AUTOMATED TESTING

USING AI CONTROLLED PLAYERS TO TEST THE DIVISION

JOSE PAREDES
SENIOR TECHNICAL DIRECTOR, UBISOFT REFLECTIONS

PETE JONES
LEAD GAMEPLAY PROGRAMMER, UBISOFT REFLECTIONS

GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

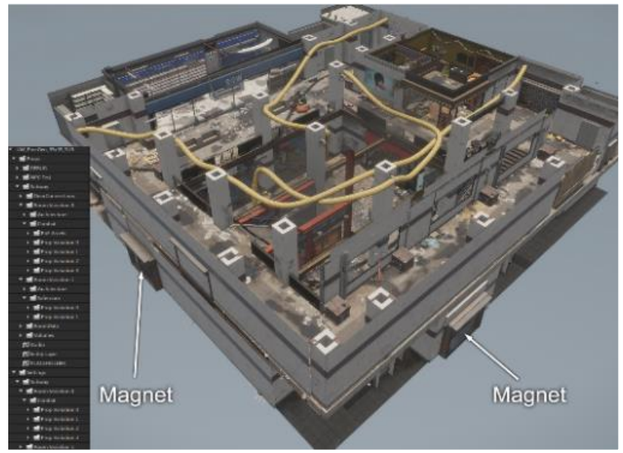
AGENDA

- MOTIVATION
- SYSTEM DESIGN
- IMPLEMENTATION
  - DIVISION 1 & 2
- PERFORMANCE TRACKING
- NEXT STEPS
- CONCLUSION

PART 1 - MOTIVATION

THE DIVISION UNDERGROUND DLC

- Dungeon crawler.
- Procedurally generated.
- 8x5 km2 / 256 dungeons / 100 hours.



THE TESTING CHALLENGE

- 100 hours of play.
- 100 hours of testing!

and…

- The Division is a live game!

A LIVE GAME NEEDS…

- Constant maintenance.
- With every patch → new code and data.
- With new code and data → budgets and behaviour are challenged.
- QA passes during the life of the product.

AND ON TOP OF THAT…

Our worlds are **BIG!**

OUR GRAND VISION - AUTOMATION

- Bots that take control of the input and play.
- Test all Main Missions overnight.
- World crawling and performance statistics gathering.
- Support ‘player-like’ follow bots to assist developers and QA.

OUR GRAND VISION - AUTOMATION

- The Client Bots were born!

PART 2 – SYSTEM DESIGN

OBJECTIVES

- Unattended play, functioning as close to a human as possible.
- Exercising client-side code.
- Adhering to game rules.
- Able to run on PC or any target console.
- No modifications to memory usage, minimal CPU overhead.
- Pragmatic development – Reuse & Repurpose.

OPTIONS

- Repurpose an existing NPC.

- Create a Player AI from scratch.

REPURPOSED NPC

- Create an NPC behaviour to mimic a human player.
- Modify all necessary systems to consider this NPC a human player.

\+ Faster to get a complex behaviour up and running.

\- Bypasses many systems which a human player would normally use like player animations, camera, UI, etc.

\- Needs support for AI to run on the client.

BESPOKE PLAYER AI

Create a layer allowing an AI to directly control player inputs.

+ Every system considers it to be a human player.
+ All client code exercised.

- Significantly longer development time.

This was the option we selected!

PART 3 - IMPLEMENTATION

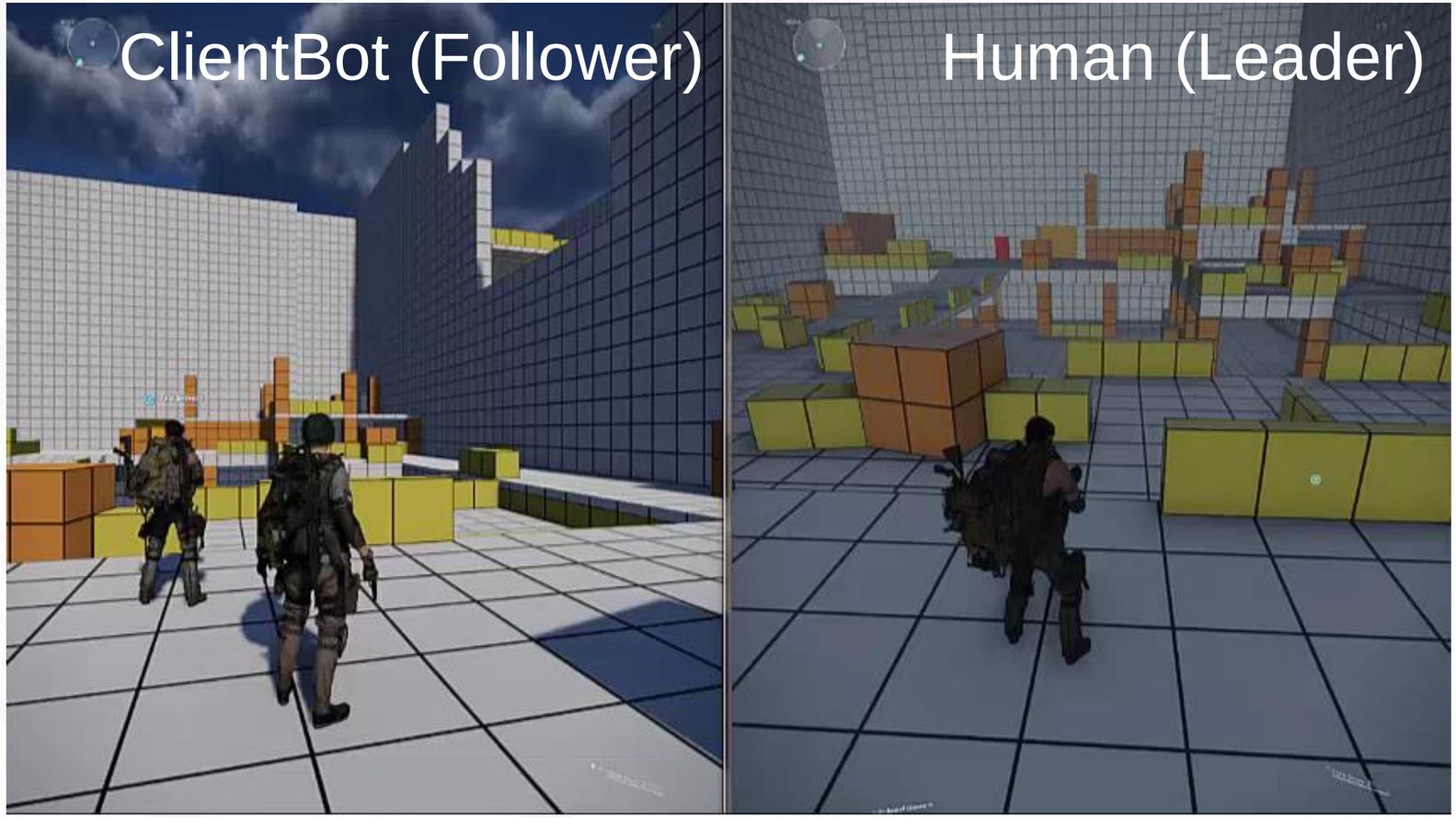
IMPLEMENTING CLIENTBOTS INTO A STABLE CODEBASE

BASIC MOVEMENT

- 100% Nav-Mesh coverage.
- Player constrained to it.
- Repurpose Cover to Cover
- Add support for ladders and ropes.

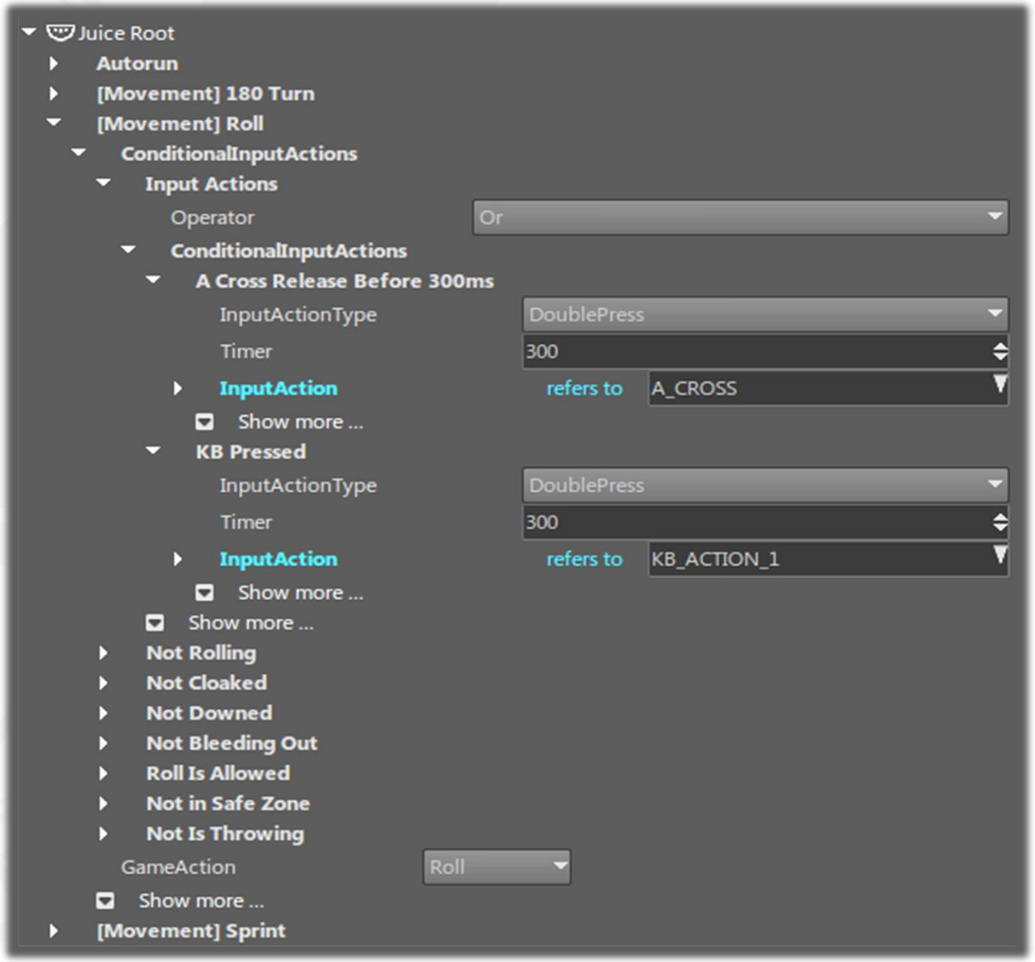
FOLLOW BEHAVIOUR

- Add grouping support.
- Path-find to leader.
- Yields a very useful basic behaviour.



TRIGGERING PLAYER ACTIONS

- Game Action System.
- Input actions / state conditions which must be valid.
- Inject actions directly
  - Bypassing only the controller layer



TRIGGERING PLAYER ACTIONS

- Added a simple interface through the debug console.

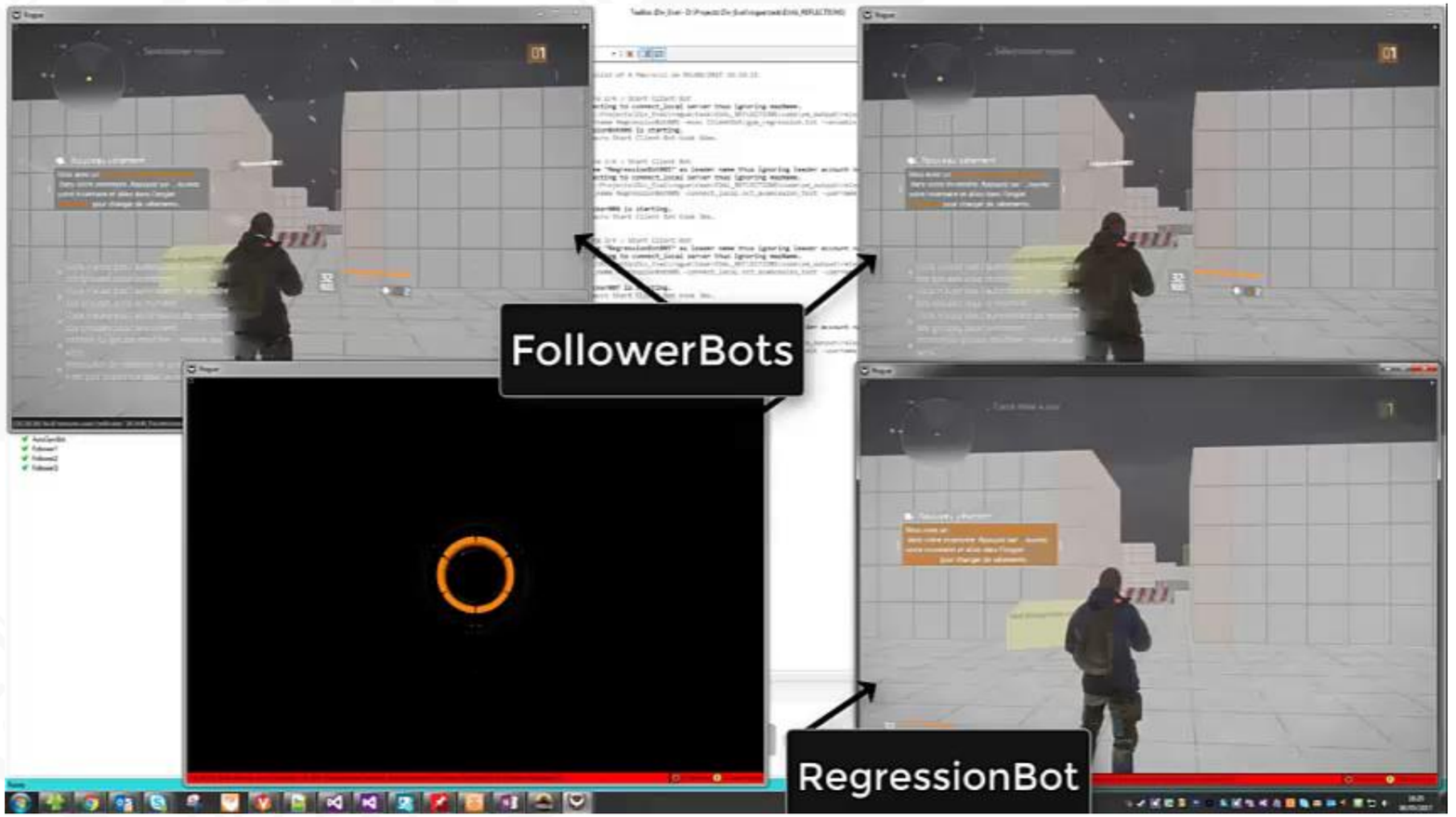
- Basic commands
  - Move to / Is close to
  - Inject Game action
  - Wait

```
console.cmd cmd="ClientBot.SetParam enable 1"
script.wait time=0.5

console.cmd cmd="ClientBot.SetParam destination (24.98 0.01 -21.83)"
console.cmd cmd="ClientBot.SetParam request_go_to 2"
ClientBot.WaitForAroundPosition timeout=5.0 pos="(24.98 0.01 -21.83)" radius=3.0
console.cmd cmd="ClientBot.SetParam request_go_to 1"

console.cmd cmd="ClientBot.InjectGameAction Roll"
script.wait time=1


console.cmd cmd="ClientBot.SetParam destination (36.90 0.01 -21.96)"
console.cmd cmd="ClientBot.SetParam request_go_to 2"
ClientBot.WaitForAroundPosition timeout=5.0 pos="(36.90 0.01 -21.96)"
script.wait time=0.5
console.cmd cmd="ClientBot.SetParam request_go_to 1"

console.cmd cmd="ClientBot.InjectGameAction EnterCover"
script.wait time=1

console.cmd cmd="ClientBot.SetParam destination (42.20 0.01 -25.28)"
console.cmd cmd="ClientBot.SetParam request_go_to 4"
ClientBot.WaitForAroundPosition timeout=5.0 pos="(42.20 0.01 -25.28)"
script.wait time=1
```

AUTOMATED GYM PLAYTHROUGH

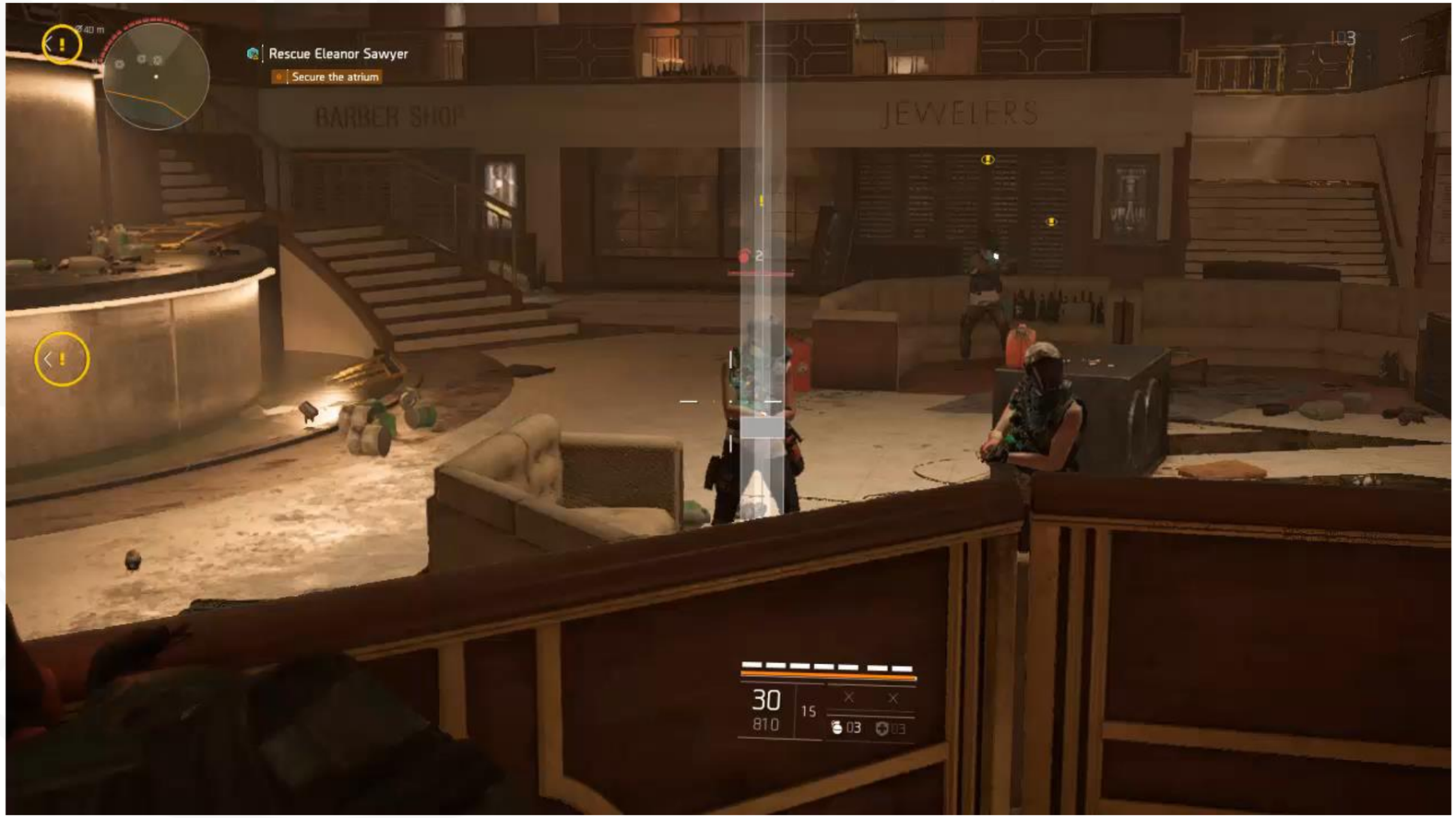
- Human play-through required for every check-in to test core functionality.

- Automate it!



SIMPLE COMBAT BEHAVIOUR

- Manipulate camera.
  - Aim for the head!
- Inject Game Actions for Shoot & Reload.
- No health management / hazard avoidance.



AUTONOMOUS MISSION COMPLETION

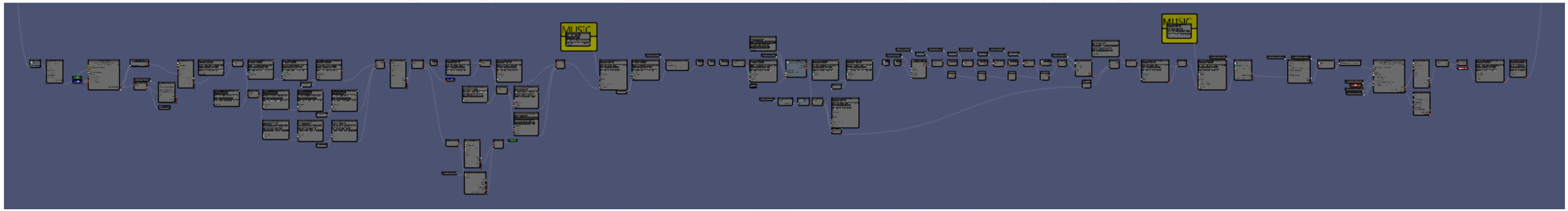
Big step up in difficulty!

Query systems to fathom how to progress:

- Mission System.
- Friendly and Hostile NPC’s.
- Interaction Markers.
- Destructible Objects.

INTERROGATING THE MISSION SYSTEM

- A mixture of bespoke and general purpose nodes.
  - Too complex to static analyse.
- Use the outputs and treat it as a ‘Black Box’.



PLAYING A MISSION

IT WORKED! … EVENTUALLY!

We encountered many problems along the way:

- Physics/NavMesh resident radius.
- Entities/Interactions not created until you are close.
- Mission objectives ‘evolve’ as you progress.

- AND the game was released so data needed to be treated as “immutable”.

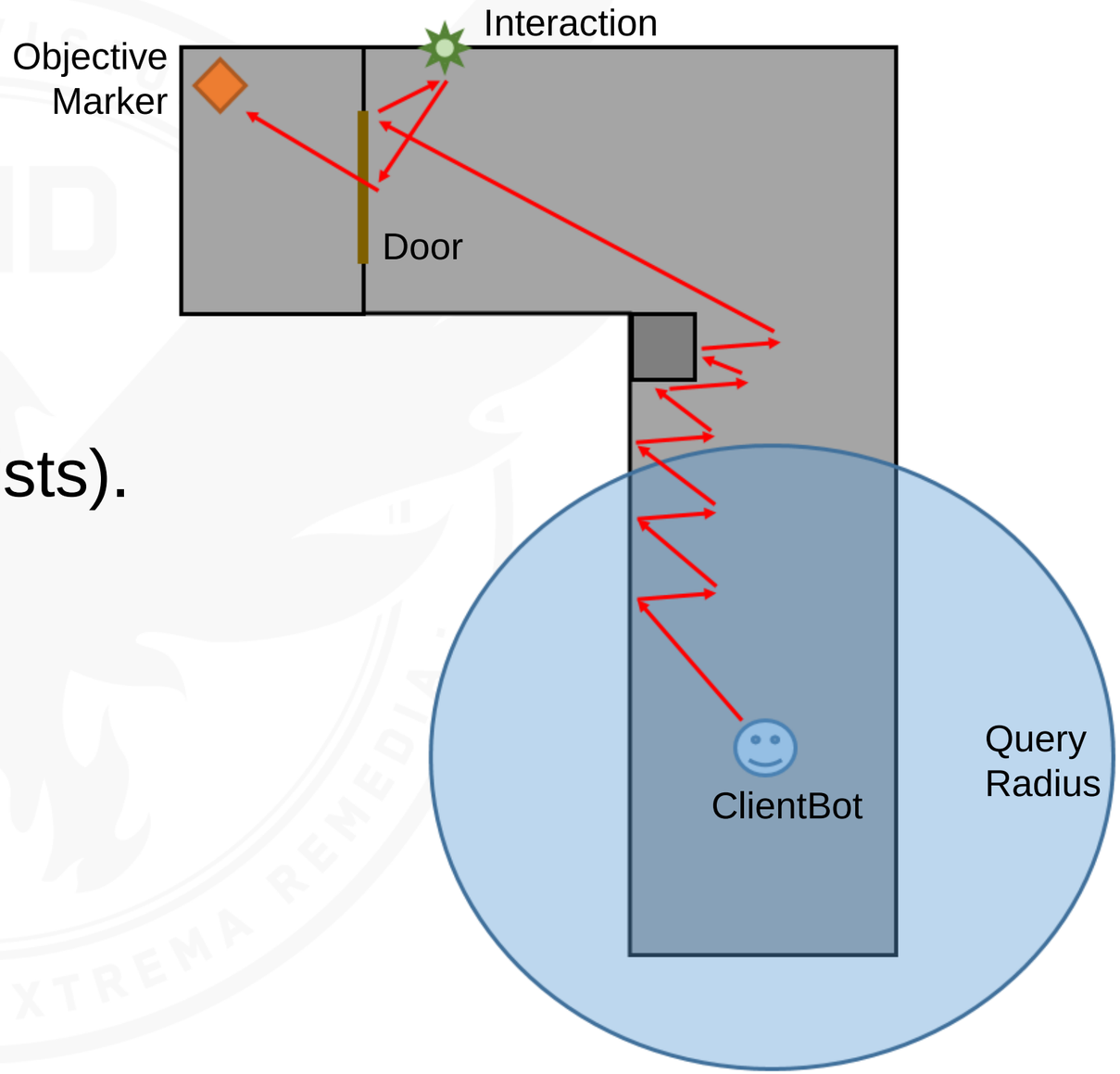
ITERATIVE DISCOVERY

- Path-find to destination often fails.
  - Doors can be closed, interactions must be triggered to progress.
- Objectives update based on proximity.
- Iterate toward destination and exhaust all options.

THE PROCESS

Priority List:

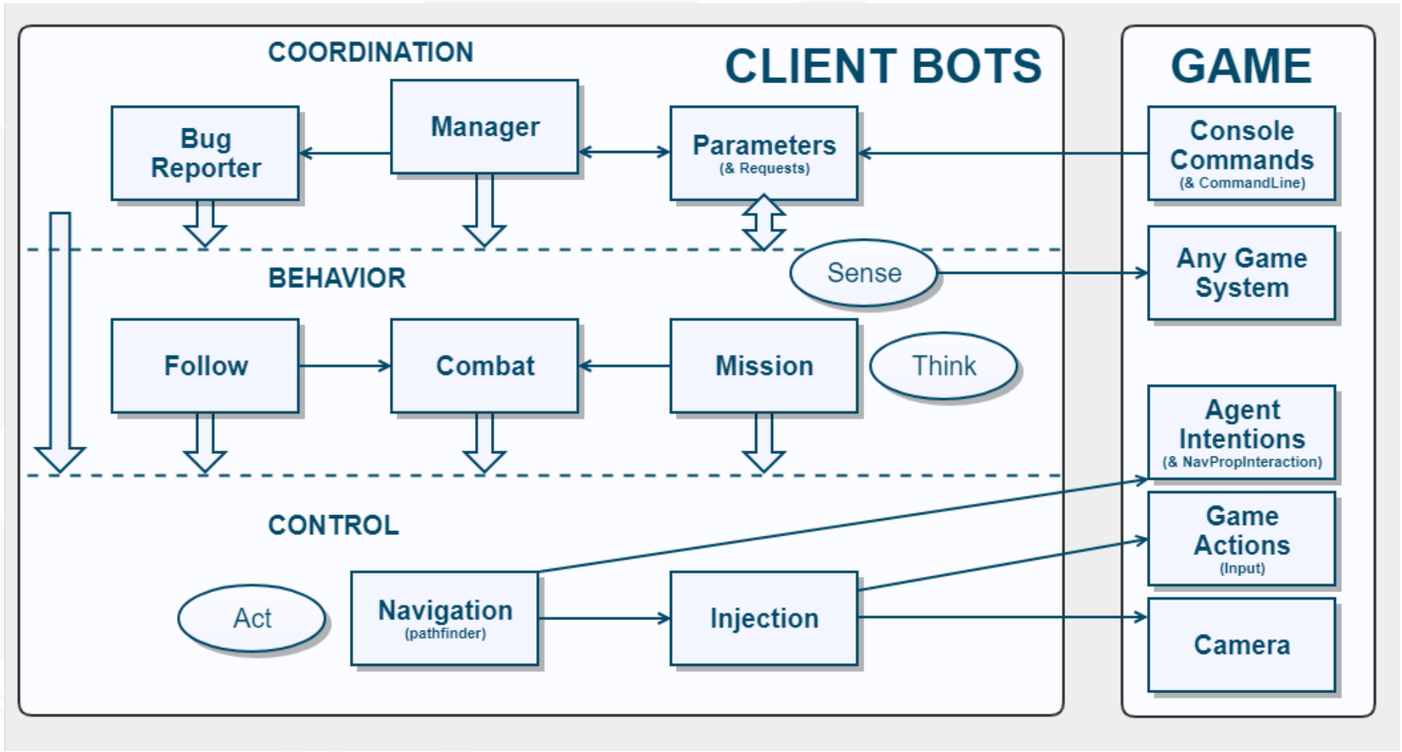
1. Kill Enemies.
2. Follow path to objective (if one exists).
3. Trigger useful interactions.
4. Destroy specific destructibles.
5. Travel directly toward.
6. Wander.



OVERALL ARCHITECTURE

- All access through a stub
  - Compiled out for retail
  - 6 queries for Client Bot state
  - 9 Invocations

- No additional ‘checkers’
  - Code bloat was a concern



USAGE IN THE DIVISION

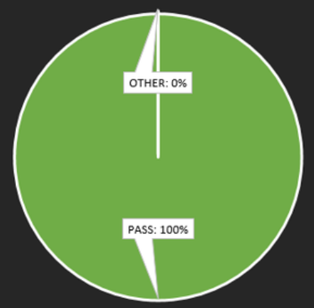
- Used for **mission smokes.**

- All Manhattan missions every night.
  - Completed in 5 hours
- Underground content on week-ends.
  - Completed in 7 hours

More hardware – faster results!

CLIENT BOTS — SMOKE TEST

MANHATTAN — MAIN MISSIONS



Project
Branch MAIN
Changelist
Date 08/08/2018

MISSION	PASSES	STATUS	PROGRESS	DURATION
atlas_detention	5/5	PASS	100%	0:09:37
		PASS	100%	0:09:24
		PASS	100%	0:09:00
		PASS	100%	0:09:13
		PASS	100%	0:08:58
chip_gateway	5/5	PASS	100%	0:15:27
		PASS	100%	0:12:58
		PASS	100%	0:16:12
		PASS	100%	0:15:58
		PASS	100%	0:12:21
construction	5/5	PASS	100%	0:14:14
		PASS	100%	0:14:26
		PASS	100%	0:14:42
		PASS	100%	0:14:13
		PASS	100%	0:13:49
de_kipsbay	5/5	PASS	100%	0:11:13
		PASS	100%	0:10:48
		PASS	100%	0:11:12
		PASS	100%	0:11:06
		PASS	100%	0:11:01
dyer	5/5	PASS	100%	0:12:41
		PASS	100%	0:12:39
		PASS	100%	0:12:49
		PASS	100%	0:12:19
		PASS	100%	0:13:55
grandcentral_a	5/5	PASS	100%	0:09:32
		PASS	100%	0:10:16
		PASS	100%	0:09:52
		PASS	100%	0:09:16
		PASS	100%	0:09:16

PART 4 - IMPLEMENTATION

WORKING WITH A GAME UNDER DEVELOPMENT

MISSIONS: LESS HAND-HOLDING

Division 2 Missions have less explicit objectives.

Player needs to use their initiative more.

Bots have significantly less guidance.

A CHANGE OF APPROACH

Iterative discovery only gets you so far.

- Results can be unreliable.
  - Need for multiple runs to confirm results.
- Mission completion times vary.
- Was our only option for Division 1.

A PARTNERSHIP WITH LEVEL DESIGNERS

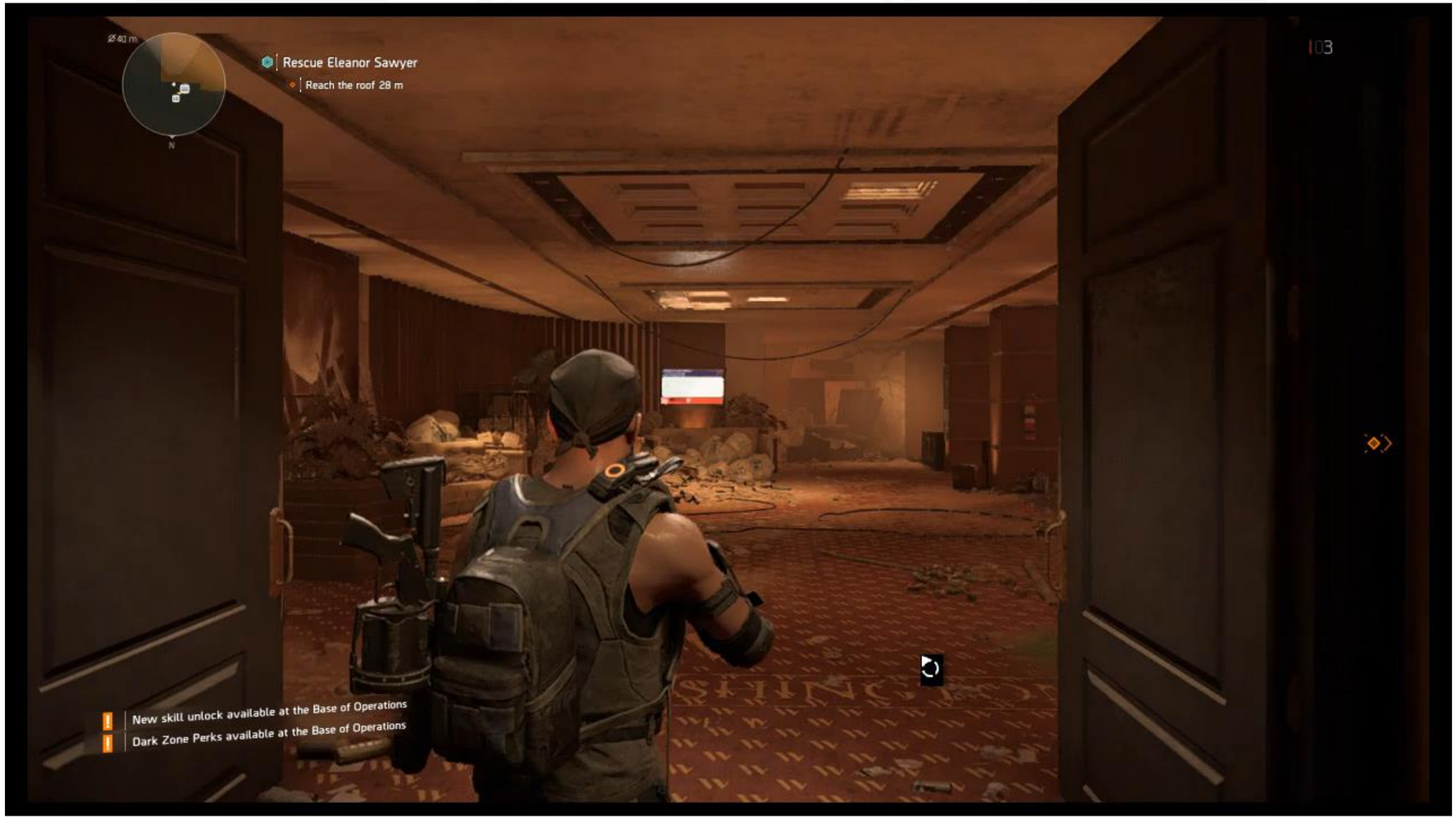
- Embed information in missions to assist.
- Aim for reliable validation.
- Both sides benefit.
  - Embedded information helps the development process.
  - Level Designers benefit by having their missions automatically testable.

HIDDEN TRACKED ENTITIES

Add hidden objectives.

- Gives information about the next objective.
- 1 hour for initial setup per mission.
- Invisible to player.
- Requires minor maintenance.

DIVISION 2 MISSIONS

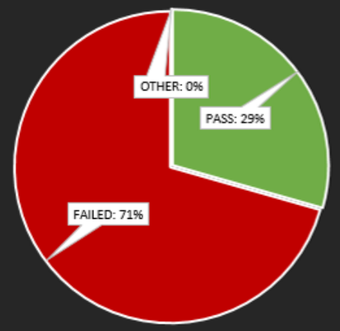


CHALLENGES IN DEVELOPMENT

- Only get mission success when *EVERYTHING* works!
- The 3 Main Breakages
  - Issues with Mission Content
    - Work with LD to fix / add tracked entities
  - Game Systems behaving incorrectly
    - Test in isolation to mission & fix
  - Client Bot behaviour insufficient
    - Improve as necessary
- Aim to get dashboard green, then keep it that way.

Client Bots — Smoke Test

Main Missions

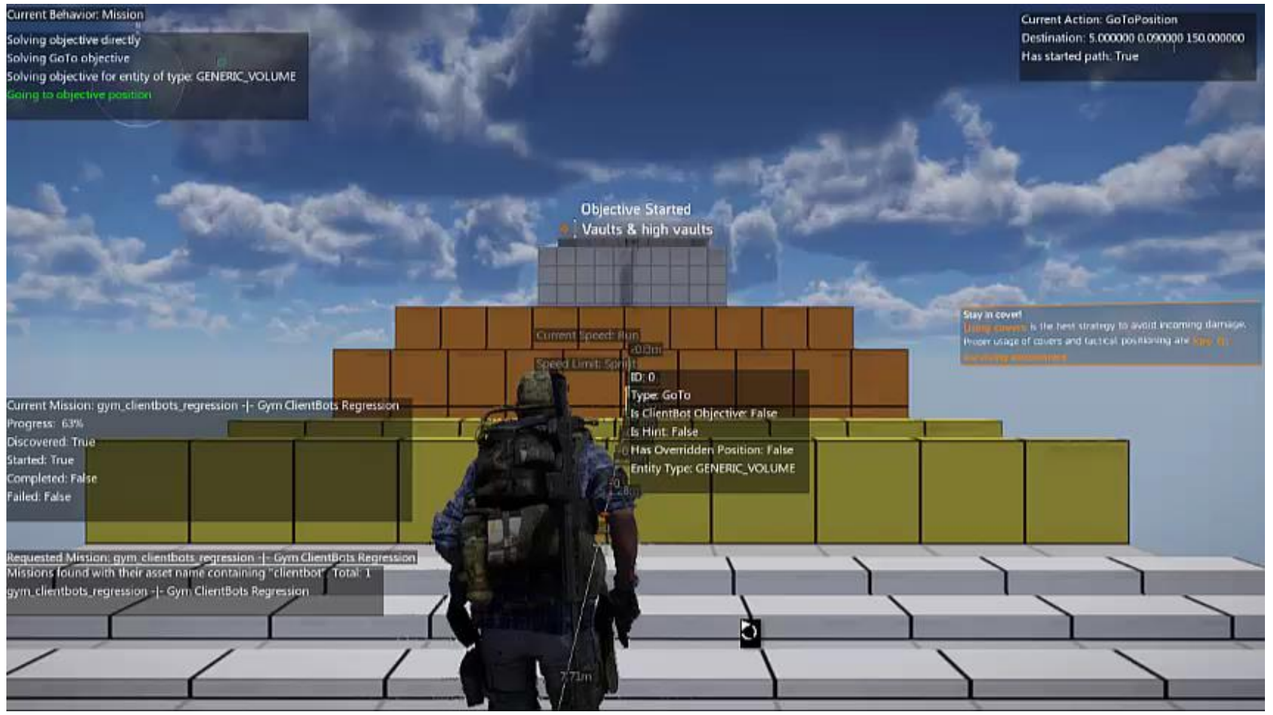


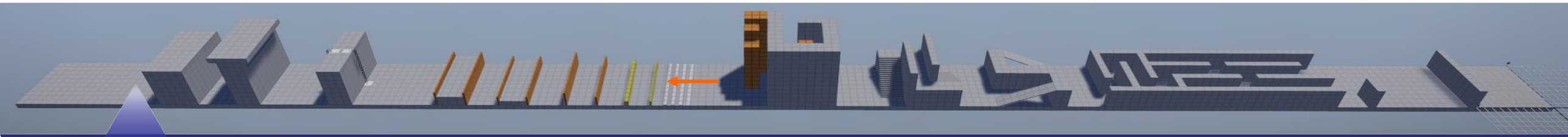
Project
Branch
Changelist
Date 15/08/2018

MISSION	PASSES	STATUS	PROGRESS	DURATION
mm02	0/1	FAILED_STUCK	5%	00:08:04
mm03	0/5	FAILED_STUCK	23%	00:10:38
		FAILED_STUCK	23%	00:10:39
		FAILED_STUCK	23%	00:10:55
		FAILED_STUCK	23%	00:10:55
		FAILED_STUCK	23%	00:10:56
mm10	5/5	PASS	100%	00:22:41
		PASS	100%	00:22:20
		PASS	100%	00:21:28
		PASS	100%	00:20:47
		PASS	100%	00:20:41
mm11	0/1	FAILED_STUCK	0%	00:08:09
mm15	0/5	FAILED_STUCK	16%	00:15:54
		FAILED_STUCK	24%	00:18:03
		FAILED_STUCK	24%	00:20:24
		FAILED_STUCK	24%	00:22:44
		FAILED_STUCK	24%	00:26:21

TESTING GAME SYSTEMS

- Creation of bot gyms:
  - Tests bot functionality.
  - Tests core gameplay mechanics.
  - Aim to integrate into Build System.





CLIENT PERFORMANCE TRACKING

Wandering around the game world, logging performance statistics.

PERFORMANCE TRACKING

- Wandering level behaviour.
- Bots visit all playable space (engaging in combat or not).
- Collect performance information on our tracking framework.
- Results inspected by QA / Tech Art and Bugs are created from that.
  - A nice method for avoiding ‘automated bug spam’.

WORLD COVERAGE

WORLD COVERAGE

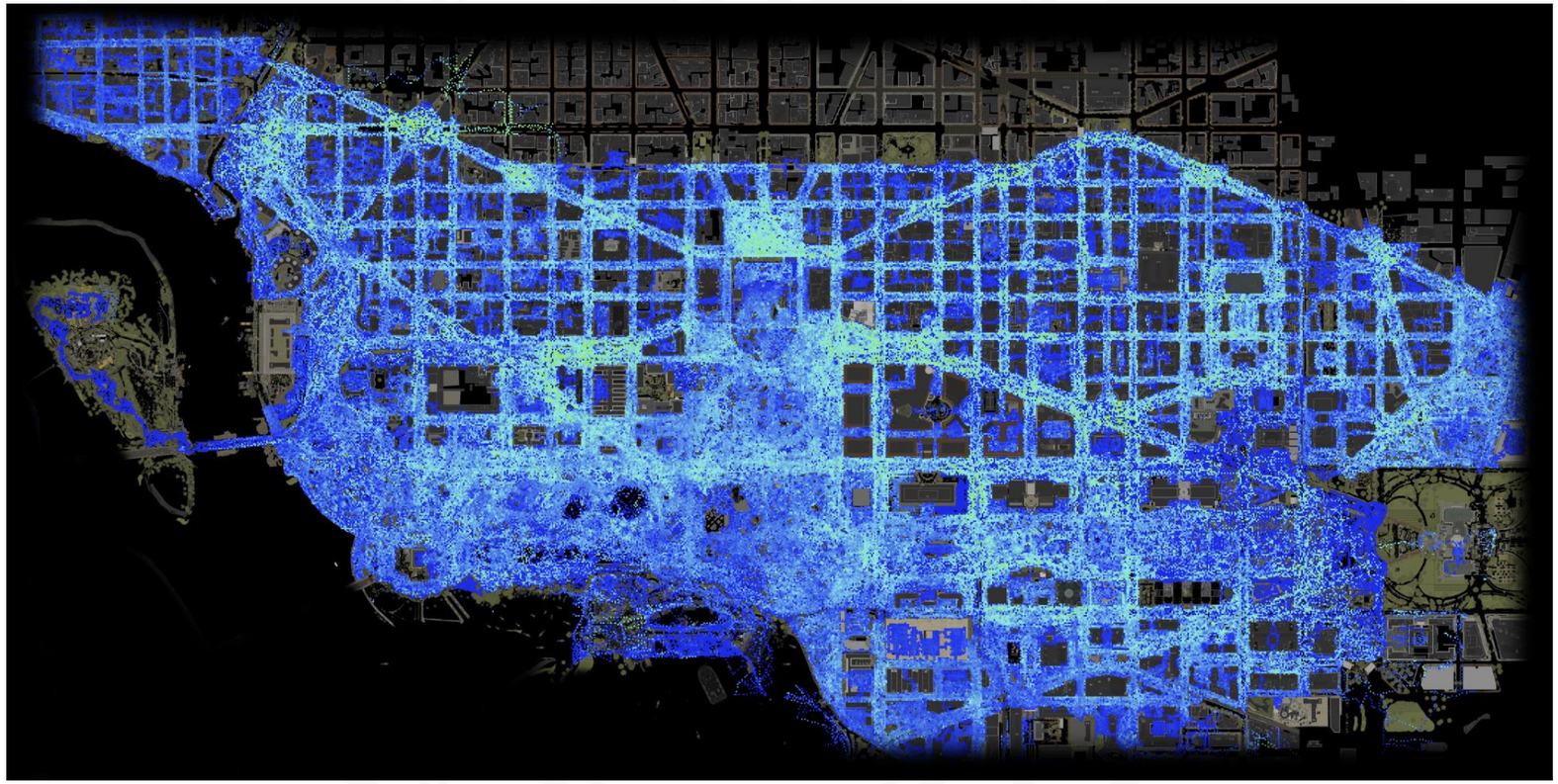
PERFORMANCE ANALYTICS

FPS < 25

PERFORMANCE ANALYTICS

FPS < 25

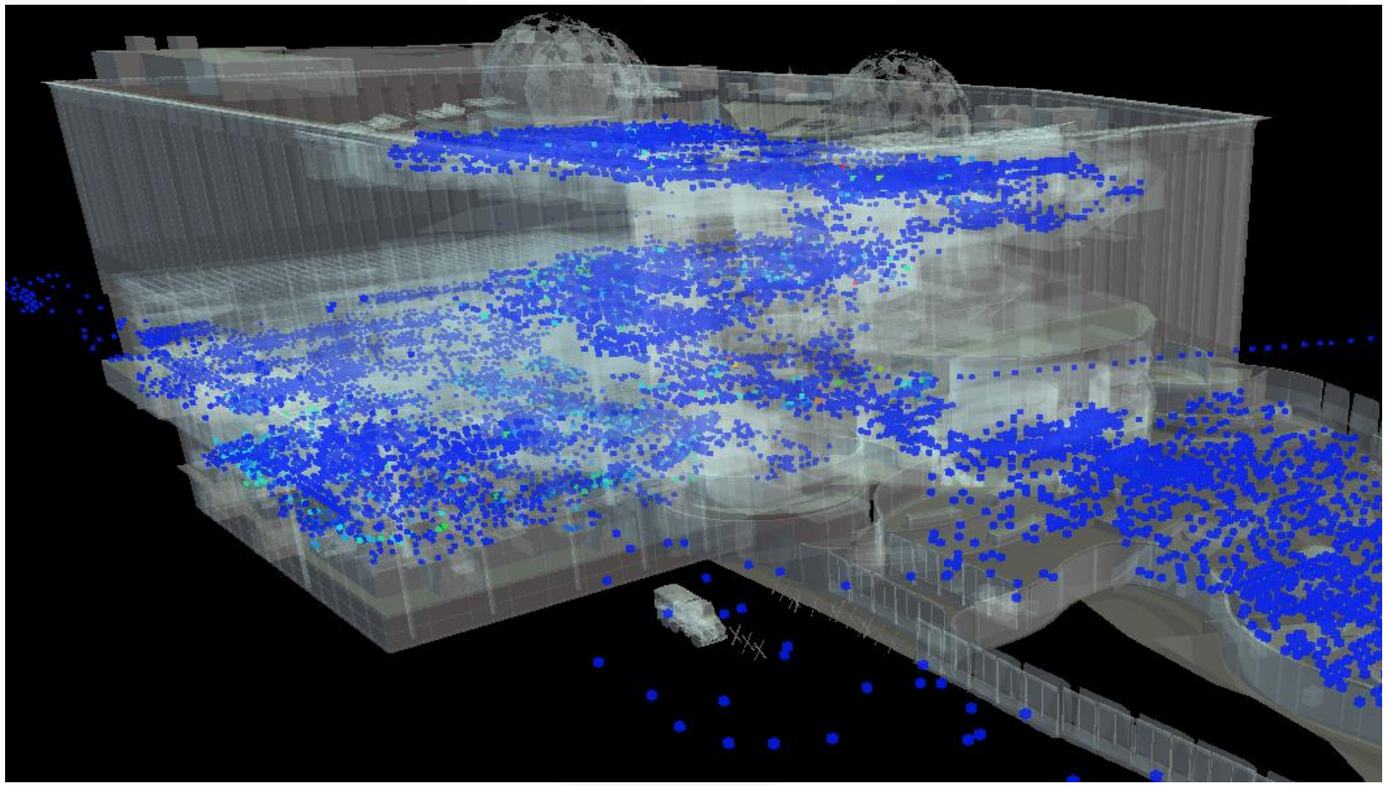
PERFORMANCE ANALYTICS



Mesh Count

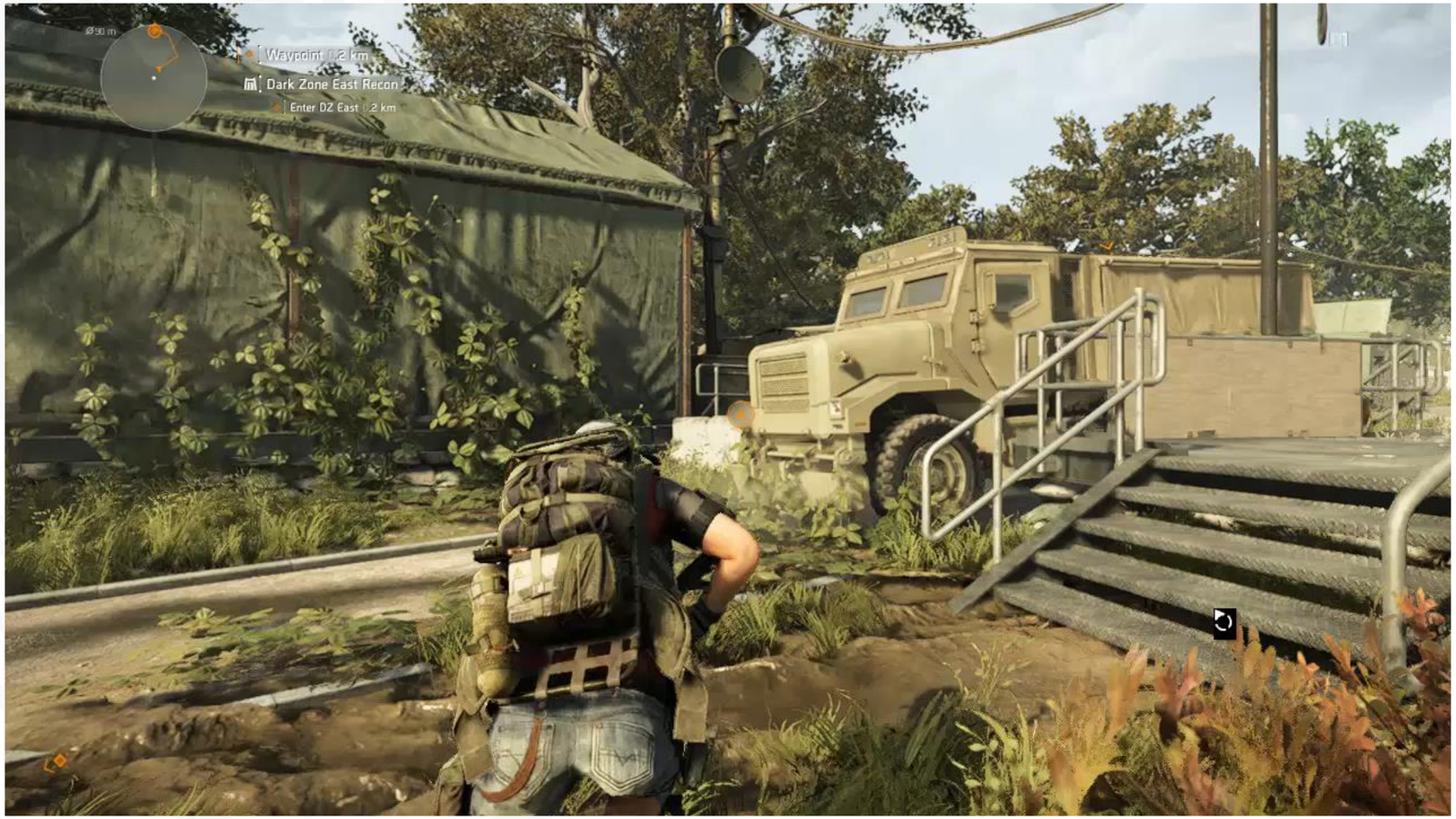
3D PERFORMANCE RECORDING

- 3D Performance tracking within missions:

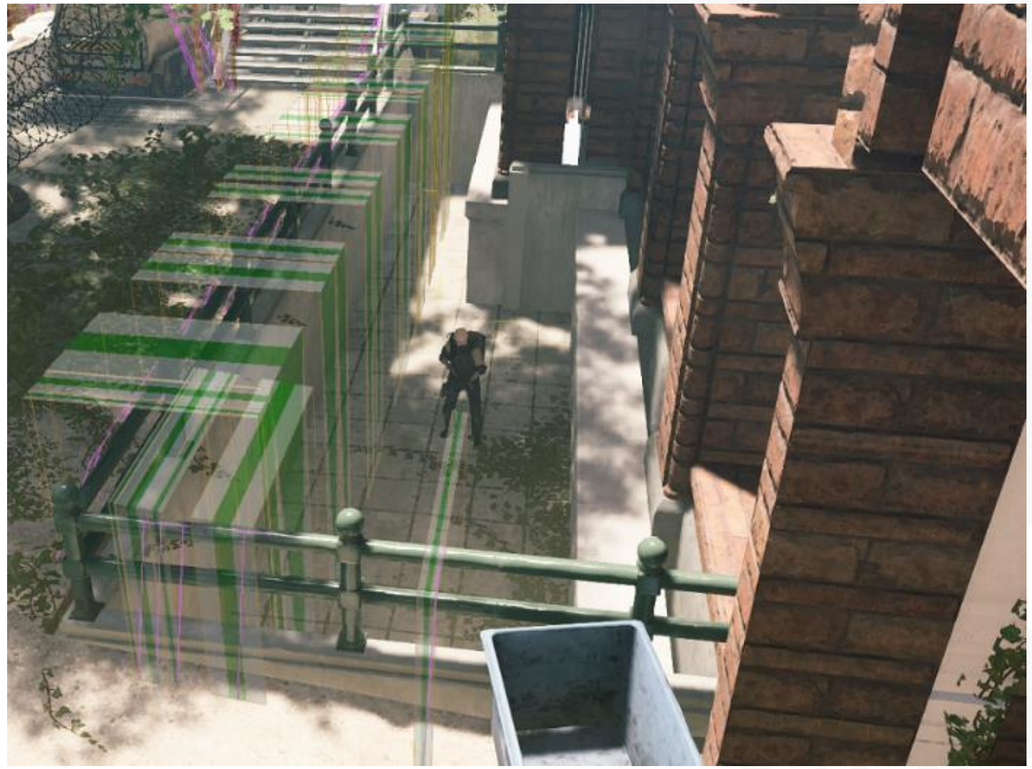


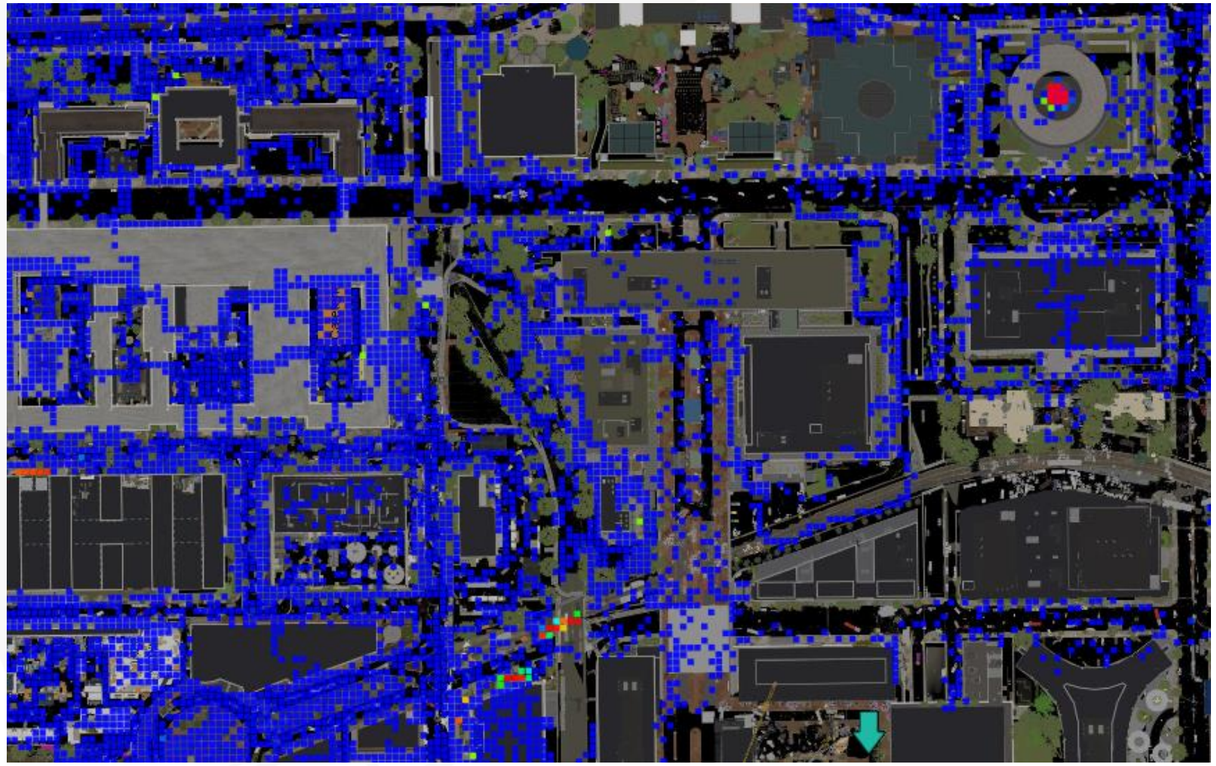
WANDERING BEHAVIOUR STEPS

- Queries neighbouring go-to positions.
- Chooses the most distant.
- Validates that return path-find succeeds.
- Repeats.



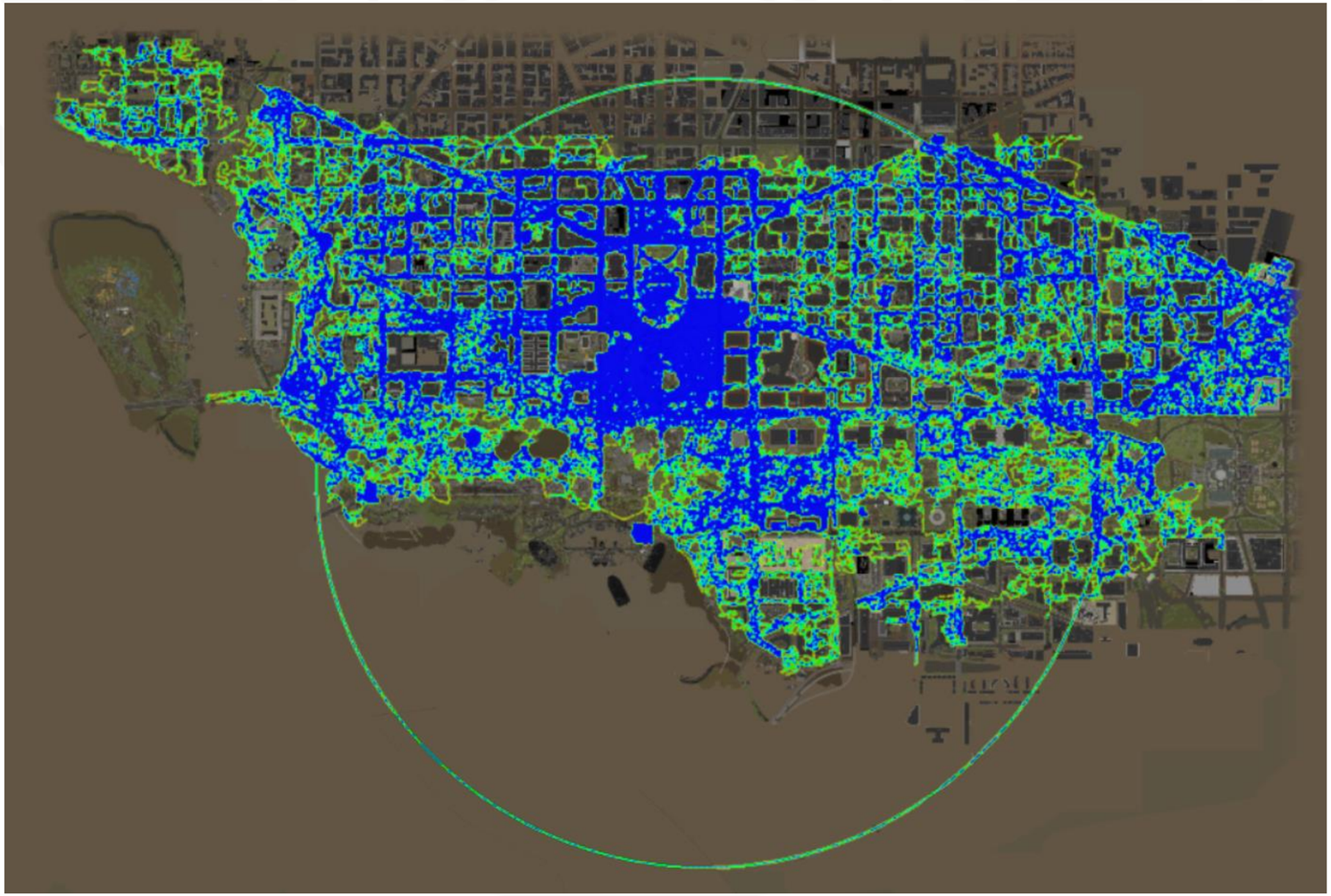
STUCK DETECTION





Areas which can be entered into but not returned from

BOTS CONSPIRACY THEORIES…



PERFORMANCE TRACKING – WHEN?

- **Nightly**
- **Weekend**
  - 48 hour non stop runs.

PART 5 – NEXT STEPS

NEW FUNCTIONALITY

- Improved Wandering.
  - Loot.
  - Engage in game activities.
- Support for testing new content.

PART 6 – CONCLUSION

DOESN’T HAVE TO BE COMPLEX

- Complex AI is not always needed.

- Reuse what is available for you.
  - Game systems (NavMesh, player actions, etc).
  - Tracking framework.

DEVELOPMENT CHALLENGES

- Mission smoke testing.
  - On Division 1
    - Game was “immutable” -> easier to develop.
  - On Division 2
    - Game was changing -> not very resilient to every day changes.
    - False positives issue.

DEVELOPMENT CHALLENGES

- Performance testing.
  - Very resilient to every day changes.
  - We could cover most of the world in every run.

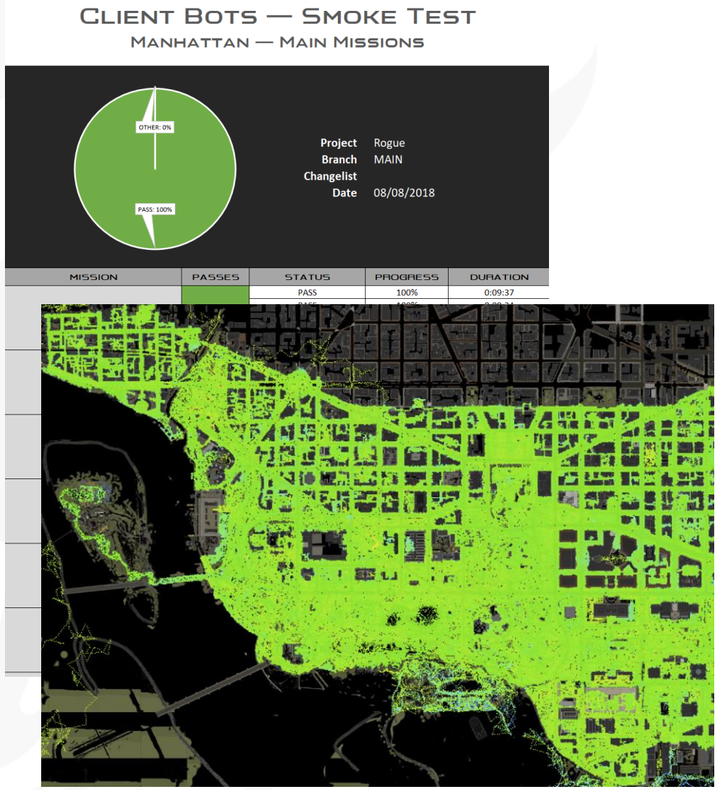
AD-HOC USAGES

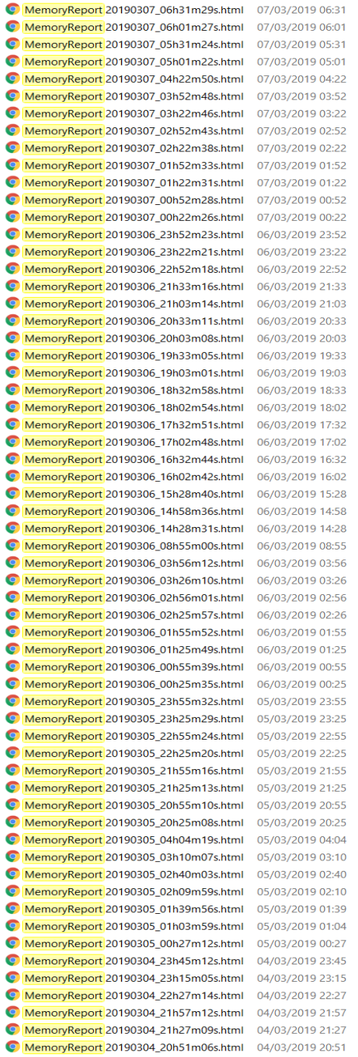
- Once the system is running…
  - Awareness of usefulness raises dramatically!
    - Memory reports in long hours.
    - Bots sending fake VOIP data to test VOIP.
    - And much more!

DATA CHALLENGE

- Lots of data!
  - Mission smoke reports.
  - Performance statistics dashboards.
  - GBs of Memory Reports.

- Data generation automated.
- Data consumption not automated.





FINAL CONCLUSION

- Big worlds.
- Live for years.
- Ongoing testing needed.
- **We NEED Automated Testing!**

SPECIAL THANKS

- PABLO RODRIGUEZ CODES
- ARTHUR VAN-CEULEN

THANK YOU!

QUESTIONS?


ARE YOU READY TO
CREATE THE UNKNOWN?
jobs.ubisoft.com
LEARN MORE AT OUR BOOTH!
WEST HALL - FLOOR 2
UBISOFT