will throw an error if the **snake** array is reset in the **restart** function, but we keep checking its indices as if it were as long as ever. To fix this, use a **break** to exit the loop.
5. This checks that food isn't spawned inside the snake's tail. If it is, then respawn it somewhere else!

The next collision check looks at whether the snake's hit a wall:

```
if (snake[0].x < 0 || snake[0].y < 0){ 1
  restart();
}

if (snake[0].x >= canvas.width || snake[0].y
>= canvas.height){ 2
  restart();
}
```

1. If the snake-head's X position is less than zero, it hit the left wall of the screen. If the snake-head's Y position is less than zero, it hit the top wall.
2. If the snake-head's X position is greater than or equal to the canvas width, it hit the right wall. If the snake-head's Y position is greater than or equal to the canvas height, it hit the bottom wall.

The final part of the collision check is for the snake eating food:

```
if(snake[0].x == food.x && snake[0].y ==
food.y) { 1
  snake.push( {x:-10, y:-10} ); 2

  food.x = randomRange(0, canvas.width-10); 3
  food.y = randomRange(0, canvas.height-10);
}
```

1. If the snake's head is colliding with food, it needs to be eaten.
2. Since the snake just ate some food, it has room to grow. Add another segment to its tail.

NOTE

Remember what I said about checking collision? There's collision anytime the pixels of two objects overlap. Why does our collision check only test if the XY positions are the same? Simple: if everything's rounded to squares on a grid, then if two objects are in the same square, they must be touching.

3. Move the food to a new position.

Next, we need to give players control of the game. JS handles this via **`EventListeners`**:

```
window.addEventListener('keydown', function
(evt) { 1

  if(evt.key == 'ArrowUp' && yDir != 1) { 2
    yDir = -1; 3
    xDir = 0;
  }
  if(evt.key == 'ArrowDown' && yDir != -1){
    yDir = 1;
    xDir = 0;
        }
  if(evt.key == 'ArrowRight' && xDir != -1){ 4
    xDir = 1;
    yDir= 0;
  }
  if(evt.key == 'ArrowLeft' && xDir != 1){
    xDir = -1;
    yDir= 0;
  }
}, false); 5
```

1. **`window.addEventListener`** adds a listener (which is a checker) to the window object. Generally, the **`window`** object is a representation of everything in the current browser tab. The **`'keydown'`** string specifies which event the listener is checking for. In this case, the listener is triggered whenever a key is pressed. Finally, **`function(evt)`** is a parameter that's called every time the listener is triggered. **`evt`** is an object that holds all data related to the event that triggered the listener.

2. This **`if`** statement first checks what key press triggered the listener. If it was the 'up' arrow, then it does the next check to make sure the snake is not currently moving down. The snake can't be allowed to turn on top of itself or it'll trigger a 'game over'. Remember, the Y-axis is flipped, so a positive **`yDir`** means the snake is moving down the screen.

3. Change the Snake's **`yDir`** to be moving upward (Y-axis is flipped). Since the snake can only move in straight lines, the **`xDir`** needs to be set to zero. This way, there's no change in the X position when the snake's moving vertically.

4. The same for horizontal movement: if the snake's heading right, it can't immediately move left. Mirroring vertical movement, the **`yDir`** is set to zero so that the snake only moves in straight lines.

5. **`false`** is the last parameter of **`window.addEventListener`**. Its purpose is a bit out of the scope of this tutorial, but, in a nutshell, it affects the order of when event listeners are called.

The developer console is a good tool for debugging code. To open, use key combination **Ctrl+Shift+K** in Firefox, or **Ctrl+Shift+J** in Chrome.

```
window.addEventListener('resize', function()
{ 1
  canvas.width = window.innerWidth; 2
  canvas.height = window.innerHeight;

  restart(); 3
}, false);
```

1. This event listener checks for when the window is resized.

2. If the window has changed size, the canvas needs to reflect that change. If it doesn't, the game might continue to render part of the game outside the viewport of the player.

3. If the game's canvas is resized, the game restarts. This guarantees that both snake and food are on the screen.

And that's it – you should now have a fully functional game of *Snake* that'll run on any device that supports JavaScript.

To run the game, open **index.html** in a browser. If you want to take the game further, try making a score UI – start by investigating **`draw2D.fillText(text, x, y)`**. Until then, take a moment to enjoy the fruits of your labour – open the game and have fun.

Now that *Snake* is complete, see if you can beat it. We recommend playing in a small window.

PICO-8: beyond the limits

Download the code from GitHub: **wfmag.cc/wfmag12**

Break PICO-8's strict limits to create games much larger than you thought were possible

AUTHOR
NICK WALTON

Gidday, I'm Nick Walton, an indie game developer from New Zealand. I'm making *Notemon*, a monster-taming RPG in PICO-8.
CroakingKero.com

The outer limits

PICO-8's other limits include the 256 sprites, map size of 128×64, 64 SFX, and 64 music patterns. Can you figure out how to break these limits too? Perhaps a procedurally generated map that never hits an edge, or several minutes of music loaded in from other cartridges.

A PICO-8 cartridge which shows how bits work in a 15-bit integer.

One of the core features of PICO-8 is the list of strict limitations on various resources. These limits can cure decision paralysis and enable us to come up with creative ideas. Sometimes, though, an idea outgrows these limits, so I'm here to show you some ways to subvert PICO-8's bounds, and produce games far larger than you may have thought possible.

BREAKING THE SAVE LIMIT

The most important limit many people don't consider is save data. The strict limit of 64 values can turn what could have been a sizeable, progression-based game into a short arcade experience. Blowing through this limit allows you to save a character's level progress, inventory, stats, and a myriad other possibilities.

The core of this limit break is using the 64 discrete 32-bit 'persistent cart data' values as a continuous array of 2048 bits, which can be arbitrarily divided into variably sized values. First, we need to be able to load bytes from cart data to the bit array and save the bit array into the cart data per-byte. Before you can access the save data, you must call the function: **`cartdata("some_name")`**. Cart data is stored in memory at 0x5e00-0x5eff (256 bytes).

```
function load_data()
 for i = 0, 255 do
  local byte = peek(0x5e00+i)
  for j = 8, 1, -1 do
   bits[i*8+j] = band(byte, 1)
   byte = shr(byte, 1)
  end
 end
end
```

When loading, we extract each byte with **`peek()`**, use the binary **`AND`** function to extract the value of just the lowest bit, then shift the bits right to move the next bit to the lowest spot. Note that we're extracting the rightmost bit each time, so we save it from right to left in the bit array, thus the inner loop goes from 8 to 1 instead of 1 to 8.

```
function save_data()
 for i = 0, 255 do
  local byte = 0
  for j = 1, 8 do
   byte = shl(byte, 1)
   byte = bor(byte, bits[i*8+j])
  end
  poke(0x5e00+i,byte)
 end
end
```

When saving, we start with an empty byte and use the binary **`OR`** function to set the byte's lowest bit to the value of the current bit in the array. Over the remaining loop iterations, this bit gets shifted left to the correct position in the byte. As we shift the bits left, the first bit placed in the byte becomes the highest in the byte. Easy! Now we can do something useful with our array of bits and save/load an integer value.

```
function int_to_bits(int, addr, num_bits)
 for i = 1, num_bits do
```

```
  bits[addr+num_bits-i] = b(int, 1)
  int = shr(int, 1)
 end
end
```

This is very similar to our **load_data()** function on the previous page. When converting an integer to bits, we use the binary **AND** function, as before, to extract the bits one at a time into the bit array. The **num_bits** should be the number of bits required to represent the range of values of your int. If you're saving a character's level, which can range from 1–100, that's a range of 99, so you'll need 7 bits: 2^7 = 128.

```
function bits_to_int(addr, num_bits)
 local int = 0
 for i = 0, num_bits-1 do
  int = shl(int, 1)
  int += bits[addr+i]
 end
 return int
end
```

When converting the bits back into an integer, we load each bit into the rightmost bit of the integer. This bit gets shifted left over the remaining iterations of the loop to its correct position in the integer. With just the bits and integer conversion, you can achieve a lot. For instance, you could have a table where each letter in the alphabet, plus some extra characters, are represented by an integer from 0–31. Then you could use five bits per letter to save the names of party members. I'll leave it up to your creativity to figure out other interesting ways to use these 2048 bits.

BREAKING THE CODE LIMIT

When creating my current big game project, *Notemon*, I quickly hit the 8192 code token limit. After much deliberation over whether to cut features or try to break the limit, I decided that this monster-taming RPG simply requires certain features that can't fit into that many tokens, like dozens of unique moves, boss fights, NPCs, and so on. Code isn't stored in memory that is accessible from within PICO-8 itself, so, unfortunately, you can't **poke()** new code to be executed. Instead, code must be stored in entirely different cartridges, which you load into automatically during gameplay. For example, *Notemon* loads from the World cartridge to the Battle cartridge for a random encounter, and to the Barn cartridge when you're looking after your friends. Here's how you can do it:

cartridge_1:

```
function _update60()
 if(btnp(❎)) load("cartridge_2")
end

function _draw()
 ?"1 "
end
```

cartridge_2:

```
function _update60()
 if(btnp(🅾️)) load("cartridge_1")
end

function _draw()
 ?"2 "
end
```

To type the ❎ and 🅾️ symbols in PICO-8, press **SHIFT+X** or **SHIFT+Z** respectively. You can also produce the directional arrows with **SHIFT+** any of **R**,**L**,**U**,**D** (right, left, up, down). These symbols work with the **btn()** and **btnp()** functions so you don't have to remember the number code for each button.

Run either cartridge and it'll incessantly tell you which cartridge number it is. Press the corresponding key (**X** for 1, **Z** for 2) and it'll load into the opposite cartridge and continue execution.

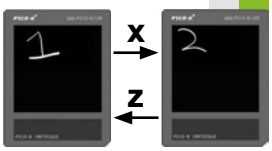
It seems deceptively simple, but keeping gameplay coherent between cartridges is difficult. The reason I showed you how to break the save limit first is because it's a great way to store important data when switching between cartridges. Use the same **cartdata()** name in each cartridge and load/save at the right addresses, and you'll be able to create more interesting and detailed gameplay than previously thought possible in PICO-8.

Here's a final tip: multiple cartridges can be bundled together in PICO-8's exported binary and web formats – up to 16 altogether. Type 'export my_game.html cart_1.p8 cart_2.p8'. I hope you've been inspired to tap away in PICO-8 and create something cool!

A visualisation of the loops we use to extract each bit from a byte into the bit array.

A brief note on integers

PICO-8's limit for integers is 32767, which takes 15 bits, so be sure not to attempt to convert more than 15 bits at once when using this method.



This cartridge swapping technique is similar to a 'finite state machine', only on a grander scale.

Nick Walton's *Notemon*, a monster-catching RPG so large that it's spread over multiple PICO-8 cartridges.

Directory

PICO-8: beginner's resources

If you're keen to delve deeper into PICO-8, here are some useful tutorials and other links

The official PICO-8 manual

An in-depth tour of PICO-8's innermost workings, written by creator Joseph White.

wfmag.cc/pico-man

PICO-8 cheat sheet

Perfect for printing out and keeping close to hand, this is a handy reference guide to PICO-8's commands and functions.

wfmag.cc/pico-cheat

Arnaud Debock's PICO-8 zines

A collection of zines that provide a wealth of tutorials and sample programs for beginners and more advanced users.

wfmag.cc/pico-zine

Game dev with PICO-8 zine

Dylan Bennett's 72-page zine is another valuable resource for PICO-8 users, with tutorials on game development and even itch.io publishing.

wfmag.cc/pico-dev

Animating a simple character tutorial

This beginner's tutorial takes you through the basics of creating an animated sprite.

wfmag.cc/pico-anim

Create an open-world adventure

Created by our own Rik Cross, this series of YouTube videos shows you how to create a dinky open-world game from scratch.

wfmag.cc/pico-open

Sound and music tutorials

Matt Tuttle's short series of videos provide some useful tricks and techniques for getting the best music and sound from PICO-8.

wfmag.cc/pico-tunes

PICO-8 wiki

Fandom's PICO-8 wiki offers a variety of helpful stuff, from quick-start guides to tutorials and a glossary of commands.

wfmag.cc/pico-wiki

GET INVOLVED

Do you have an online tutorial you'd like to share with readers? Have you created an online resource that other game developers might find useful? Maybe you have a local code club you're keen to promote? If you have something you'd like to see featured in the Directory, get in touch with us at **wfmag.cc/hello**

This PICO-8 cheat sheet provides an invaluable at-a-glance guide to commands, colours, and controls.

Wireframe

LIFTING THE LID ON VIDEO GAMES

Download the app

Out now for smartphones & tablets

Save 45% with an annual subscription

£1.99 rolling subscription

or

£34.99 subscribe for a year

Available on the App Store

GET IT ON Google Play

MODERN GAMES RETRO CONSTRAINTS

Pixel art is more popular than ever, but is our nostalgia only skin deep? What does it really mean to make an 8-bit or 16-bit game today?

WRITTEN BY
ALAN WEN

Thierry Boulanger, co-founder of Sabotage Studio, and creative director of *The Messenger*.

Video games have always been driven by the latest technological trends, with big-budget blockbusters frequently pushing the limits of graphical fidelity and processing power. In more recent years, however, we've become more appreciative of previous generations' achievements. If anything, both industry and consumers are now more enamoured with the past than ever, from miniature versions of classic consoles to the widespread use of pixel art in the indie sphere. Indeed, some of the most acclaimed and brilliant games of recent years have featured a visual style harking back to the past, from *Celeste* to *Undertale*. You certainly wouldn't expect anything different from *Starbound* and *Wargroove* publisher, Chucklefish.

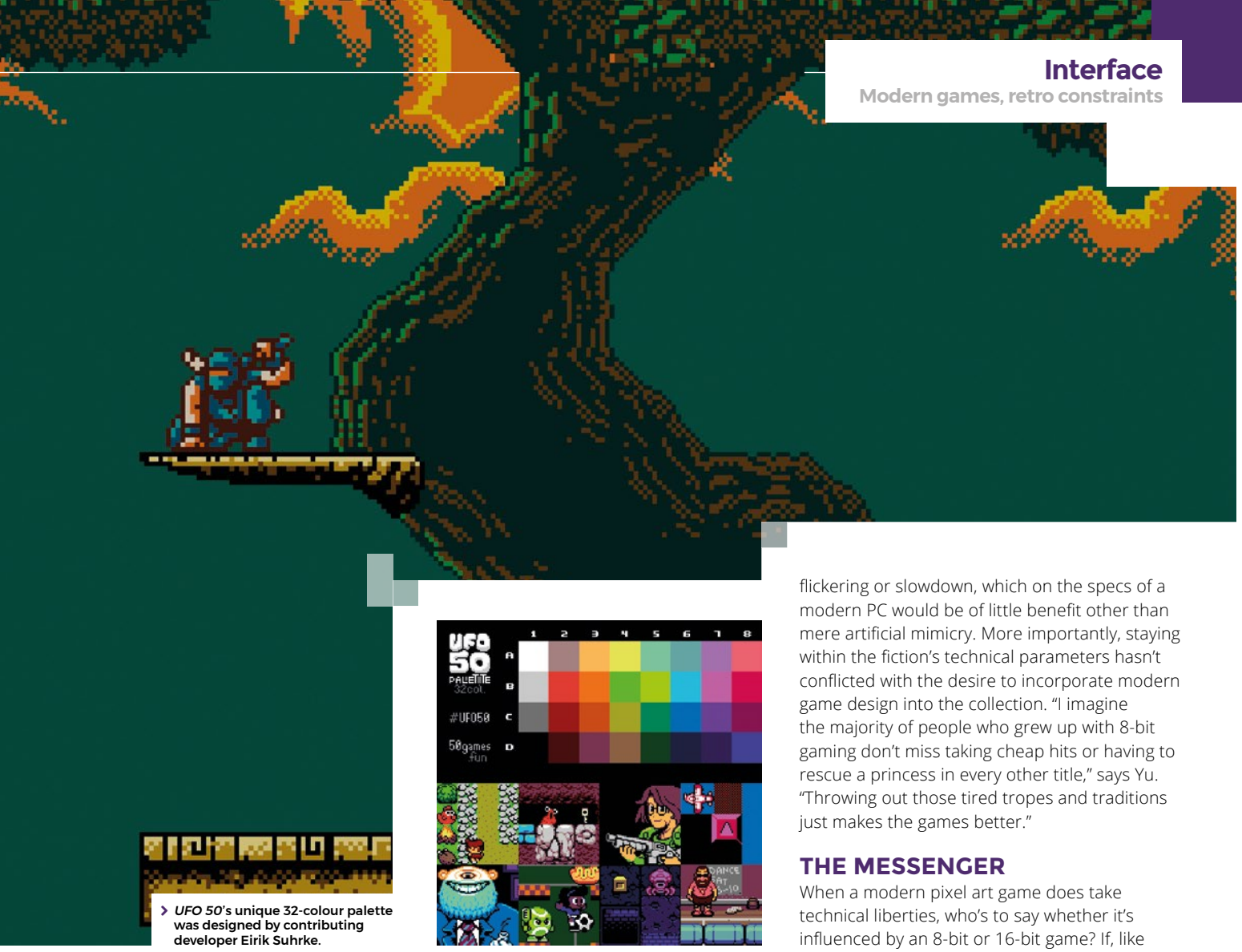
While pixel art is merely an aesthetic, it's a style that transports the player back to the 8-bit and 16-bit eras of the 1980s and 1990s. But when developers aren't bound by the same hardware restrictions as they were 20 or more years ago, is there more to the use of pixel art than mere rose-tinted nostalgia? Surely these games are just a case of applying a retro aesthetic to a modern game, with complex AI and physics that couldn't have been achieved on an ageing piece of hardware like the Super Nintendo?

There are some developers, though, who are capable of coming up with modern ideas and sensibilities within the parameters of what a genuine 8-bit or 16-bit game would allow. In our first case, it's about creating a whole fictional 8-bit *platform.*

UFO 50

UFO 50 is a forthcoming collection of 50 games made by a collective of developers, among them Derek Yu of *Spelunky* fame. These aren't mini-games nor *WarioWare*-style 'microgames', either, but complete experiences spanning a wealth of genres and ideas. What ties them together is a backstory in which each title was created by a progressive company in the 1980s before it tumbled into obscurity. The story provides the parameters within which each developer could create games with an authentic 8-bit look.

For many gamers and developers, particularly in the US, the 8-bit era is immediately synonymous with the Nintendo Entertainment System. In reality, hardware specifications varied wildly between such systems as the Commodore 64, ZX Spectrum, and Sega Master System (even the Atari 2600 was 8-bit, though not graphically). It's something Yu acknowledges, hence the team's decision to look at other hardware besides the NES. "The vibe we were going for is slightly off the beaten path," he says.



› ***UFO 50*'s unique 32-colour palette was designed by contributing developer Eirik Suhrke.**

"So the MSX was a big influence on us – we liked that it was a computer standard rather than a dedicated games console. But we did also research consoles like the Dendy, a Taiwanese Famicom clone that ended up being a huge hit in Russia."

In *UFO 50*, its games all adhere to a 32-colour palette, while sprites are only allowed four colours (including black). Animations are also restricted to around four frames – something which Yu says took some time to get used to. Then again, these parameters were less about restriction and more about balancing authenticity with speed of creation. "It's a lot easier adding characters when you can get away with only two or three frames for an animation," Yu says, "We went for limitations that would get us the most 'bang for the buck', rather than being so strict that you're forced to constantly check whether something is right or not, which can slow development down."

Yu and his collective also made a conscious decision not to be too slavishly authentic to the technical limitations of old hardware, like sprite-flickering or slowdown, which on the specs of a modern PC would be of little benefit other than mere artificial mimicry. More importantly, staying within the fiction's technical parameters hasn't conflicted with the desire to incorporate modern game design into the collection. "I imagine the majority of people who grew up with 8-bit gaming don't miss taking cheap hits or having to rescue a princess in every other title," says Yu. "Throwing out those tired tropes and traditions just makes the games better."

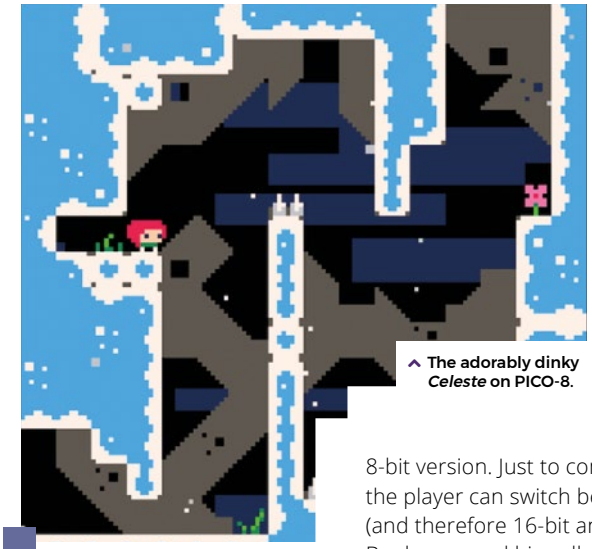
"Throwing out those tired tropes and traditions just makes the games better"

THE MESSENGER

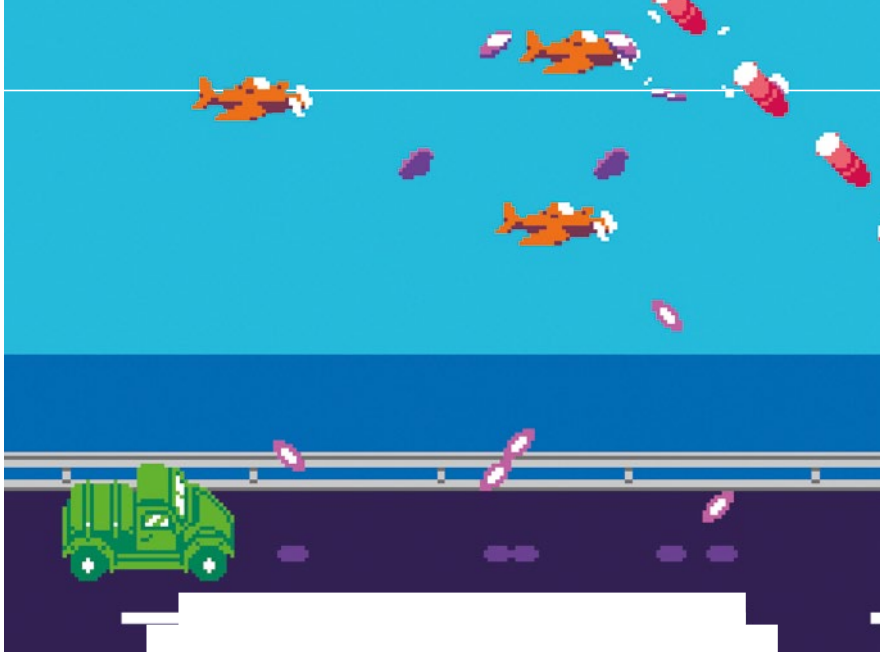
When a modern pixel art game does take technical liberties, who's to say whether it's influenced by an 8-bit or 16-bit game? If, like developer Sabotage Studio, you're making a game that incorporates the visuals of both 8-bit and 16-bit eras, then that distinction needs to be crystal clear.

The Messenger began as an 8-bit throwback to director Thierry Boulanger's favourite game, *Ninja Gaiden II*, on the NES, albeit with its own mechanical twist: cloudstepping, which allows your character to perform another jump in the air after slashing an object. *The Messenger*'s time travel narrative also inspired the decision to depict the future in a technically advanced 16-bit style, allowing for more colours and detail, and double the animation frames of the ➔

^ **Controls in both 8-bit and 16-bit worlds are the same, but the latter animates more smoothly, with double the frames.**



The adorably dinky *Celeste* on PICO-8.



FANTASY CODE

While it's easy to break your own rules of making a retro game, there are easier ways to work within defined limitations without learning to code on the Mega Drive from scratch. Occupying this niche is the 'fantasy console', such as the PICO-8 – a 'virtual console' you'll find plenty more coverage of in this very issue. Its specs are actually much more harshly limiting than most genuine 8-bit hardware, but it's nonetheless proven to be a fertile ground for creative hobbyists – including for prototypes of full-fledged games like *Celeste*.

8-bit version. Just to complicate things further, the player can switch between these two eras (and therefore 16-bit and 8-bit styles), requiring Boulanger and his collaborators to draw every asset twice.

"It was quite a big move to double all the assets, because every single area is made in both renditions," says Boulanger. "At first it wasn't going to be all of them, but then we designed accordingly, because we knew players expected to see the sprites in both versions and to hear tracks in both versions."

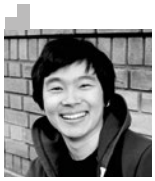
"You can add more colours than palettes supported back then"

What's perhaps surprising is that while the 8-bit version is clearly influenced by the NES, the 16-bit style is actually based on the Mega Drive, notably the soundtrack from composer Rainbowdragoneyes, which was created using the Deflemask tracker's Yamaha YM2612 Mega Drive sampler. This makes *The Messenger* not only a time-travelling adventure, but a cross-platform one.

Although Boulanger says he plays *Ninja Gaiden II* on the NES every two weeks, he wasn't looking to mimic the game's every aspect in *The Messenger*. "Everything to do with the controls was about removing friction," he says, "So things that are different from *Ninja Gaiden*: say you're running and you attack; your character stops running to attack while standing, which hinders your movement. The game becomes about the timing and precision, and not feeling the weight of having to stop to attack."

It might be better, then, to consider the 8-bit iteration of *The Messenger* as more of a super-optimised NES game than one that reflects it accurately. But even with the graphical differences between the two, it was ultimately important for elements like hitboxes and mechanics to remain consistent, even if, say, the number of abilities you acquire in later areas are more complex than you could realistically input on a two-button NES controller.

"That was an idea we had, where going into the future you'd be the master ninja and you'd have all your skills, and then when you're stuck in 8-bit you have to find a portal to access this path with the skills," explains Boulanger. "But as we had real-time seamless transitions and you'd switch very often, sometimes mid-jump or in the middle of a boss fight, it became clear that it would be unfair to switch the controls on you."



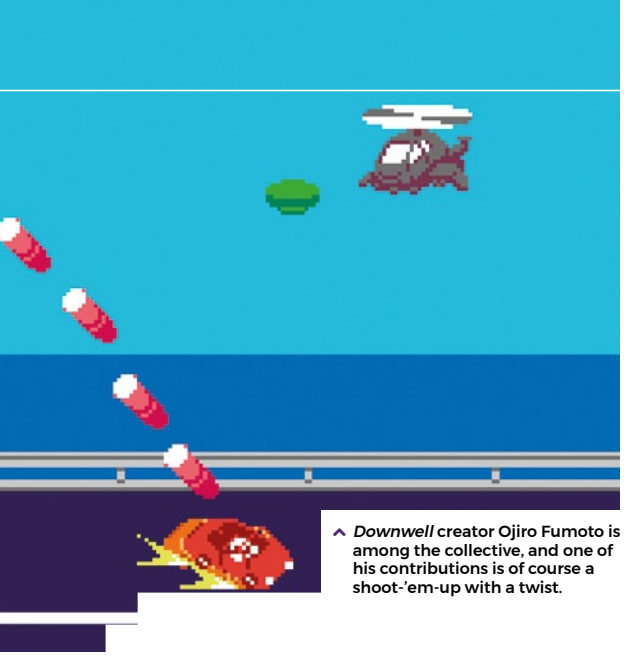
Derek Yu is one of six developers working on *UFO 50*, which he's also publishing via his Mossmouth studio.

For those who don't still have a Mega Drive kicking about, *Tanglewood*'s available on Steam, too. Other planned ports include homebrew favourite, the Dreamcast.

TANGLEWOOD

Of course, by playing around with two console generation styles in one game, *The Messenger*'s time-bending concept unsurprisingly bends the rules on technical limitations. When you consider that it and *UFO 50* are games made using modern engines like Unity and GameMaker Studio respectively, sticking within the authentic parameters of retro platforms may be a fool's errand. In Matt Phillips' case, the only solution was to go right back to the original hardware itself.

Unlike the above, Phillips has a development background in working on triple-A projects,



Downwell creator Ojiro Fumoto is among the collective, and one of his contributions is of course a shoot-'em-up with a twist.

from the *LEGO* franchise to *Homefront: The Revolution*, but had always dreamed of releasing his own Mega Drive game. Not just a game that replicates the look and sound of a classic Mega Drive game, but one produced on a cartridge that can run on the original hardware. He finally realised this dream in 2018 with the release of adventure platformer, *Tanglewood*.

This not only required Philips to learn how to program in the raw assembly language of the Mega Drive, but also the painstaking effort of trawling niche forums and contacting electronics experts over several years just to acquire all the necessary parts to get a complete and functioning Assembly kit. Unsurprisingly, having been out of the hardware race for nearly two decades, Sega was unable to assist.

It's certainly the most extreme of all the methods of making a game that's faithfully within the parameters of retro hardware, because this really was about using the original hardware: no nips, no tucks, no shortcuts. "[With retro indie games] you can cheat," says Phillips. "You can add more colours than palettes supported back then, you can go over sprite limits. Even down to the resolution, you can render up to 1080p widescreen, which wasn't a thing back then. So if you wanted to make a game that was very true to the 16-bit style, the only way to do it is to make a 16-bit game, then your hand's forced to obey the restrictions."

Just assembling the original development kit for the Mega Drive was an arduous mission.

Tanglewood looks retro, but its exploration and puzzle-solving feel modern.

Although the setting of *Tanglewood*, where you play as an adorable fox, echoes the classic mascot platformers on the Mega Drive, from *Sonic the Hedgehog* to *The Lion King*, Phillips actually took inspiration from more contemporary games and game design, citing *Limbo* or PS1 era platformers like *Heart of Darkness* and *Abe's Oddysee*, while much preferring the physics-based movements of modern platformers.

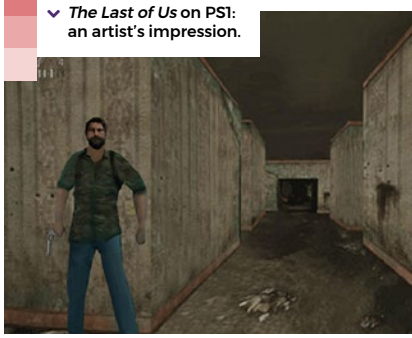
Attempting to translate modern physics to a 16-bit console's limited CPU presented a huge challenge that other contemporary, retro-style developers don't need to take into account. To manage this, Phillips had to simplify the physics and write bespoke code to give the impression of objects behaving like there are collision effects going on. "There were a lot of little hacks and tricks to get things to work and save CPU cycles," he says. "One of the things I had to take out was buoyancy. I had a lot of puzzles designed around objects floating on top of water, but that was using way too much CPU." It was just one of many features that, owing to the hardware limitations, had to be left on the cutting room floor.

It's fair to say that few developers would go to the same lengths to faithfully create an authentic retro game, not when the likes of GameMaker and Unity make the process so much simpler. But Phillips' passion has paid off, and he already has plans to make more Mega Drive games in the future. Ultimately, it proves that there's more to making a retro game than just paying lip service to pixels. "I recommend all developers try and limit themselves with regards to CPU usage, memory, palettes, and try to make a game on a limited system, because it changes your perspective on how you approach game design and programming," he explains. "It means you have to go back to the drawing board with regard to game design, because you have to make it fun from the outset. You can't throw pretty graphics or high-resolution audio at it to try and polish it – you have to base it on game design and playability alone. I think there's a lot to learn from that."

DEMASTER PLAN

Pixel art may be associated with retro, but what about generations who grew up in the 32-bit era experiencing the early forays of 3D gaming? While the low poly models and textures haven't aged well, there's still some fondness for the style that's prompted the creation of PS1 interpretations (referred to as 'demakes' or 'demasters') of contemporary hits like *The Last Of Us* or *BioShock*. That these are more artist tributes than genuine games however may suggest it's more of a curio than something people genuinely want – the underwhelming sales for the PS1 Classic rather says it all.

The Last of Us on PS1: an artist's impression.



Matt Phillips, director of Big Evil Corporation.

How I made Deus Ex Machina

Video game pioneer Mel Croucher shares the story of his groundbreaking 1984 multimedia experience

WRITTEN BY
MEL CROUCHER

This is the story of a video game called *Deus Ex Machina*. Some players reckon it changed their lives. It certainly changed mine. First I wrote it, then it put me out of the business for 30 years. Here comes the story.

There are only four elements in any video game ever written: chess, dice, ping-pong, and bullshit. Every shoot-'em-up, all swords-and-sorcery, each adventure or sports simulation, everything and anything that passes for computer gaming is a combination of these elements, remixed and regurgitated. The strategies of chess, the throw of the dice, the hand-eye coordination of ping-pong, and the con-trick of bullshit. And if this is the story of *Deus Ex Machina*, then it is essentially the story of computerised bullshit on a grand scale.

Six years after starting my video games company back in 1977, I had grown weary of it all. What I wanted to create next was an extension of dreaming, and the time had come for me to dream big time. So I set about creating the world's first interactive movie.

Deus Ex Machina took me three weeks to design. I wrote the mechanics as a screenplay and gave the instructions for the programmer as a shooting script. It took me another six weeks to write and record the soundtrack. I played all the instruments myself, and I was a rubbish musician, so it took ages to edit out all the bum notes. And the reason I played all the instruments myself was because I couldn't afford to hire any real musicians.

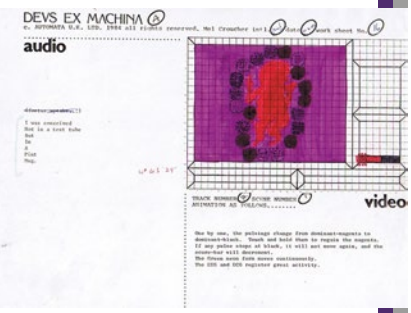
The thing I wanted to achieve with *Deus Ex Machina* was to force the player to become the active central character, not the passive viewer. And I figured the best way to do that was to get them to go with the flow of the music and the gameplay, so the soundtrack would play inside their head, before dragging them through the screen to live out their entire lifespan in one hour. In 1984 I was restricted to the home-computer capacity of the most popular machine around, which was the 48kB Sinclair Spectrum. When I was told I couldn't fit *Deus Ex Machina* into the 48kB available, it took all of one minute to figure out that I could double that available memory by inviting my players to pause halfway through the game while the computer saved their current score, then flip over the data cassette and wait for the second half to load. Suddenly, the 48kB Sinclair Spectrum could play a 96kB game. And although that seems like a breakthrough in hindsight, I had absolutely no notion it hadn't been done before.

SOUND AND VISION

Doubling the computer's virtual memory also meant I needed to package the game on two separate cassettes. One for the soundtrack, the other for the code. I added an audio countdown to the soundtrack, and let players sync the start of each half of *Deus Ex Machina* themselves. In that way, the soundtrack would run in sync with the action on the screen, or at least as far as stretched magnetic tape would allow.

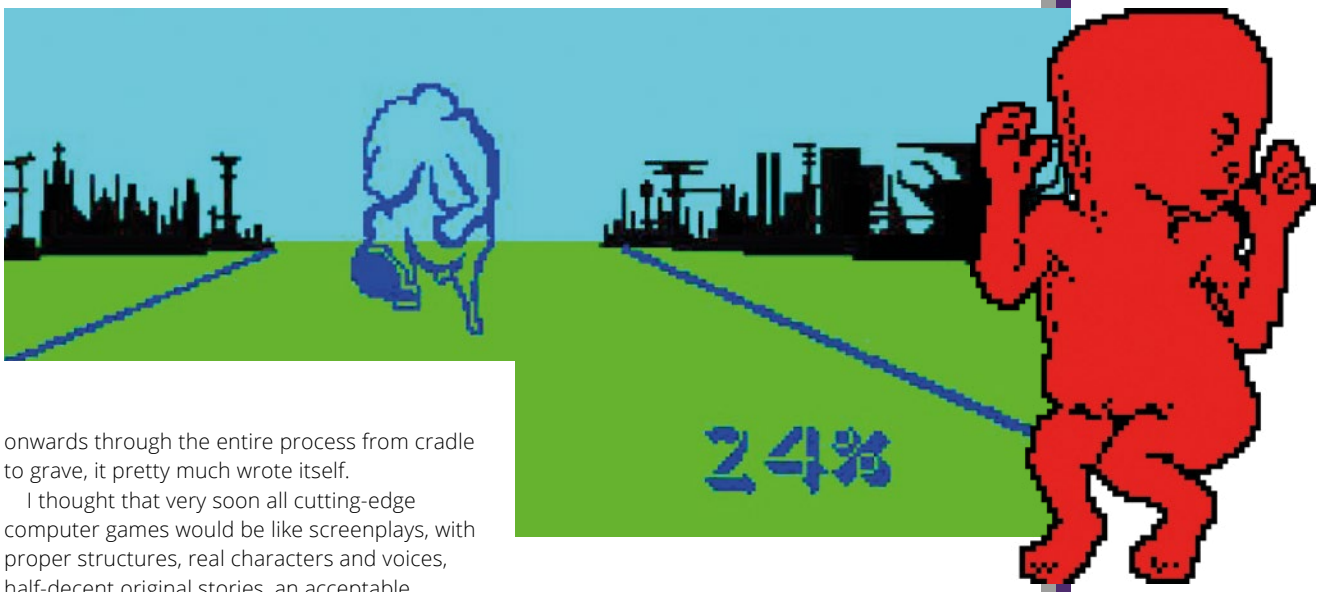
The game concept was almost irrelevant. I used a sequence of interlinked game-plays, no more original than the stuff I had produced before, but that wasn't the point. Movies are also a series of unoriginal sequences. It's the recombination and original presentation of stolen ideas that sells tickets. I wanted to allow players to control the way they reached their inevitable destination. So from the fertilisation of an egg by a sperm, to their own birth, then

One of Mel Croucher's original storyboards, detailing how the audio syncs up with the game's imagery.



CULT VOICES

The total cost of creating *Deus Ex Machina* was £8,760. That was everything I had at the time, and I spent nearly all of it paying celebrity voices to appear on the soundtrack. Today it's common practice to hire celebrities to recite in-game twaddle, but because I knew no better, it seems I was the first to do it. I targeted three cult voices, phoned them up, and offered them cash. It was a doddle. I paid *Doctor Who*'s Jon Pertwee, with his golden voice. I paid the godfather of punk, Ian Dury. And I paid *Carry On* star Frankie Howerd. Then I ran out of money.



onwards through the entire process from cradle to grave, it pretty much wrote itself.

I thought that very soon all cutting-edge computer games would be like screenplays, with proper structures, real characters and voices, half-decent original stories, an acceptable soundtrack, a variety of user-defined narratives, and variable outcomes. It still gives me great satisfaction when people tell me about the influence it had on them. Mind you, many of them say they were on hallucinogens at the time.

As it happens, *Deus Ex Machina* became known as a 'multimedia' product, slightly ahead of its time. But only by a quarter of a century. The game wouldn't work without the concept album, and the music wouldn't mean much without the game. I also wanted to make the packaging an important part of the whole thing, and I bunged in a movie poster, a short story, and an instruction manual. And I made the key image a beautiful robot to hopefully give some of my players crowded trousers. But I never suspected the utter debacle that was about to happen. It stemmed from organised piracy on an industrial scale, and I never even knew it was happening until it was too late.

Deus Ex Machina was just another computer game, but for a few short weeks, I thought I had produced a real winner. The delusion began as soon as the reviews came out. It wasn't that I was naive or stupid about the possibilities of organised piracy. The thing was, I thought *Deus Ex Machina* was too complex to get ripped off. After all, it would not be cheap to reproduce a vacuum-moulded presentation case, a double-sided poster and the twin-cassette format. And as for the audio and printed storyline, they were hardly going to come up with the full monty. Were they?

PRICE WARS

In 1984, punters were paying around five quid for a popular video game. A vinyl rock album cost about the same, and so was a trip to the movies. So I got it into my head that seeing as how I was offering players a combination of all three entertainments, the street value should reflect this. Besides, I wanted to recoup my investment sooner than later. And so I overpriced the product at 15 quid, the most expensive price tag of any game at the time. Big mistake. As a premium price product, it made the game a motivational gift to the pirates, but they only bothered to reproduce the computer code cassette and ignored the rest. I still cringe at the thought of a legion of players left completely baffled by a crap silent movie of a naked avatar blundering through a load of dopey sprites.

"I set about creating the world's first interactive movie"

Deus Ex Machina got the best reviews I've ever had for anything, and yet it was a commercial disaster. By the time it was awarded Game Of The Year by the Computer Trade Association and won the Golden Joystick Award, I had gone into an epic sulk, fuelled by disappointment and frustration, because the game simply wasn't selling enough.

I refused to attend the awards ceremony and disappeared on holiday. I blew it. What is never predictable is how a bestselling creator handles the downward slide towards oblivion. The way I did it was to give up. Then I waited 30 years to remake it, and ended up in a wheelchair.

But that's another story.

***Deus Ex Machina*'s surreal mini-games took the player on a journey through the human life-cycle.**

The packaging may have pushed up its cost, but *Deus Ex Machina* certainly looked and sounded unlike anything else available in 1984.

Developer Profile

FromSoftware

So much more than a one-trick studio

That FromSoftware is boiled down to being 'the Soulsborne studio' would be, for any other company, a badge of honour. *Demon's* and *Dark Souls*, *Bloodborne* – even *Sekiro: Shadows Die Twice* – they're all superb games; inventive, challenging, and honed to a ludicrous degree. If another studio were linked to this loose collection of vaguely similar titles, it would be a happy studio indeed. This is FromSoftware though, and to distil its output down to just the games it's best known for is, frankly, a slap in the face. From has done a lot more over the years than it's given credit for.

After dabbling in productivity software for almost a decade following its 1986 formation, FromSoftware took the plunge into game-making with *King's Field*. A launch title for the Japanese PlayStation, this first-person RPG brought with it an impressive 3D world, a dark atmosphere full of foreboding, and a stamina bar players were always having to keep an eye on. Safe to say this first title acted as an inspiration for what was to come.

Before that, though, FromSoftware branched out – a couple of *King's Field*

sequels soon led into 1997's *Armored Core*, and the team's first shot at a genre that would become something of a calling card: that of giant stompy robots (or 'mecha', if you prefer). The sheer level of customisation on offer throughout *Armored Core* also introduced a concept that would prove an inspiration for the studio's later years: the idea that the player was free to approach a situation how they wanted to. Prescribing how they should play was a no-no.

PREPARE TO TRY

Well, unless From was to make a bunch of more traditional RPGs, of course, which it subsequently did. *Enchanted Arms*, *Evergrace*, *Lost Kingdoms* – few gave Square Enix a run for its money, but each introduced unique ideas and aspects that made them worth a go. And, not content with walling itself in, FromSoftware kept on expanding, trying new things, and forging ahead with seemingly any half-decent idea it could come up with. Sure, they weren't always great – the PlayStation 2 horror title *Kuon* proved somewhat lacking, for example – but laurels weren't there to be rested on.

The giant stompy robots theme continued with plenty of *Armored Core* titles showing up, but there were a fair few diversions along the way with the likes of *Murakumo: Renegade Mech Pursuit* and *Metal Wolf Chaos* proving mediocre and ample distractions, respectively. *Another Century's Episode* brought together mecha and licenses – another area of gamemaking From never shied away from – with reasonable success. The studio even went cutesy at a couple of points, with *Kuri Kuri Mix* surely providing

***Armored Core*: missing, but not forgotten.**

Miyazaki's work at From was rewarded with promotion to president in 2014.

From concept to finish, *Dark Souls* maintains an air of... well, darkness.

at least a *bit* of a surprise to the ardent *Soulsborne* fan.

And – skipping ahead – even after achieving worldwide recognition and acclaim with a certain batch of releases, FromSoftware has still shown its willingness under president Hidetaka Miyazaki to take chances and do things a bit differently. 2018 saw the PSVR release of *Déraciné*, which quite frankly took everyone expecting a fifteenth *Armored Core* game by surprise when it was announced. This is a studio with a 33-plus-year history, 25 of which have been spent

"FromSoftware has managed to never fall into the trap of predictability"

making games, and it has managed to avoid falling into the trap of predictability or triviality – all the while taking chances and flexing its creative muscles wherever and whenever it chooses.

But... well, the *Soulsborne* games *are* special, aren't they? Even when trying to look past the obvious choices (see the next couple of pages for a concerted effort at that), it's nigh-on impossible to ignore the fact that *Demon's Souls*, *Dark Souls*, *Bloodborne*, and – now – *Sekiro: Shadows Die Twice* exist. Because (with some ups and downs) they're stunning. The focus might often be on the difficulty of each title, but it's the purity of design that shines through for those really engrossed in them.

While *Demon's Souls* acted as a test run for what became the *Dark Souls* games, each of the spin-offs has proved more than capable of being its Own Thing too. *Bloodborne* brought with it a need to relearn your fighting styles and incorporate 'trick' weapons, while *Sekiro* demanded players get off the back foot and firmly onto the front. Fundamentally similar they might be, but in actually playing them they couldn't be more different.

And maybe that's it – the *Soulsborne* titles really do personify what FromSoftware represents: a strong foundation; smart design; a willingness to mix things up and take chances; and a basic, primal desire to punish you for having the temerity to think you can look past it all. Don't underestimate FromSoftware – the studio's about way more than just a collection of the best games ever made.

Soulless

10 non-Soulsborne corkers

Because there's more to FromSoftware than Dark Souls

01

King's Field

PS1 – 1994

From's first ever game was also a launch title for the PlayStation in Japan, the only country in which it was released. A first-person RPG smothered in darkness and riddled with difficult moments to overcome, it set the template for most of the studio's output for the next quarter-century. The series continued for several years, with limited success.

02

Armored Core

PS1 – 1997

Players of a certain age are likely to have encountered this as their first taste of From's software, with *Armored Core* managing to make an international dent thanks to its deft mix of engaging robo-action and almost *Gran Turismo*-like customisation options. Allowing the player to choose their own approach was a pattern the studio would continue to abide by.

03

Frame Gride

DC – 1999

Initially a PlayStation-only dev, From's first move elsewhere was to the Dreamcast for this – another giant stompy robot-'em-up, *Frame Gride*. Taking cues from *Armored Core*, *Gride* saw two players battling it out in customisable robots, while the game's online mode made sure to make this a unique offering by taking advantage of the DC's hardware.

04

Kuri Kuri Mix

PS2 – 2000

Already owning a bit of a reputation for making dark-and-serious games, *Kuri Kuri Mix* showed From's lighter side. This cutesy number offered a creative approach to things, with its multiplayer section requiring two players to work together to overcome obstacles – using just one controller, if they so chose. Single-player was… less engaging, let's say.

05

Otogi: Myth of Demons

Xbox – 2002

A bit stilted and never a true great, sure, but *Otogi* acted as a tentative dip into what would become the *Soulsborne* sub-genre. The Xbox exclusive saw its protagonist fending off a demon invasion, existing in a world between life and death, and engaging in difficult melee combat in a series of dark environments. Sound familiar?

06

Lost Kingdoms

GC – 2002

From had already dabbled in the more traditional RPG realm by this point, but *Lost Kingdoms* marked a standout point for the studio's efforts in the genre. The real-time combat sequences relied on deckbuilding/card battling strategies, providing a unique experience in a somewhat oversubscribed early noughties genre.

07

Shadow Tower Abyss

PS2 – 2003

Another title that's often overlooked, this *Shadow Tower* sequel features numerous elements that would become standard features in *Demon's Souls* and beyond. Dark fantasy, role-playing elements, and the collection of souls of vanquished foes to level up all featured in the PS2 game. While largely forgotten, it certainly helped pave the way.

08

Metal Wolf Chaos

Xbox – 2004

Recently re-released via Devolver Digital, *Metal Wolf Chaos* was Michael Bay in video game form. Players took control of the US President, who in turn was controlling a giant mech battle suit in order to fight back against the rebel forces led by his own Vice President. The game itself was fun, but the idea behind it was pure, manic brilliance.

09

Ninja Blade

360 / PC – 2009

Before breaking through in a big way with *Dark Souls*, From tried to crack the West. *Ninja Blade* was made with the US console market in mind, designed with all the over the top action and exciting ninjutsu you could hope for in such a game. It wasn't very good, mind, but it's at least a noteworthy waypoint in the studio's long and storied video game history.

10

Déraciné

PSVR – 2018

Firmly established following the *Soulsborne* games, From took a chance with this Miyazaki-directed VR title. A first-person adventure, *Déraciné* certainly wasn't a case of fan service or more of the same, and while it was ultimately a bit lacking, it showed the studio still has a hunger to try out new things as and when it wants.

Subscribe today

SAVE 49%

wfmag.cc/**subscribe**

13 issues
for just £20

Subscriber benefits

- **Free delivery**
  Get it fast and for free
- **Exclusive offers**
  Great gifts, offers, and discounts
- **Great savings**
  Save up to 49% compared to stores

Introductory offer

Rolling monthly sub

- **Low initial cost** (from £4)
- **Cancel at any time**
- **Free delivery to your door**
- **Available worldwide**

Subscribe for 12 months

Receive all 26 issues

£40 (UK) **£75** (USA)

£65 (EU) **£75** (RoW)

Offers and prices are subject to change at any time

Digital subscriptions from

£1.99

Available on the App Store

GET IT ON Google Play

Visit **wfmag.cc/subscribe** or call **01293 312192** to order

Subscription queries: **wireframe@subscriptionhelpline.co.uk**

Review

Generation Zero

Avalanche's robot uprising soon runs out of energy

Info

GENRE
FPS / Survival

FORMAT
PC (tested) / PS4 / XBO

DEVELOPER
Avalanche Studios

PUBLISHER
Avalanche Studios (PC) / THQ Nordic (PS4, Xbox One)

PRICE
£29.99

RELEASE
Out now

REVIEWED BY
Joe Parlock

HIGHLIGHT

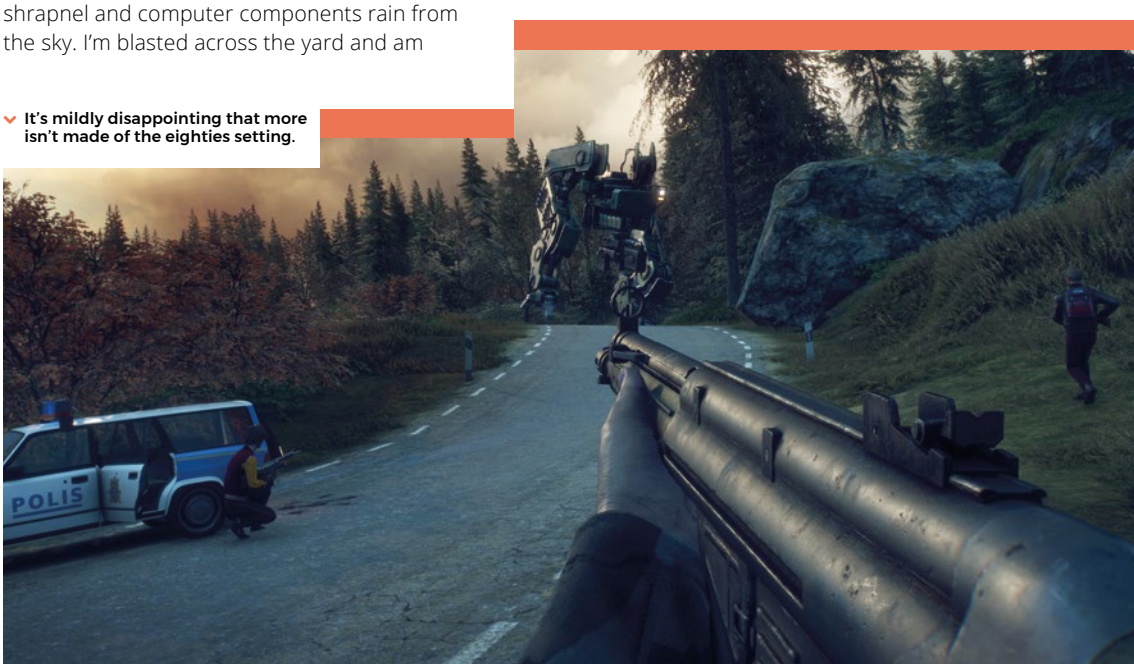
When you're fighting for your life against a horde of machines, *Generation Zero* comes alive. Desperate improvisation and gambling your life on snap decisions is intense, and does a fantastic job of putting you into the mindset of a survivor, rather than that of someone just playing an FPS.

There's a robot in the garage. I managed to trap it in there when it charged at me with its giant chainsaw, and now it's pacing around, buzzing and bleeping and doing other murderous robot things. I've got a plan on how to deal with it, but it's a terrible one. The first step is to find a gas canister. There's one around here somewhere, but the tiny spider-bots leaping at my face make searching for it difficult. Once that's done, I'll be able to open the door, drop the canister in front of the robot in the garage, and blow it, myself, and all the other robots in the area to kingdom come. Fortunately, I've got an adrenaline shot to make this less of an idiotic idea than it sounds, but it'll still be a tad messy.

I dash into the nearby house, shotgunning robotic spiders as I go. One of the medium-sized dog-bots spots me and opens fire, but I take cover in the bathroom upstairs and... bingo. I grab the canister and return to the garage. I have less than a second to pull this off before the big guy in the garage is clear of the blast, but I manage it and, with a meaty, satisfying boom, shrapnel and computer components rain from the sky. I'm blasted across the yard and am knocked out, but the adrenaline shot keeps me from dying. Now I get to search the corpses of every robot caught in the explosion for goodies, before moving on to my next objective.

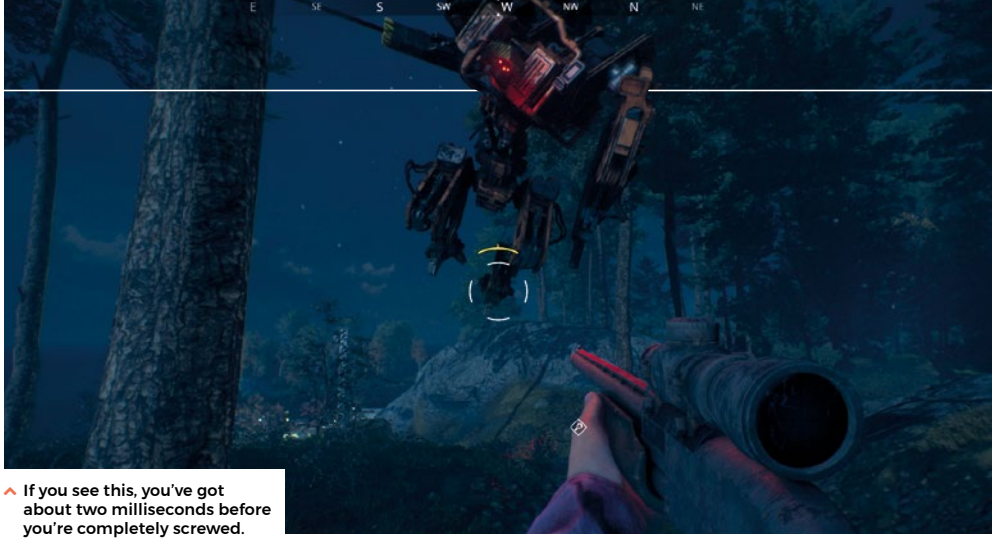
It's moments like this that make *Generation Zero*. Set in an alternate 1980s where Sweden has been abandoned and overrun by killer robots, the setting should've been enough to carry this game without the remarkably clever systems-based interactions. Situations like the one above are rarely scripted, making the game an excellent canvas for player-driven stories. Unfortunately, the world that holds these stories doesn't hold up to scrutiny, and *Generation Zero* begins to rust and fall apart after only a few hours spent wandering its forests.

It's the lack of handholding that makes Avalanche's newest title (only four months after the dismal *Just Cause 4*) such a compelling sandbox. With an almost complete lack of tutorials and very vague objective markers, it's up to you to learn the landscape, the intricacies of your equipment, and the new, mechanical ecosystem that threatens to eviscerate you at



It's mildly disappointing that more isn't made of the eighties setting.

Make a mental note of this house, because it's going to pop up all over the place.



If you see this, you've got about two milliseconds before you're completely screwed.

every turn. The more you understand the rules, the game turns you from frantically running for your life and into a hardened and weary survivor. If any game comes close to evoking the self-determination and smart survival of *Black Mirror*'s '*Metalhead*' episode, this is it.

The enemy designs are also stunning. Piecing together what they even are before engaging them is part of the challenge. These robots have moved beyond their human creators' understanding of anatomy, often only slightly resembling the living creatures we'd use to relate to them – that floating doohickey may look harmless, but its alarm that summons even more vaguely dog-shaped bots definitely isn't. The way enemies move, strategise, and eventually fall apart (which is immensely satisfying) all come together to build some of the most interesting robot designs seen in games for a very long time.

Generation Zero sets up fantastic systems and a compelling aesthetic, but the world they're let loose in is by far the game's biggest flaw, and, sadly, it kills the game dead on its feet. When it's not empty, it's repetitive. When it's not repetitive, it's mundane. When it's not mundane, it's half-baked. Avalanche never manages to do the premise justice, and it's impossible to not question whether the game was actually close to being finished by the time it launched.

The environments are cookie-cutter – in the space of just the first hour, you could easily go into five different houses in five different locations that all, for some reason, have the exact same internal layout. Exploring a house should be an integral and intimate part of a survival story like this, but instead, you go through the motions because you already know where every room is and what is in it – right down to knowing where the toilet is before stepping in through the front door.

The same applies, to a lesser extent, to the military bases. The first time you crawl through one, stumbling through the pitch blackness, it's an awesome experience. The second time, you've already seen everything a bunker has to offer, despite it bearing on the other side of the country from the last one. What is the point in a game with such a focus on exploration when it spills all of its secrets in the first few minutes?

The narrative isn't much to phone home about, either. *Generation Zero* could've learned a thing or two from *Gone Home* or *Everybody's Gone to the Rapture*, as everything is told through boring audio logs, abandoned letters, or exposition-dumping newsletters conveniently left for you to find. None of it does anything worthwhile with the 1980s setting – this could have happened in 1960 or 2019 and nothing would have felt different, bar a few items of clothing you can wear for stat boosts.

"When it's not mundane, it's half-baked"

The result is that all of *Generation Zero*'s brilliant moments feel more like a fluke than something Avalanche actually intended to happen. You may come away from it with dynamic and player-driven stories of survival in the harsh, post-human landscape, but you'll just be taking joy in the few gleaming moments that happen by accident.

Ultimately, with a bland and underused world, unfinished environments, and an entirely forgettable story, *Generation Zero* is the poster-child for wasted potential.

Let's get David Attenborough to narrate the robot uprising.

VERDICT

With a neat premise and some genuinely awesome robot designs, *Generation Zero* starts out strong, but runs out of steam fast.

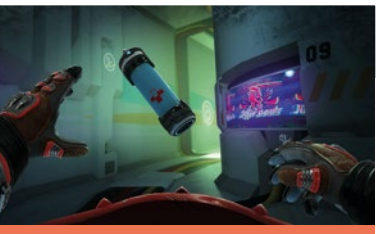
55%

Big, beautiful, mind-numbingly empty.

HIGHLIGHT

The pre-game lobby has a set of bouncy beach balls and a target practice minigame, so you can shoot some hoops and hone your shooting skills as you prepare to in the orbital arena.

You can choose two active abilities prior to each multiplayer match, which range from protective shields to medkits and laser sabres.



One of the few comfort options in the game allows you to toggle your helmet and blinders if you're keen on improving your FOV.

Space Junkies

Ubisoft's daft dogfighter gets lost in space

Info

GENRE
Arena Shooter

FORMAT
PSVR (tested)

DEVELOPER
Ubisoft Montpellier

PUBLISHER
Ubisoft

PRICE
£29.99

RELEASE
Out now

REVIEWED BY
Jordan Oloman

VERDICT

Despite the pleasing polish, *Space Junkies* is a barebones control scheme catastrophe in PSVR.

39%

When you're first flung into the stratosphere as part of *Space Junkies*' Zero-G tutorial, it's hard not to be impressed. The fluidity of the animations and the polished textures stick out and immediately set a gulf between this game and most PSVR competitors.

Then you hear it. A decidedly aloof narrator sounds off, doling out zany one-liners and goading you through the firing range. As you awkwardly tumble through a set of gates, algorithmic dubstep starts playing. It's happening again. Why does every futuristic VR game settle for an inane, self-aware "Isn't it crazy that we're in virtual reality right now" premise when it could be so much more?

You're not left looking for more with the cornucopia of weapons to choose, ranging from explosive slingshots to electric snipers, and when you run out of ammo, you can throw them like grenades at your enemies, which is deeply satisfying.

Space Junkies pulls from the iconic arena shooters of yore, but eschews a sprawling deathmatch format. Instead, there's a more intimate approach, delivering small matches where you spend most of your time searching for an enemy to kill rather than firing a gun. But when you eventually do stumble upon someone else, you also find yourself coming into contact with the game's bewildering control scheme.

See, *Space Junkies* features crossplay; necessary, given its small playerbase. But when you – a console player, in my case – are dropped into a lobby with PCVR players, it quickly becomes apparent they have plenty of control and you... don't. *Space Junkies* only supports the DualShock 4 instead of the (ancient, but nifty) Move controllers.

Using your head to look where you want to go, the pad's thumbsticks for movement and aiming your gun by physically moving the controller is confusing at best. A huge omission is smooth turning, so if an enemy appears behind you, you must jerk your viewpoint in 90-degree jumps. For an arena shooter where you need to be accurate, this is a (nauseating) deathblow.

It does have its moments. Audio design is a triumph – dashing through asteroid innards, leading in King of the Hill mode, I hear my character's breath get heavy as I await the next challenger. And the crunchy sound effects and accompanying vibration when turning the opposition into virtual mince with a minigun is grim fun. On occasion, the network stars would align and I'd play a four-person deathmatch; a boon for any game. But when a team got the upper hand, one player would often just leave. As there doesn't appear to be any hot-joining system, the remainder of the match became a laughable, unbalanced game of cat and mouse.

So I loaded back into the pre-game lobby, dejected, feeling cynical toward one pre-teen PCVR player enthusiastically onomatopoeic while he tested guns. And that's where it hit me: the PSVR port of *Space Junkies* makes me feel like I'm in detention, while everybody else is running free in the playground.

Many of the trials you'll oversee have historical precedents.

HIGHLIGHT

The court case scenes are the clear standout here. Exhaustive, detailed case notes give way to impassioned pleas and barbed heckling from the stands. Some cases tug at heartstrings, others boil the blood. Devastatingly, you can often only afford the indulgence of staying true to your own morality once your political associations are satisfied.

Review

We. The Revolution

Violence baguettes violence

There's a repeated scene in *We. The Revolution* that feels like a recurring nightmare. A bloodied guillotine blade lifts, revealing a crowd of spectators. Some are grinning, satisfied justice has been meted out. Others are sickened, or weeping. It's in the faces of the latter I get the impression the game's soul lies. You can imagine the fervour that led them to the execution, only to have their illusions give way to an underlying humanity that cannot abide suffering, no matter what ideals it supposedly supports. Are notions of liberty, egality, and fraternity just academic utopianism when faced with the baser elements of human nature? Should an official's heart lie with his family, or his country? Are a few deaths acceptable in the struggle for a fairer society? These are the questions *We. The Revolution* asks, and it asks them with an unflinching gaze at our capacity for evil. Oh, and there's also a bunch of rubbish minigames. There's no poetic way to say it, they're just a bit naff.

You'll experience the game's events through the eyes of judge Alexis Fidèle. Somewhere between ruthless aspiration and abject cowardice, he's a hard man to like, but his folly and tragedy are expressed well enough through extensive, deft writing that his tale is immediately gripping. The initially focused story later expands its scope, with varying degrees of success, but the game's writing is at its best when it feels like an intimate, almost claustrophobic stage play.

As a player, your own engagement with the story comes from a seemingly endless string of decisions. These are dressed up in various minigames, only one of which doesn't feel mind-numbingly arbitrary. The courtroom scenes, which I'll talk about in a moment, are mechanically interesting and thematically strong. Everything else, from dice games to mock battles, feels like padding. Aiming for variety is admirable, and each of these distractions is elevated by pleasingly detailed animation, but they mainly just feel indicative of a lack of trust in the player to not get fidgety if a new system isn't catapulted at them every half hour.

"A grimdark lens on an already grim and dark period"

Those courtroom scenes are great, though. What starts out as a succession of lengthy, thoughtful trials soon slides into desperate affairs as the passions of the populace run high and the need to keep various factions content threatens to erode justice altogether. Like *Papers, Please* before it, *We. The Revolution* aligns player and character to make the manipulation of its systems feel like a stark betrayal of your own morality.

There are points in *We. The Revolution*'s story where the unshrinking portrayal of historical atrocity feels too close to voyeuristic cruelty. A grimdark lens on an already grim and dark period of human history feels like overkill, but is perhaps preferable to an attempt to sand down the setting's sharp edges. Either way, I wouldn't play *We. The Revolution* on a bad day, but it's still refreshing to switch out one-liners for thoughtful prose, grit and gravel for guillotines and gavels.

Info

GENRE
Narrative/ Minigames

FORMAT
Steam (tested) / XBO / PS4

DEVELOPER
Polyslash

PUBLISHER
Klabater

PRICE
£15.49

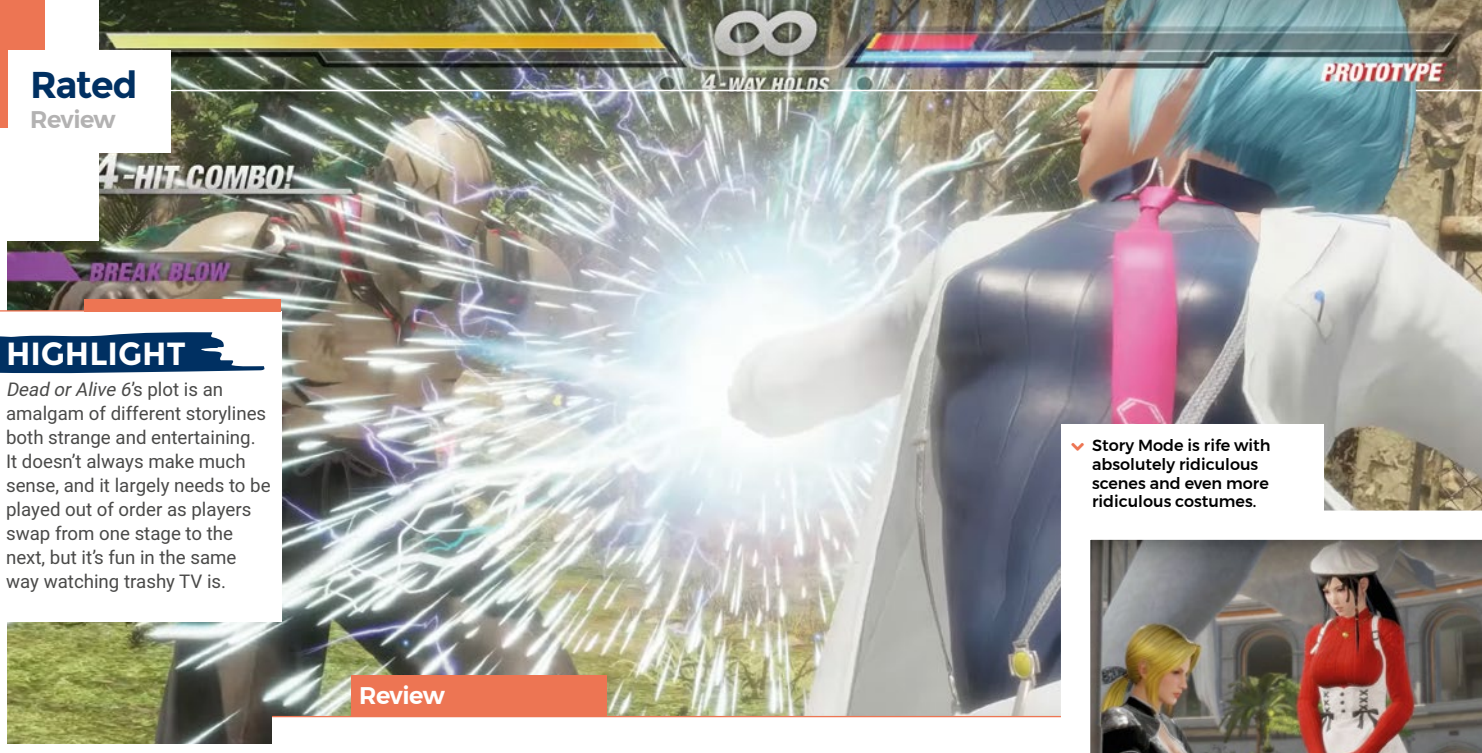
RELEASE
Out now

REVIEWED BY
Nic Reuben

VERDICT

As gruelling as it is creative, *We. The Revolution* spins an evocative historical drama.

66%



Story Mode is rife with absolutely ridiculous scenes and even more ridiculous costumes.

Break Blows are explosive attacks used with full Break Gauges that can absolutely wreck opponents.

HIGHLIGHT

Dead or Alive 6's plot is an amalgam of different storylines both strange and entertaining. It doesn't always make much sense, and it largely needs to be played out of order as players swap from one stage to the next, but it's fun in the same way watching trashy TV is.

Review

Dead or Alive 6

This isn't exactly the fighter fans have been wanting – dead or alive

Info

GENRE
Fighter

FORMAT
PC (tested) / XBO / PS4

DEVELOPER
Team Ninja

PUBLISHER
Koei Tecmo

PRICE
£54.99

RELEASE
Out now

REVIEWED BY
Brittany Vincent

Dead or Alive 6 takes the franchise's better moments and delivers a mostly delectable entrée that'll delight players looking for tried-and-true flavour. Unfortunately, by the time dessert arrives, you'll be left wanting something more palatable.

The same fighters fans have come to expect to fill out the roster, but street brawler Diego and scientist NiCO steal the show; Diego brings some much-needed swagger to matches, while NiCO fills the 'quirky anime scientist' quotient nicely. The old fighters shouldn't be ignored, of course; Honoka can blaze past the competition and KO any sucker before they know what's happening, while series stalwart Bass is (still) heavier, deliberate, and extremely powerful.

"It's all impressively physical and decidedly crunchy"

Combat is meaty and satisfying, with *DOA6* happy to let players feel as though they're batting around flesh-and-bone fighters instead of flailing bags of sand. When your blows connect, one word will flash through your mind: 'Wrecked'. Those punches and kicks don't just look like they hurt – they *do*.

Meanwhile, fighters display visible damage for the first time in the series – sweat flying through the air, noses bloodied... There's something devilishly satisfying about seeing your opponent visibly roughed up, sweat pouring down their cheeks. And there's just as much satisfaction in ending a battle with a Break Blow. It's all impressively physical and decidedly crunchy.

Unfortunately, where combat shines, just about everything else disappoints. Story mode finds players completing a series of fights that feel lazily cobbled together. DOA Quest features a series of missions which aren't much better, but they're the only reliable way to unlock new items for each fighter. Finish enough missions and you'll get in-game currency for bonus goodies, but the system is fundamentally broken.

How? Well, points are earned through completing quests, but they don't go toward the specific character you've completed the quest *for*. So a quest for Helena could end up with parts for Kasumi, and so on. It's a clumsily designed system rewarding only completionists and those with a lot of time to spare – not a general fighting audience.

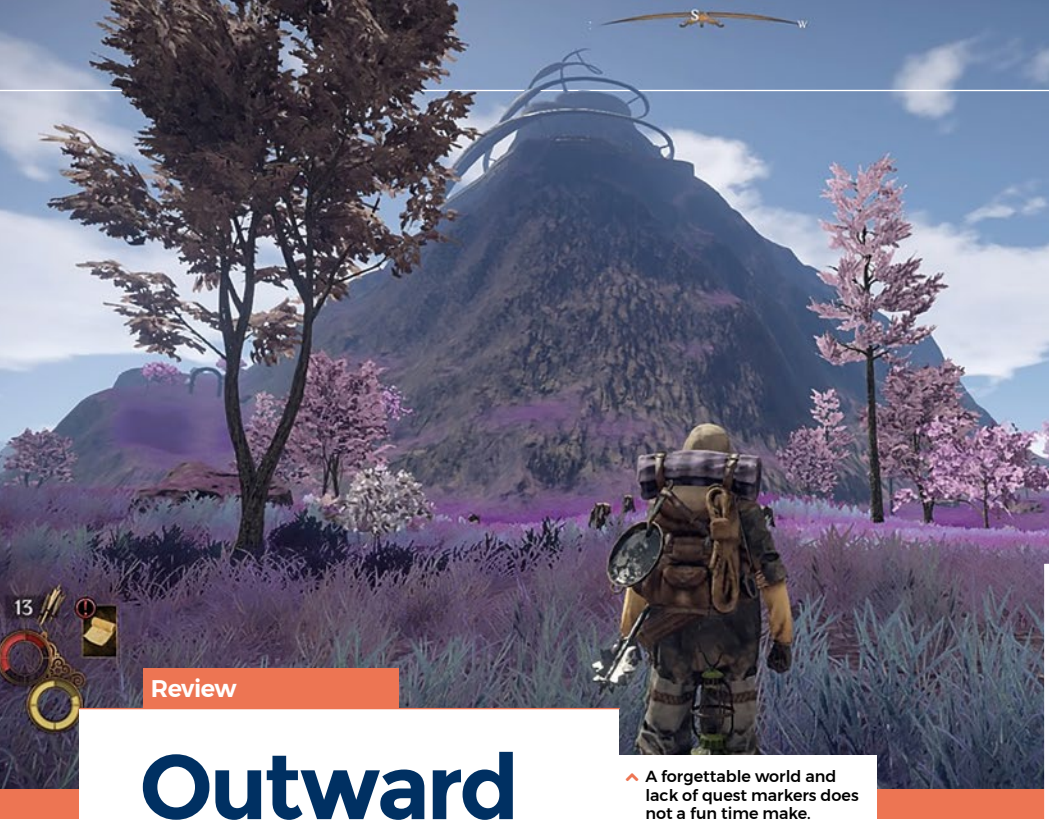
There *is* excellent netcode, thankfully, making it a breeze to start brawling online right away. Unfortunately, match options are bare bones, featuring only ranked matches at the time of writing, but set-up is at least quick, reliable, and easy.

Dead or Alive 6 is, in terms of where it fits within the overall series, more of the same. It's decent but not great, with flaws stopping it from being a truly accessible choice for those looking to get into the genre, while hardcore fans looking to collect outfits for their favourite waifus won't be too pleased, either. By and large, it's a serviceable step forward that could have used a bit more time in the oven.

VERDICT

A punched-up fighter with underwhelming game modes that make it difficult to recommend.

62%



Review

Outward

A forgettable world and lack of quest markers does not a fun time make.

Escape to a magical world of hunger and debt

For many of us, video games offer escapism – a power fantasy. That's especially true of RPGs: strange worlds, magical powers, extraordinary creatures. In *Outward*, though, you play as an ordinary human. You are not the chosen one. In fact, there's nothing special about you at all.

If you think this might strip the fun out of a survival RPG, then you'd be right. In separating their game from the competition, developer Nine Dots Studio has you exploring a lifeless world filled with laborious mechanics where the odds are never in your favour. That you start the game in debt is symbolic of the mundane life of your personality-free hero. There are no cutscenes in *Outward*; the story is minimal and the quest design basic. Find some cash or lose your home – and it's downhill from there.

Everything revolves around money. Cities have shops selling the usual RPG wares, but you can also pay for training – for a high price. There are no experience points, so unless you stumble on some decent equipment out in the wild, buying it is the only way to improve your stats. Combat takes its cue from *Dark Souls*. There's the usual stamina bar, blocks, dodges, and attacks, but without any of the finesse or satisfaction. It's brutally hard, despite some laughable AI, and more often than not you'll just run away, which only leaves you weaker down the line.

In another cue taken from the *Souls* games, *Outward* plays with death. Instead of dying, you'll black out and awaken somewhere totally different; it may be back in town with a handy recovery potion, but you're often left stranded with reduced health and stamina, and your hard-earned money stolen. Then you'll fail immediately again. It's incredibly frustrating.

So too are the survival aspects. Eating, sleeping, and drinking are required to maintain your health. Thirsty? You'll need to drink to stop the negative effect of slowed stamina recovery, so first, collect some water in your waterskin. Then boil it to prevent disease. But to boil water you'll need to make a campfire, requiring wood. Then you'll need flint to light it. Then you'll need a cooking pot, which requires money to buy, at which point you'll struggle on, parched.

"It's brutally hard, despite some laughable AI"

There's at least a strong sense of adventure, as your regular Joe slings on his backpack and trundles off into the wilderness, to the sounds of a rousing orchestral score. Yet the world is drab and empty, with functional design and flat textures. Rarely are you rewarded with useful items or intriguing discoveries. The game wants you to explore, but with difficulties at every turn, you're actively discouraged from it.

There's a lack of polish, too. Loading times are long, animations are wooden, text is tiny, and the game does little to explain its systems, leaving you lost and confused. These systems aren't inherently bad – in fact, you can see what the developer was trying to achieve – but the end experience is tedious to the extreme. *Breath of the Wild* this is not.

HIGHLIGHT

Combat does have some interesting wrinkles. Once you gain access to magic, you can mark runes on the ground to boost your attacks, and if you drop your backpack, you'll have increased mobility. Die, though, and you risk losing your entire inventory.

Info

GENRE
RPG

FORMAT
PS4 (tested) / XBO / PC

DEVELOPER
Nine Dots Studio

PUBLISHER
Deep Silver

PRICE
£39.99

RELEASE
Out now

REVIEWED BY
Ed Nightingale

VERDICT

An intriguing concept let down by poor execution. Overall, it's a decidedly deflating experience.

42%

Review

Heaven's Vault

An intergalactic linguistics dissertation

Info

GENRE
Adventure

FORMAT
PC (tested) / PS4

DEVELOPER
Inkle

PUBLISHER
Inkle

PRICE
£19.99

RELEASE
Out now

REVIEWED BY
Malindy Hetfeld

HIGHLIGHT

Heaven's Vault features one more form of language – its lovely music, with an emphasis on strings and piano, punctuates each new discovery and moment of reminiscence. It's just one other part that makes up this wonderful whole.

As a linguist, I know language can be a frustrating thing. It's frequently imprecise, changes constantly, and learning one can be a lifelong process. I wouldn't trade it for the world. Nowadays, when learning a language, we have access to a tonne of resources and can, for the most part, easily make contact with native speakers.

Archaeologist Aliya Elasra isn't quite so lucky. A prolific university researcher, Aliya travels across the galaxy in search of Janniqii Renba, a fellow archaeologist. All she has to go by is a talisman with a mysterious inscription, and Renba's last known whereabouts. The talisman belonged to her civilisation's last Emperor, whom Aliya has been looking into for a while already. It's her (albeit limited) knowledge of their language that makes her the best candidate for working out what exactly Renba was onto before he disappeared. Soon Aliya finds out that Renba was looking to uncover the truth behind a long-standing societal belief regarding the relationship between humans and robots, to which a ruin called the Heaven's Vault may hold the key. To help with her investigation, Aliya's university bestows her with a robot she names Six. This is asking me to suspend *heavy* disbelief, seeing as most researchers and their students share one copy of one book among the whole team, but hey.

To follow their trail of ancient breadcrumbs, Aliya and Six travel in a flying ship across the streams of their home expanse of the Nebula. Since the Nebula actually flows like a river, it makes sense that this spaceship is a gently creaking sailboat, complete with sails you have to fold and expand in order to steer. Moving the ship is fiddly; failing to catch a current can cause you to completely lose momentum, and the arrows representing Six's directions are often difficult to see. If you miss a turn, the game resets to a point just prior to the turn you missed on the intergalactic motorway. But we're going to ignore all of that, because...

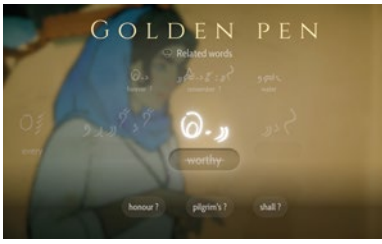
The real fun begins once you arrive on a planet. Deciphering a language from scratch is one of the game's core aspects, and luckily you get to dive in straight away. There's ancient writing everywhere – walls, doors, old items. Whenever you're meant to translate something, Aliya will first ponder whether you're looking at an entire sentence or a sentence fragment. There's no clear way to tell where one word ends and another begins, so in the beginning, you'll collect single words until you can build a sentence. This approach is just like reading

The Nebula is full of small ruins with new items.

We could say 'The game's afoot,' but we won't.

Conversation is a big part of *Heaven's Vault*.

Computer says no.



You'll find text even on small items.

Chinese or Japanese – languages in which a single character has meaning that may change in combination with other characters. If the sentence contains any words you've previously deciphered, they'll come up as options. Words that look similar to something you haven't translated yet are displayed in a procedurally generated list for context – for example, it's fair to assume the word 'empress' looks similar to the word for 'emperor'.

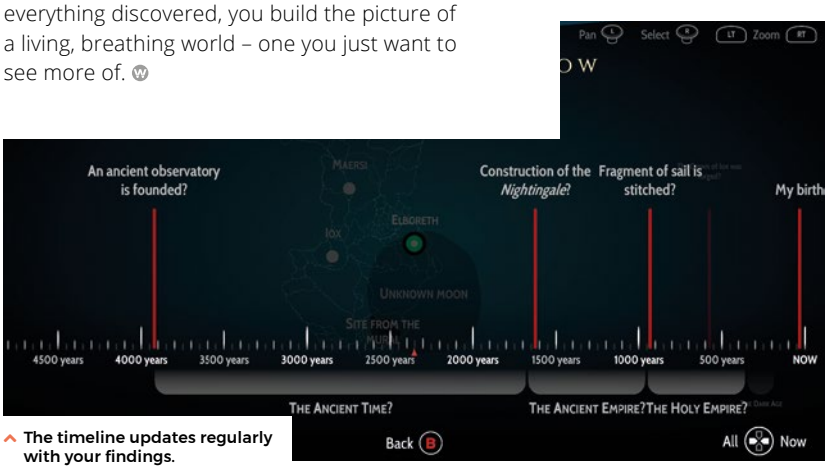
And that's another element central to the experience: translations are up to your own interpretation. Sometimes Aliyah will say that something definitely looks wrong and replace a prior translation with another possible solution, but come across a word often enough and she will grow confident in her translation. Revising a translation never feels like failure – the notion that you learn from your mistakes has never been more true than in *Heaven's Vault*. Each new interpretation opens up fresh avenues of understanding, and besides that, it's just really fun to collect words, see your vocabulary grow and become able to put longer and longer sentences together. Language is, of course, a means of communication, first and foremost. Nothing you find has been left specifically for your benefit, but thanks to the power of language, Aliyah is able to piece together stories of those long dead, thus forming a connection with them. At one point she comes across a house in which the writing, however sparse, allows her to understand how its inhabitants might have lived – it's these glimpses, these puzzle pieces of people's lives and societies, that give such depth to *Heaven's Vault*.

Heaven's Vault takes place both in the ruins of old and the world of the living; as such, talking to other people is another vital aspect of the game. Clues gathered at sites can tell you where to go next, but often it's highly encouraged to talk your findings over with the people you meet around the Nebula, be it colleagues, old friends, or even strangers who take an interest in your work. Depending on what you let them know, new avenues can open up. Talking also lets you learn a lot more about the social structure of *Vault*'s diverse world, which is marked by a deep class divide just like ours. The villages and cities you visit all have a distinct look; the planet of Elboreth, for example, highlights the contrast between its wealthy district, with high sandstone walls, and the precariously stacked-together shacks of the slum. There's history to soak in even in the 'modern' regions of the game.

"Aliya pieces together stories of those long dead"

Negatives are few and picky. The 3D landscapes pose a nice contrast to the 2D character design, but moving around could be smoother – the fact that character models have two frames and are only visible to the knee while moving gives them an eerie effect, and sometimes the camera makes discoveries difficult.

But these are minor complaints when you look at *Vault*'s sheer scale; its whole timeline is filled with events both past and present, and wealth of language fragments so big you will want to keep building your vocabulary beyond a single playthrough into the New Game Plus mode. Aliyah and Six can talk about anything and everything, if you choose to, and with everything discovered, you build the picture of a living, breathing world – one you just want to see more of.



The timeline updates regularly with your findings.

VERDICT

It's all talk, which is exactly what you want – intricately built worlds and systems to put language to use in and discover copious secrets with.

89%

The swishy sword effects look quite nice, so at least we have those to enjoy.

HIGHLIGHT

Although it has to include Jason and Tommy for the fans, the roster does a good job of including less popular characters from the *Power Rangers* canon. Kat Manx represents *S.P.D.*, Gia Moran for *Super Megaforce*, and even the comics are included with the Ranger Slayer. More of that, please.

Zordon? What have they done to you?

Review

Power Rangers: Battle for the Grid

Oh no, Power Rangers

If you thought the 2017 *Power Rangers* film was bad, just wait until you see *Power Rangers: Battle for the Grid*. It's a three-on-three, combo-heavy 2D fighter that borrows a lot of elements from games like *Marvel vs. Capcom* and *Skullgirls*. Taking down all three members of the other team requires you to tag out with your partners, rack up massive combos, build your meter and pull off special attacks, and, if you've really taken a pounding, summon your Megazord to tear up the arena.

That may sound exciting, but the sensation lasts as long as a *Power Rangers* baddy-of-the-week. None of the characters has even a fifth of the depth of a *Skullgirls* fighter, with only a couple expecting directional inputs (I haven't seen a single quarter-circle so far), and the rest being mostly the same in how they play. While it's good to be welcoming to fighting game newcomers, it's almost patronising to have a tutorial mode when things are this basic.

At launch, there are nine characters, with three more planned as a free download sometime after launch. On the plus side, they span the whole of the *Rangers* canon, including comics and later series nostalgic *Mighty Morphin* fans might not have encountered before; on the downside, having to pick three out of this roster of nine means you're almost guaranteed to be fighting the same characters over and over in consecutive fights.

The stages are few, and those included are bland and poorly realised, with zero dynamism and a soundtrack that mostly consists of dead silence. It's almost impressive how Zordon looks worse than he did in the show almost 30 years ago, and yet he's the focal point of one of the most prominent stages.

If you were at least hoping for enough single-player content to make up for the inevitable absence of an online player base, you may be sorely disappointed. Beyond ranked and unranked online play, a tutorial and a standard practice mode, there's a disgustingly basic arcade mode where you, in essence, fight the same couple of teams over and over until poorly written dialogue boxes pop up and you fight a final boss... who was spoiled on the character select screen. That's it, credits roll, we're done here.

Power Rangers: Battle for the Grid could've been a fighting game event bigger than *Marvel vs. Capcom*. Instead, it's an insulting, bare-bones, minimally engaging, buggy mess of a game that should have been sent back to the drawing board a long, long time before it ever got onto platforms. Just go play *Skullgirls* and sing "go go Power Rangers!" while you're at it. You'll have more fun.

Info

GENRE
Fighting

FORMAT
Switch (tested) / PS4 / XBO / PC

DEVELOPER
nWay

PUBLISHER
nWay / Lionsgate Games

PRICE
£19.99

RELEASE
Console versions out now, PC version TBC

REVIEWED BY
Joe Parlock

VERDICT

Not even a property with as much history as *Power Rangers* could hold together this mess.

32%

Better (25 years) late than never

DICE's lost Amiga/Mega Drive shooter Ultracore is finally here

The vaguest of memories, of seeing a game called *Hardcore* from the studio that made *Pinball Dreams*, of wondering if I could be bothered playing anything other than *Championship Manager* on the Amiga at that point, of wishing I hadn't sold the Mega Drive to put into the PlayStation savings pot. It's weird how much wistfulness the renamed *Ultracore* conjured up on loading it up for the first time, but then this is a game I hardly remember which I never thought would get a proper release. Now, thanks to the team at Strictly Limited Games, it has.

Analogue's Mega SG, the FPGA-based Mega Drive clone, landed in March and immediately made thousands of us around the world avid players of a Sega(-alike) console once more. Included as standard with the machine is *Ultracore*, as well as an overwhelming feeling of chunky 1990s shooty-blast nostalgia. It's not quite up there with the *Contra: Hard Corps* of the world, but by crikey, if DICE's previously lost game isn't a welcome, dare I say it, blast from the past.

It looks about as Amiga as they come, all rounded and metallic, with a pulsing, 'soft' synth soundtrack backing up the action. In that way, it shows it's not an imitation: it is a mid-1990s game, just one arriving 25 years late and with the minimum of fuss. And the action doubles down on that fact, offering a traditionally European take on the run-'n'-gun genre, all multiple paths back and forth, storylines to follow, keys to obtain, and super-hard difficulty to come to terms with.

Ultracore is, in short, a curio. A fun one. It's intensely satisfying in short bursts, and short bursts are exactly how I've found myself playing it recently. Had it arrived in 1994, I likely wouldn't have ended up paying it any attention – there was so very much else around then to compete with, and I can actually see why the plug was pulled on the project. And it's not like it did DICE much damage.

But arriving now in 2019, *Ultracore* manages to get a new lease on life, and to stand out from gaming's current crowd. When it comes to PS4 and PS Vita, which it might already have done when you read this, I do hope people give it a chance.

"It looks about as Amiga as they come, all rounded and metallic"

There are sure to be dozens, hundreds – maybe thousands – of games over the years that have dropped by the wayside at various points. Maybe they didn't get past the planning stage; perhaps they were just a vague thought scrawled on a napkin, lost forever after a spillage and the need to wipe it up with something. Maybe they were even more finished than the '99% done' *Hardcore* was back in 1994 when Psygnosis, its original publisher, pulled the plug.

There's so much out there that never got a chance, that we'll never see, and that will remain a vague memory; a complete unknown; an unfulfilled desire. *Ultracore* is a fun distraction and something I'll stick with for a while, for sure, but it's so much more than that – it manages to represent the hope that we all have to see what we never thought we would, and to play what we never thought we could. What I'm saying is: can Strictly Limited Games pick up on *Mega-Lo-Mania 2* next, please.

Wireframe Recommends

Contra: Hard Corps

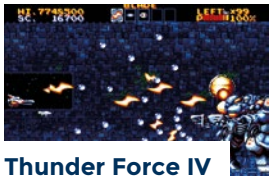
MEGA DRIVE

Just don't play the US/EU versions, given they're ludicrously hard and – in the European version – turn it into that *Probotector* nonsense. Anyway, this is one of *Contra*'s finest hours, and one of the Mega Drive's best run-and-gunners.

Mega Turrican

MEGA DRIVE

Sega's console wasn't short of Euro-shooters, and *Mega Turrican* proved there was a market for well-made, surprisingly smart shooters from the region. Plus, it's slightly better than the Amiga port, renamed *Turrican III*.



Thunder Force IV

MEGA DRIVE

Alright, so it's not running and gunning, but I'm not going to recommend shooters on the Mega Drive without talking about *Thunder Force IV*. It is, in short, magnificent. Still. Promise.

Bionic Commando

Less famous than its NES successor, the Bionic Commando coin-op was the original king of the swingers

CAPCOM / 1987 / ARCADE

Whoever heard of a platform game without a jump function? That, at least, might be what players were thinking when they first encountered Capcom's *Bionic Commando* way back in 1987. And yet, as is so often the case in video games, what initially seems counterintuitive about *Bionic Commando* soon feels absolutely natural. In place of mighty leaps, *Bionic Commando*'s hero, Super Joe (apparently the same protagonist as Capcom's 1985 hit, *Commando*, according to its American marketing) has a telescopic arm, which he can variously use to swing across gaps, pull himself up to walkways above him, or simply launch into an opponent's face, temporarily stunning them. It's an idea that completely changes the tempo of what might have been a conventional run-and-gun platformer.

As we saw in our developer profile in Wireframe issue seven, legendary game designer Tokuro Fujiwara came up with a similar concept while he was at Konami, but its original deployment – in the lesser-known arcade game for that company, *Roc'n Rope* – felt less satisfying than it does in *Bionic Commando*, which he made a few years after he decamped to Capcom in 1983. Where the grappling hook mechanic in *Roc'n Rope* felt wooden and ponderous, with the central character climbing hand-over-hand between platforms at an agonising pace, *Bionic Commando* is swift and liberating. Super Joe's gizmo allows him to manoeuvre across the screen at remarkable speed, and swinging from one platform to the next, then reeling in the grappling arm and quickly detaching it to land just so, is an ability that both looks and feels impossibly cool. (There's also a certain thrill to learning that the telescopic arm can be used to pluck falling power-ups from the air and drag them towards you.)

Actually sharpening your skills to the point where you can pull off these superhuman feats takes considerable practice, and in arcades, that meant pumping in entire pockets full of spare change. This might partly explain why it was the NES incarnation of *Bionic Commando*, with its very different visuals and level layouts, that became more widely remembered, at least in North America; certainly, it was this version that was remade by Swedish developer Grin in 2008, and not the arcade original.

"Whoever heard of a platform game without a jump function?"

This makes the earlier *Bionic Commando* something of a forgotten masterpiece, at least in our estimation: it's certainly faster, more colourful, and more intense than the NES version, even if it doesn't have the same level of depth or variety. Besides, it was in this game that Fujiwara's bionic arm concept was first perfected: a concept that eventually found its way into dozens of other games in one form or another, whether it was Rico Rodriguez's Grappler doodad in the *Just Cause* series, Radical Entertainment's *Prototype*, or Grin's somewhat rickety attempt at a 3D *Bionic Commando* reboot, released in 2009. Later incarnations were great games in their own right, but it was the swinging action of *Bionic Commando*'s arcade edition that first had us hooked.

Next Issue

ON SALE 9 MAY

FELIX THE REAPER

Looking ahead to a spectacular, puzzling danse macabre

Also

- The reality of life as a solo developer
- How detective games are evolving with the times
- Create a teleport ability in Unity
- Afrofuturist sci-fi in We Are The Caretakers

Don't miss out! Subscribe PG54

Editorial

Editor
Ryan Lambie
Email ryan.lambie@raspberrypi.org

Features Editor
Ian Dransfield
Email ian.dransfield@raspberrypi.org

Sub Editors
David Higgs & Vel Ilic

Design

criticalmedia.co.uk

Head of Design
Lee Allen

Designer
Harriet Knight

Contributors

Diego Arguello, Rik Cross, Mel Croucher, Konstantinos Dimopoulos, Malindy Hetfeld, Dan Lambton-Howard, Steve McNeil, Holly Nielsen, Ed Nightingale, Jordan Oloman, Joe Parlock, Nic Reuben, Reid Schneider, Rhett Thompson, Brittany Vincent, Nick Walton, Alan Wen

Publishing

Publishing Director
Russell Barnes
Email russell@raspberrypi.org

Director of Communications
Liz Upton

CEO
Eben Upton

Advertising

Commercial Manager
Cha Milligan
Email charlotte.milligan@raspberrypi.org
Tel +44 (0)7725 368887

Distribution

Seymour Distribution Ltd
2 East Poultry Ave, London EC1A 9PT
Tel +44 (0)207 429 4000

Subscriptions

Unit 6, The Enterprise Centre, Kelvin Lane, Manor Royal, Crawley, West Sussex, RH10 9PE

To subscribe
Call 01293 312192 or visit wfmag.cc/subscribe

Subscription queries
wireframe@subscriptionhelpline.co.uk

recycle

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Wireframe magazine is published by Raspberry Pi (Trading) Ltd, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised in the magazine.

ISSN: 2631-6722 (print), 2631-6730 (online).

G-MASTER™
#MONITORS4GAMERS
by iiyama

UNLOCK YOUR GAME

gmaster.iiyama.com

1ms
144Hz
Free Sync

JOIN THE PRO SQUAD

RED EAGLE™

GB260HSU | GB270HSU | GB2760QSU

1ms
75Hz
Free Sync

GET IN THE GAME

BLACK HAWK™

GE2288HS | G2530HSU | G2730HSU | GB2530HSU | GB2730HSU

4K
Free Sync

IMMERSE YOURSELF IN THE GAME

GOLD PHOENIX™

GB2888UHSU

2560 x 1440
1ms
75Hz
Free Sync

ENTER A NEW DIMENSION

SILVER CROW™

GB2783QSU