



**Santa Monica** Studio



# TestMonkey: Automated Testing at Santa Monica Studio

Ben Hines



Hello, welcome to TestMonkey: Automated Testing at Santa Monica Studio

Before we start, Please mute your devices, and make sure to submit feedback at the end of the talk.



I am Ben Hines, I'm a DevOps slash Infrastructure slash Tools Engineer.

I have about 23 years in the industry. I started in QA then moved to Build Engineering and Infrastructure / Tools roles at Vivendi Games, Disney, Zynga, EA, Santa Monica Studio

Some favorite titles i've shipped

- Simpsons Hit and Run
- Empire Earth II
- World of Cars
- God of War 2018
- God of War Ragnarok



This is a talk about what we've done with automated testing.  
I will include some ideas about implementing these systems in general, and issues we have encountered.





Automated testing is has not always been common in the game industry. But, it has a long history at Santa Monica Studio for over 20 years.  
At SMS, testing and hearing from the test systems is well ingrained into the team's workflow.

# TestMonkey

- Web Based Reporting
- Can run Local/Remote
- Also run by CI





Santa Monica Studio

The system we came up with is called: TestMonkey!

Its Key Features include...

- Web based reporting
- It can run local Local Or Remote -- A developer can run all the tests using a local command or Build GUI.
- Once they submit, the same tests are run again on by Jenkins server and results reported.

## What Tests Do We Run?

- Data Build Tests
- Smoke Tests
- Save Game Tests
- Visual Tests
- UBSan / ASan Tests
- DART Tests (GamePlay, Cinematic capture)



We run various suites in testmonkey.

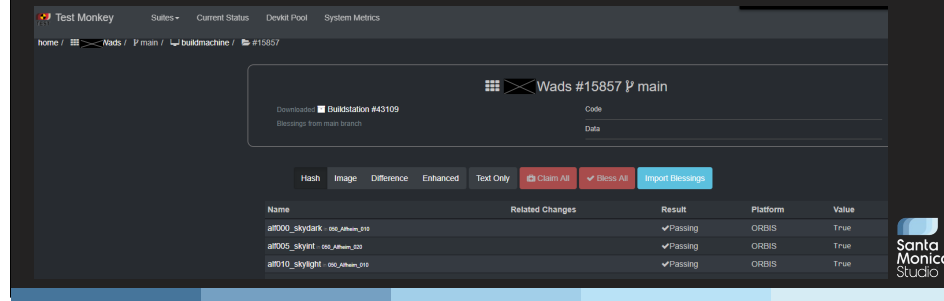
- Data Build Tests
- Smoke Tests
- Save Game Tests
- Visual Tests
- UBSan / Asan Tests
- DART Tests

I'll go into more detail on each of these types of tests and how we use them.

## What Tests Do We Run?

### Data Builders – aka “WAD Monkeys”

- Implemented as ‘tests’
- Also creating binary builds of game data





The first ‘tests’ we run is the actual data build of the game. We call these “Wad Monkeys”

- The term wad is sort of our level unit
- Wad Monkeys are implemented as ‘tests’
- But they are also the things creating the binary builds of the game data.
- If the wad monkeys have a failure, this means that particular level will also fail to build for a designer or artist locally, so it’s important to be fixed quickly.
- Once they finish, the results are published for the team to download as a ‘binary’ build of the game data. These binary builds can be used to speed up development (so a programmer won’t need to build locally)

## What Tests Do We Run?

### Smoke Tests

- Run after Data Build
- Runtime basic launch tests
- Simple Console Output Check

Name	Related Changes	Result	Platform	Value
04_Foothills_010_Start - Foothills	 Samuel Stenkiar	 Failing	ORBIS	<code>(*(uint8_t*)ptr == static_cast&lt;uint8_t&gt;(PTR_NEW_POISON_HIGH))</code> allocation was not properly poisoned
01_Forest_010_HuntStarts - Forest		✓ Passing	ORBIS	True



### What are smoke tests?

Once the Wad Monkeys have a binary build of the data, the smoke tests actually run each level.

Basic ‘does this level launch’ test.

- This knows when to stop by doing a simple check of the console output of the game.
- This will catch any crashes which only occur in a particular area.
- Here’s an example of a failing and passing smoke test – can see it has an Assert message there.

## What Tests Do We Run?

### Smoke Tests – Issues We've Hit

- Progression
- Spawn Points



There are some gotchas / Issues We've Hit with smoke tests.

A little game engine specific, but..

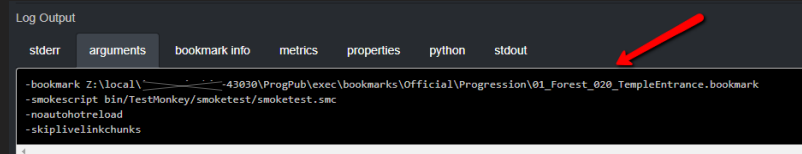
Progression: One tricky thing here is handling progression. Found we couldn't launch directly into each level since you may not necessarily be at proper progression / game state.

Spawn Points: Additionally, each WAD may not have a spawn point at all for the main character.

## What Tests Do We Run?

### Smoke Tests – Issues *Solution*

- Launch into saved progression 'bookmarks'



```
Log Output
stderr arguments bookmark info metrics properties python stdout
-bookmark Z:\local\43030\ProgPub\exec\bookmarks\Official\Progression\01_Forest_020_TempleEntrance.bookmark
-smokescript bin/TestMonkey/smoketest/smoketest.smc
-noautohotreload
-skiplivelinkchunks
```

- Make sure any section to be tested has a spawn point.



### Solution:

To solve both of these, we launch into 'bookmarks' or 'snapshots' created by QA at within each area we want to test.

Additionally, made sure each area has a spawn point.

This works, but now run into 'out of date bookmarks'.

So we gave QA a tool to easily create bookmarks as they play through the game on a daily basis.

## What Tests Do We Run?

### Save Game Tests

- Separate from Smoke Tests
- Late in Development – Broken “Saved Games”
- Test suite was created to solve problem

DiscCritical_Path01_Forest Disc	✓Passing	ORBIS	True
DiscCritical_Path02_Riverpass Disc	✓Passing	ORBIS	True
DiscCritical_Path03_earn to Realm Travel Disc	✓Passing	ORBIS	True
DiscCritical_Path09_PeaksPass Disc	✓Passing	ORBIS	True



### Save Game Tests

- These are actually separate from the smoketests which test each level.
- Late in development of God of War 2018, we encountered issues with development breaking the game’s ability to load retail “Save Games”.
- To solve this, we created a new test monkey suite to load Save Games
- (Can see screenshot here) – save games created in various areas.
- Ideally Saved Games would be the same system as used for the smoke test bookmarks, but that is a problem for another day..



## What Tests Do We Run?

### Visual Tests

- Independent from testing the game content
- Generally, test engine code only
- Takes screenshots of the game
- Always deterministic
- Run after every commit, users 'bless' the results when they change
- Run on 4 platforms – PC, PS4, PS4 Pro, PS5



### Visual Tests

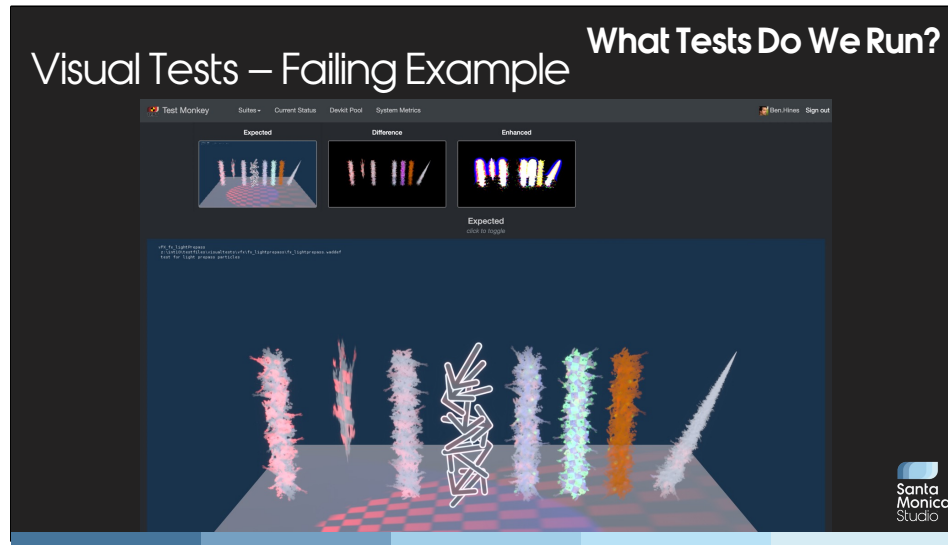
- Visual Tests are independent from testing the game content
- These generally test the code (engine) only
- **Test screenshots of the game. If anything changes from previous run, it fails**
- **So these tests must be deterministic – exactly the same results each time. They have their own data separate from the game. (no Kratos here) Even a 1 pixel change will fail the test**
- **The tests are run after every code change and the users 'bless' the results if they change purposefully.**
- **Run on 4 platforms, PC, PS4, PS4 Pro and PS5**



Here is a passing example of Visual Tests

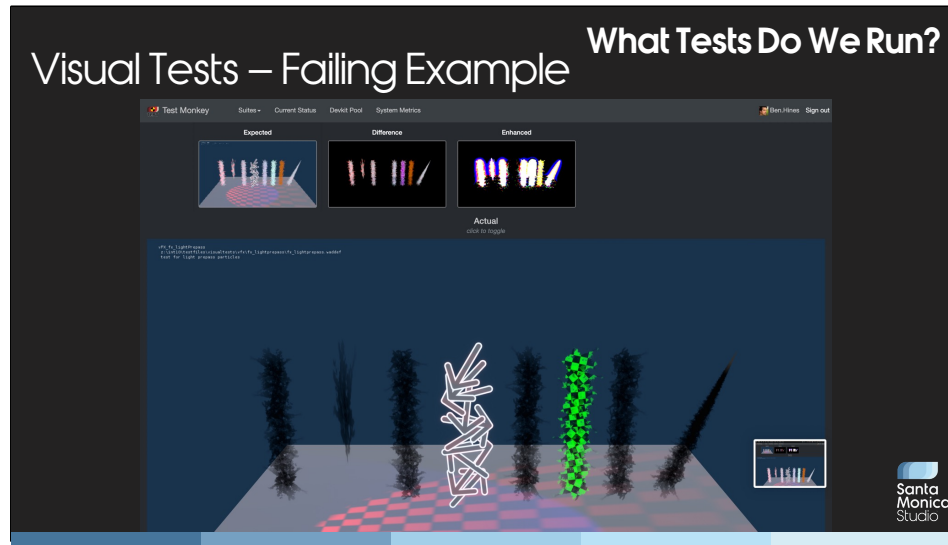
These are used to test graphical features, but can actually be used to test just about any feature of the game.

For example, put an object in the scene and trigger the script that should rotate or move that object to the left, then verify that the object actually moved to the left.



Here's a failing example. You can see the expected on the left, difference and enhanced difference. Mousing over the image lets you zoom in for details.

The enhanced view is often useful because the differences can be subtle. In this case we broke some lighting rendering. The user can click the image on the webpage in order to view the difference. (toggling back and forth)



Here's a failing example. You can see the expected on the left, difference and enhanced difference. Mousing over the image lets you zoom in for details.

The enhanced view is often useful because the differences can be subtle. In this case we broke some lighting rendering. The user can click the image on the webpage in order to view the difference. (toggling back and forth)

## What Tests Do We Run?

### Visual Tests – Issues We've Hit

- All must run on same OS
- Use Software Rendering to ensure determinism
- A single pixel change can cause a failure



### Gotchas / Issues We've Hit:

On PC, they all must run on the same OS and spec to ensure everything renders the same. Fortunately we've standardized and use WARP mode rather than hardware rendering to ensure determinism.

Even a single pixel change can cause a failure. We may or may not care. This sounds like it may be a large problem but it actually is no really significant issue with our engine. The rendering team really appreciates knowing when anything changes.

## What Tests Do We Run?

### Visual Tests – Future Improvements

- Too Many Tests: Consolidate many tests into one
- If a test fails, rerun it on other infrastructure



### Potential Improvements

- Too Many Tests: Currently we have thousands of tests that are running every single commit. Each testing an individual feature of rendering.
- Those could be combined to fewer tests in a few “Kitchen Sink” screen shots
- If, for say, the kitchen sink test, fails, we could then run the individual tests.
- Re-run failing tests to ensure not infrastructure issue.

## What Tests Do We Run?

### ASan / UBSan Tests

- Clang low level code validations

ASan  
C/C++ Address  
Sanitizer

UBSan  
C/C++ Undefined Behavior  
Sanitizer

- Great for catching \*potential\* crashes
- Game Runs 50% slower in this mode



### ASan / UBSan Tests

Similar to smoke tests, but these run the game in ASan and UBSan mode. These are low level C++ code validations implemented by Clang which is the compiler used by the game.

- ASan is short for 'Address Sanitizer'.
- UBSan is short for 'Undefined Behavior Sanitizer'
- These are great for catching \*potential\* crashes by finding cases where the code could potentially make an unbounded memory access or doing something undefined in C++. This will in theory find some crashes before they happen.
- The game runs much slower in this mode as it does all the extra validation.

# Executing Tests

- Simple Custom Scripting Language

```
14 print(0,0,__TEST_NAME__)
15 print(__PATH__)
16 print(5,2,"testmonkeylighting env preset. disabled ambient light and env map. ")
17
18 vfsInt("/System/Video/Lighting/Enable Ambient",0)
19 vfsInt("/System/Video/Lighting/Enable Environment Map",0)
20 screenshotFromCamera(__FILE_NAME__+"_SS", "testCam", 1)
21 log("Finished running: "+__TEST_NAME__)
22
```



So, How Does TestMonkey actually run the tests ?

Not enough time to go into this in too much depth, but here's a small sample:

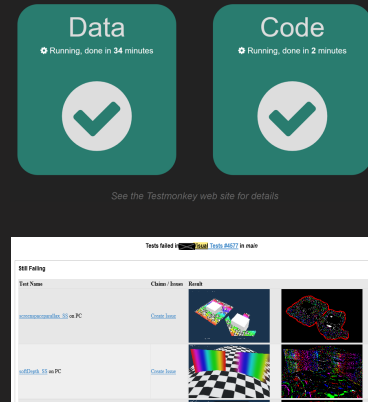
- All tests are powered by a basic scripting engine used only for testmonkey. Simply launch the game with a path to a small level with the script. This script can call C++ functions.
- Here is an example, can see setting some rendering options there as well as taking a screenshot for Visual Tests.

This system is lower level than the main game scripting engines and has few dependencies. But it could call into those other systems for testing purposes.



# Reporting Results

- TestMonkey website
- Dashboard on TVs displayed around studio
- Emailed results
- Instant Messaging



## Reporting Results

We report results in various ways

- TestMonkey Website

The site has a list of test runs with Code and Data changes for each, It also lists potential culprits next to each test

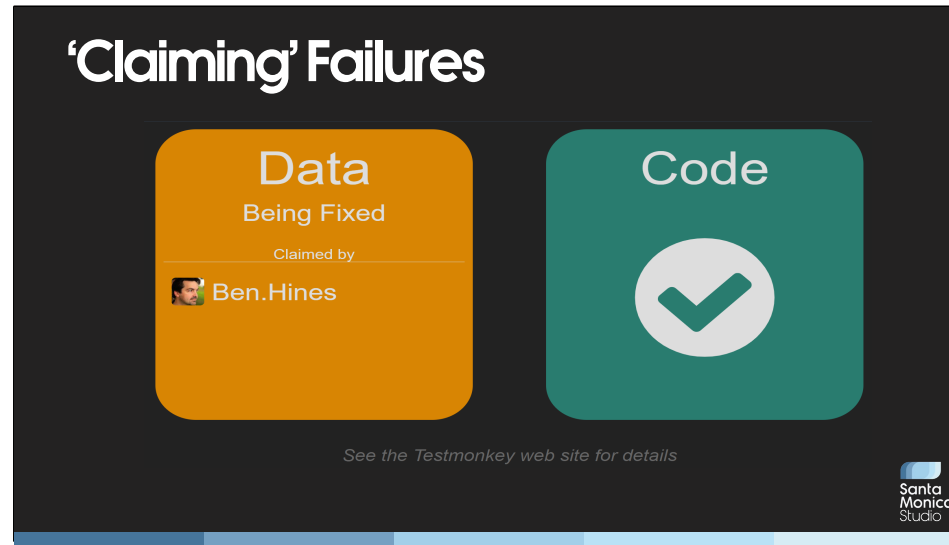
- Dashboard on TVs on the wall of the studio --

Lists the two most important test suites, displays to the team. The “potential” culprits who are responsible - mentioned on the TV

- Emailed results

- Instant Message

We have experimental Slack notifications as well though this isn't fully rolled out yet.



One nice feature is that users can claim failures.  
Say you have a level crashing and you're working on a fix.  
Just click 'Claim' on the site and everyone will know it's being looked at.

One challenge is getting people to actually claim their failures – often QA ends up asking folks to go into the website and claim something. We don't allow marking tests as claimed for someone else so there is some pain here. A nice improvement would be if a user could really simply claim something via slack or similar.

# Blocking Check-ins

- Initial: Tests Broke? Block checkins!
- Loosened up process
- Continued improvement

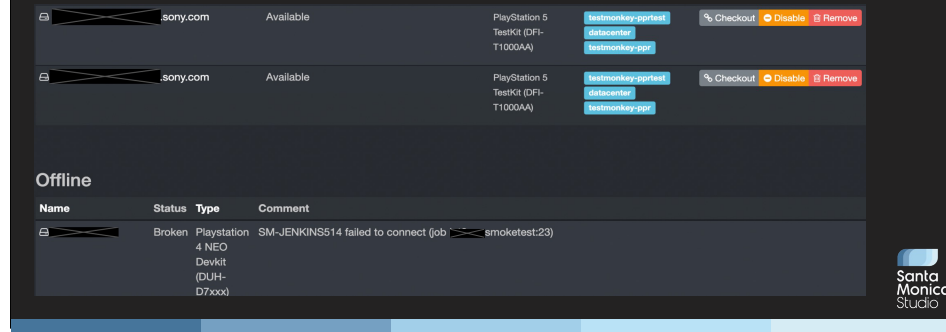


## Blocking Checkins

- Originally, failures in either of the two 'important' suites actually blocked folks checking in data or code.
- As we moved from 15 programmers to many more, blocking code check ins resulted in decreased productivity – sometimes pushing unrelated fixes to the next day.
- We moved away from this as the team got much larger, it ended up slowing things down a bit too much.
- Occasionally does cause a problem if someone checks in more changes on top of changes which broke the build and we want to revert. Could have to revert several layers of changes. Fortunately, this is relatively rare and I don't think we've seen much impact from allowing people to checkin when tests are failing.

# TestMonkey – General ‘Gotchas’

- Infrastructure failures
- Devkit management



## General Gotchas We've Run Into With TestMonkey

– Infrastructure Failures: TestMonkey runs its tests on a 'farm' of PlayStation Development and Test Kits. These kits aren't always perfectly reliable.

If a kit gives a strange error, it may be from the game crashing, or it could be because the devkit is in a weird state.

To solve this, we have a list of the most common 'kit related' errors and ignore those in the system.

– Another thing which has helped this is to run a 'devkit tester' elf when each kit is 'checked out' from the system. If the devkit tester application fails to run, the kit is not used and marked bad.

Gotcha with this system: Since devkits are managed from a Windows system, the problem could be on the Windows server rather than with the kit itself. Rarely, this can cause cascading failures where every kit gets disabled.

Not a perfect solution for this other than catching every possible error talking to a devkit.

Potential further improvements to devkit mgmt:

Run devkit tester before each test

or Reboot the kit after any test failure (or potentially after every test)

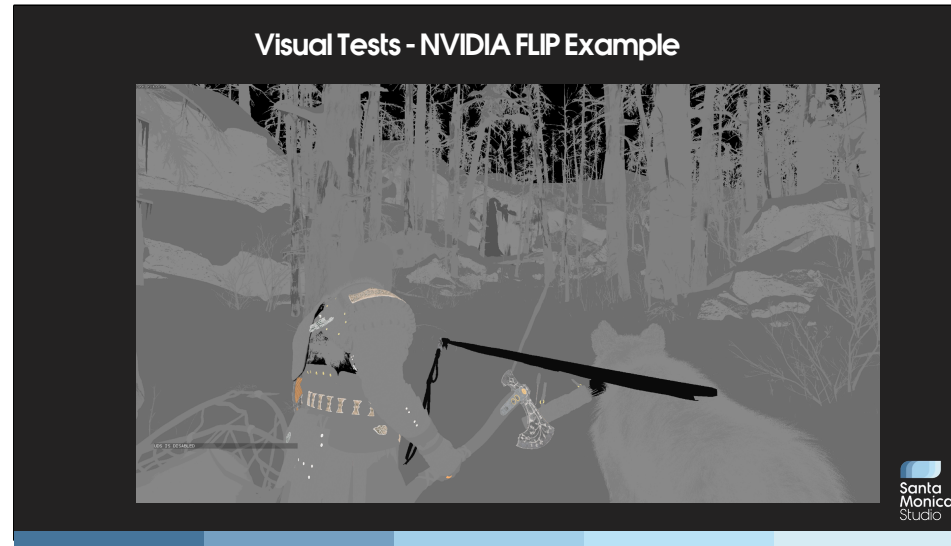
## God of War Ragnarök Improvements

- JIRA Linking!
- Moved to Python 3 / AWS
- Problem - Can't fit on disc!
- Looked at higher texture compression
- Much more deterministic in-game
- NVIDIA FLIP / In-Game Visual Tests



For Ragnarok, we made some improvements to the systems discussed earlier.

- We got jira linking into the UI
- Moved to python 3 / aws
- We were expecting that we might not be able to fit on on disc, or even on two discs.
- In support of this, we started looking at different texture compression methods.
- In order to validate that using different texture compression didn't hurt visual quality, we spent a while getting many more systems within the game to be fully deterministic.
- This allows us to now run Visual Tests within the game levels themselves. Then, rather than using dumb per-pixel diffs, we process the diff from the previous run using Nvidia Flip. This is a tool which produces a map that approximates the difference perceived by humans when alternating between two images.



Here is a NVIDIA FLIP output example

The reason it looks weird is we ended up separating the output by rendering pass. This particular screenshot is showing nvidia flip analysis in the reflectance pass, showing a human-visible difference in those particular reflective materials.

## Future Directions

- Smarter Test Running
- Better parallelism
- More DART tests
- Test More Things

```
using pad input with timing
pad.right(); // press right
time.waitFor(1); // do nothing for 1 second.
pad.right(2); // press right twice
time.waitFor(1);

script.saveImage();

// press left twice, hold button down for 0.1 seconds and released for 1.9 seconds with each press.
pad.left(2, 0.1, 1.9);
```

Santa  
Monica  
Studio

### Future Directions

#### Smarter Test Running

- TestMonkey currently runs the tests in a semi random order. Ideally we would like to run the tests which failed last time, first.
- Better parallelism - We run many testmonkey tests from the same host at once. This has sped things up, but quickly running out of kits to test on. Potentially allow users to 'donate' their devkit to TestMonkey when it's not being used by them. Would then need to rerun any failed test on a 'user' kit to make sure it wasn't from them messing with the kit.
- DART Tests
  - DART is an internal Sony tool. It runs more complicated tests involving specific controller input. We started using these to a limited degree on Ragnarok.
- Test More Things
  - We'd like to test the tools or do more unit testing of various game features.
  - We did add a 'csv' / 'logtext' mode of testmonkey which can compare output of a PC tool.

## What does this all mean?

- Scaling The Team
- Rendering - huge fans of Visual Tests
- Balance between test coverage and resources?



Summary – what is the point of all this?

- Over the years we've found this really helps us keep the game stable in development.
- Rendering team wanted me to call out that this is so core to their workflow. The visual tests really help them find stuff all the time.
- How do we balance test coverage and resource use? TestMonkey has had less than half an engineer on it for many years, so our cost is mostly devkits and a few servers. The answer is likely relative to a studio's priorities. For SMS, the value in providing timely breakage information to the team is worth the cost and time investment.



# Thank You!

**Ben Hines**  
Staff DevOps Engineer  
ben.hines@sony.com  
@benh57 



And that's it! Let me know if have any questions.


Santa Monica Studio

Our journey  
Your story

We're hiring for what's next!

We're expanding our family across disciplines and would love to meet you. Please visit [sms.playstation.com/careers](https://sms.playstation.com/careers) for all openings or drop us a line at [sms.recruiting@sony.com](mailto:sms.recruiting@sony.com)

 @santamonicastudio

 @SonySantaMonica

 @santamonicastudio

94 GOD OF WAR RAGNARÖK 94

GOD OF WAR RAGNARÖK IS A TRIUMPH  
GAMING REVOLUTION

IMPROVES ON ITS PREDECESSOR IN EVERY WAY  
WASHINGTON POST

★★★★★

★★★★★

VIDEOGAMER

★★★★★

GMAGAZINE

★★★★★

IGN

★★★★★

THE INDEPENDENT

★★★★★

EASY ALLIES

10|10

IGN

10|10

TRUSTED REVIEWS

10|10

GAMESPEW

10|10

THE INDEPENDENT

10|10

EASY ALLIES

10|10  
GAMINGNEXUS

10|10  
PSU

10|10  
SILICONERA

10|10  
GAMEBLOG

10|10  
VG247

10|10  
GAMEPUR


















10|10  
EGM

9|10  
GAMESPOT

JOIN US AT GDC 2023



BUILD YOUR GOD OF WAR GDC AT:  
SCHEDULE.GDCONF.COM

	<b>BRUNO VELAZQUEZ</b> • ANIMATION DIRECTOR DAVID GIBSON • ANIMATION DIRECTOR ERICA PINTO • LEAD NARRATIVE ANIMATOR MENBIL YSSEF • LEAD GAMEPLAY ANIMATOR Animation in 'God of War Ragnarök' • Animation Summit MONDAY, MARCH 20 • 9:00AM - 10:00AM • ROOM 303, SOUTH HALL		<b>STEPHEN MCAULEY</b> • LEAD RENDERING PROGRAMMER Rendering 'God of War Ragnarök' • Programming WEDNESDAY, MARCH 22 • 3:00PM - 3:30PM • ROOM 303, SOUTH HALL
	<b>SUE PACETE</b> • SR USER RESEARCHER Playtesting 'God of War Ragnarök' Accessibility Options • UX Summit MONDAY, MARCH 20 • 1:00PM - 1:30PM • ROOM 2001, WEST HALL		<b>ERIC GOTTESMAN</b> • SR STAFF DEVOPS ENGINEER Modernizing multiplayer services for 'God of War: Ascension' (PS3) • Production & Team Leadership • Presented by Amazon Web Services WEDNESDAY, MARCH 22 • 2:00PM - 3:00PM • IDC PARTNER STAGE, EXPO FLOOR, NORTH HALL
	<b>PAOLO SURRICCHIO</b> • SR STAFF PROGRAMMER Reinventing the Wheel for Snow Rendering • Advanced Graphics Summit MONDAY, MARCH 20 • 2:00PM - 2:30PM • ROOM 303, SOUTH HALL		<b>SAM STERNKLAR</b> • SR PROGRAMMER 'God of War Ragnarök' Visual Scripting Solution • Programming THURSDAY, MARCH 23 • 10:00AM - 11:00AM • ROOM 2006, WEST HALL
	<b>BEN HINES</b> • SR STAFF DEVOPS ENGINEER Automated Testing at Santa Monica Studio • Tools Summit MONDAY, MARCH 20 • 4:40PM - 5:10PM • ROOM 3004, WEST HALL		<b>ADAM OLIVER</b> • SR COMBAT DESIGNER Breaking Barriers: Combat Accessibility in 'God of War Ragnarök' • Design THURSDAY, MARCH 23 • 2:00PM - 2:30PM • ROOM 2002, WEST HALL
	<b>XUANYI ZHOU</b> • PROGRAMMER Realtime Neural Texture Upsampling in 'God of War Ragnarök' • Machine Learning Summit TUESDAY, MARCH 21 • 2:10PM - 2:40PM • ROOM 2010, WEST HALL		<b>GÖKSU UĞUR</b> • AI LEAD A Practical Toolkit for Transitioning into Leadership Roles • Leadership THURSDAY, MARCH 23 • 2:00PM - 2:30PM • ROOM 303, SOUTH HALL
	<b>ETHAN AYER</b> • SR ENVIRONMENT ARTIST The Art of Making Vistas • AI Summit TUESDAY, MARCH 21 • 3:00PM - 3:30PM • ROOM 3007, WEST HALL		<b>ZACH BONN</b> • SR STAFF TECHNICAL UI DESIGNER 'God of War Ragnarök' Building the UI For a AAA Title • Design THURSDAY, MARCH 23 • 4:00PM - 5:00PM • ROOM 303, SOUTH HALL
	<b>GÖKSU UĞUR</b> • AI LEAD Preparing AI Systems For God of War Ragnarök • Programming WEDNESDAY, MARCH 22 • 9:00AM - 10:00AM • ROOM 303, SOUTH HALL		<b>SALAAR KOHARI</b> • PROGRAMMER Companion Traversal in 'God of War Ragnarök' • Programming FRIDAY, MARCH 24 • 10:00AM - 11:00AM • ROOM 2007, WEST HALL
	<b>VICKI SMITH</b> • SR STAFF LEVEL DESIGNER The Final Battle of God of War Ragnarök: Techniques For Delivering High States Sequences • Design WEDNESDAY, MARCH 22 • 10:30AM - 11:00AM • ROOM 2002, WEST HALL		<b>TENGHAO WANG</b> • SR PROGRAMMER Joint Based Skin Deformation in 'God of War Ragnarök' • Programming FRIDAY, MARCH 24 • 1:30PM - 2:30PM • ROOM 2006, WEST HALL
			<b>HARLEIGH AWNER</b> • TECHNICAL NARRATIVE DESIGNER How to Build a Home: Designing Narrative For Sindri's House in 'God of War Ragnarök' • Design FRIDAY, MARCH 24 • 3:00PM - 3:30PM • ROOM 2001, WEST HALL

Santa Monica Studio **GDC**