The Art of Analytics and Automated Playthroughs (Level 3 EXCLUSIVE Content)

What's up fellow Guru!

It's great to have you here!

In this short but powerful **Level 3 Exclusive Blog Post**, you will learn about **extending your automated playthroughs to new heights**.

And if you don't have your tests in place, as shown in the Level 2 Post, you can still hugely profit from this based on real human interactions.

So, what's cooking?

Time	Marker	Health	Enemies	TimeLeft	DrawCalls	FrameTime	RenderTime	Vertices	Triangles	TextureMemory
0.00	Stalking-NPC-Jamie	50	39	4.5	7	0	0.001758299	180	73	122
1.01	Attacking-NPC-Jamie	30	38	3.47	21	0	0.0004891999	832	540	421
2.01	Running-Away-From-NPC-Jamie	65	1	2.46	30	0	0.0003982	648	454	743
3.02	Resting-In-North-Cave	17	31	1.45	6	0	0.0004142002	830	550	104
4.03	Crying-Alone-In-The-Woods	96	8	0.44	16	0	0.0004339003	124	50	305
5.04	Dead	150	0	0.2439	40	0	0.4551	2020	1300	634

Level 3 Unity Developer (Unity Immediate): Sample Output Table

Level 3 Developer: The PlayRecorder API

Time to offer you a quick tour on the exclusive ***PlayRecorder API***!

The *PlayRecorder API* is a *plug and play* script that you can easily place in your project. I paid special attention to making it simple, because **I want you to start recording your play sessions in no time**. The sooner you gather data, the more options you have.

From a very high-level perspective, this is what happens when you run the system with no special configuration:

- ✔ A new CSV file is created at your persistent data path
- ✔ General metrics are recorded once per second second
- ✔ When stopped, a explorer window will open showing you the location of the CSV file.

And this is **what you get out of the box** in your new CSV file:

- **Timestamp**: when was this entry recorded? E.g. 30.5 (in seconds)
- **Marker**: what is the game state when this entry was recorded? E.g. *Jumping-Over-First-Enemy*
- **Performance metrics:** draw calls, vertex+triangle count, frame and render time, used memory size.
  You can add more metrics here! Those, however, will only be recorded in the editor. We do not have access to these metrics on any device until at least Unity 2020.
- **Gameplay metrics**: here you optionally pass a custom dictionary to record important gameplay data. This is very specific to your game, but you can have some inspiration from this list: enemies left, time left, score, current gold/coins, player's health/mana, current level, players in the lobby, etc..

Download the PlayRecorder script and its example below!

PlayRecorder Usage

1. **Download and import *PlayRecorder.cs*** in your project and add it to a global GameObject.
Note that this object won't be destroyed (*DontDestroyOnLoad*), so keep it safe!

2. Decide when to start your play session, e.g. by pressing a button or when your level start.
When that happens, **call *PlayRecorder.instance.StartRecording***("dungeon-5")

3. Leave the user undisturbed when playing. At some point, he will finish the game or the level.
Then, **invoke *PlayRecorder.instance.StopRecording***()

When the user finished playing, **the CSV file will be waiting for you in the persistent data path.** Because we always forget where that is, I made sure a new explorer window is open pointing at location of this file.
That's great raw data, but not that useful per-se!
Here's my suggestion for you, fellow bulldog: **augment the information**! Don't settle with raw metrics, add something useful related to your game. It is actually really easy to **pass your own gameplay metrics**.

Adding Gameplay metrics

1. **Define a C# Gameplay Metrics Provider**.
This is nothing more than defining a function returning a *Dictionary*<string, object>, where the key is the name of your metric and the object is its value (usually an integer).
A metric would be for instance a *KeyValuePair* of {"*EnemiesLeft*", *MyLevel.Enemies.Count* }

2. **Pass your new Metrics Provider to the PlayRecorder API**.
Before starting the recording, invoke *PlayRecorder.instance.StartRecording* as usual but pass your function as a second parameter.

3. **Update your Game Marker.**
To identify later on where your player was at each point, update the string *PlayRecorder.instance.CurrentMarker* with a short description of what the player is currently doing, e.g. *Crafting-Items-At-Forge*.

And that's it! The system will ask you for your game metrics every second and record them along the general metrics in the CSV file.

Note I can't provide you will all possible metrics beforehand, since this is very game-specific. If you need some consulting on this, reach me at *ruben@thegamedev.guru* so we can arrange something, but doing this by yourself should be feasible.

By the way, the default **recording tick rate** is 1 second. This means, a new entry with all the defined metrics will be created every second. If you need this to be faster/slower, you can pass a different figure to *StartRecording* as the third argument. Bear in mind that this will affect performance, as there is some C# boxing going on. You might expect to see increased garbage collections while recording.

What can you do with the generated data? My suggestion is to **generate a table out of the CSV file**. That's really easy and fast if you use this free online tool. Simply paste the contents of your CSV and generate a HTML table.

If you want to take this to the next level and you know some excel, you can **generate charts to see the evolution of different variables over time**. You can go pro and compare the evolution of these graphs over time, which is especially relevant for performance analysis.

Level 3 Developer: Analytics Strategies (Extra!)

In a few weeks time, you should have gathered some data. Try to have at least 20 captures.

Now it's the time to put those numbers to work for us.

Here I will share with you some **strategies** I've successfully used in the past:

- **Analyze how player's *health/mana/gold/...* evolves over time**.
  Watch for spikes. If you see people are dying often in the same *Game Marker* (e.g. *Dungeon-5-Section-3*) , it might be time to revisit the difficulty of that area.
- **Analyze how much *Time* people spend on each *Marker***.
  People spending too much time in specific areas of your game is a common source of frustration. Consider adding some clues, as they might be stuck.
- **Create memorable and epic moments**.
  Define and watch metrics that indirectly reflect how "boring" or "fun" each gameplay section is.
  Use these metrics to your advantage to create some dynamism in your game. You can add, for instance, special sound effects in key areas of the game. And those moments are not always designed but often discovered through metrics.
  An example of this metric could be monsters *slayed per minute*. Just like in *Dungeon of the Endless*, you'll want to alternate between giving your players some rest (0 *monsters slayed per minute*), offering them a baseline stress level (5 *monsters/minute*) and spiking their adrenaline levels with peak moments (100 *monsters/minute*).
- **Balance ads**.
  If your game is in such a market, please track how often an ad is displayed so you make sure it does not become too spammy. You may do so by adding it as a custom gameplay metrics.

So far, you and I considered these systems for internal play tests. Here's a new one: **you can extend this system further to production environments so that you start gathering the same data from live players**.

The main benefit of live tracking is, of course, that its data is less artificial and much closer to reality. I have done this in the past with great success, as it is a great source of information you can use to keep improving your games. What we want at the end is that people enjoy them better.

These metrics could be sent through internet automatically, for instance. But hey,. make sure to protect users' privacy if you do this. Inform them of what you are doing and why. Also, give them the option to opt-out and do not forget to anonymize the information as much as possible. Hiring a lawyer for live tracking might be a good investment.

I hope you enjoyed this article series. If you have any feedback, you know where to find me.

Rubén