HeatLog by Evolvedlabs SAS

staff@evolvedlabs.com

www.evolvedlabs.com

DOCUMENTATION 1.0

Feature overview

Thank you for purchasing HeatLog!

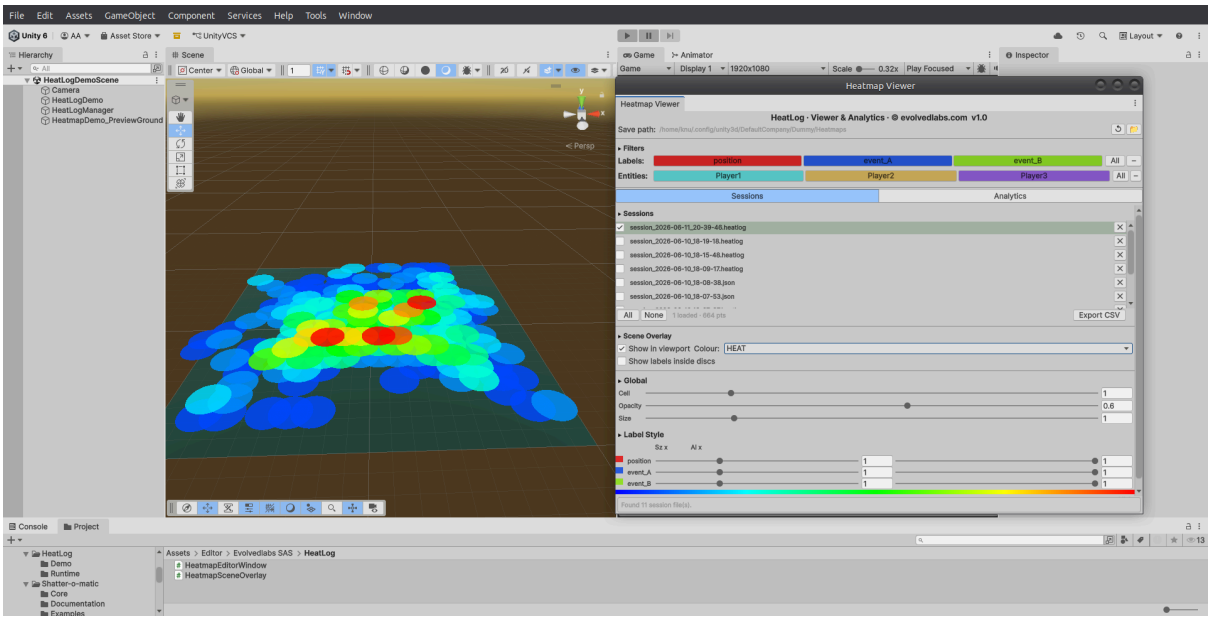
This Unity3D asset allows you to easily save and visualize spatial log data directly in your Unity3D editor.

This is especially useful to track customizable events such as player / NPC movements over time and their actions or to track hotspots (camping points, areas where bots get stuck, areas where players die often, etc).

It is suggested to set a reasonable logging time if you are planning to ship HeatLog in the final client and to pick the binary format in case you'll then be storing that log somewhere.

The log is saved locally or in memory, but how to send that back to you is left for you to implement.

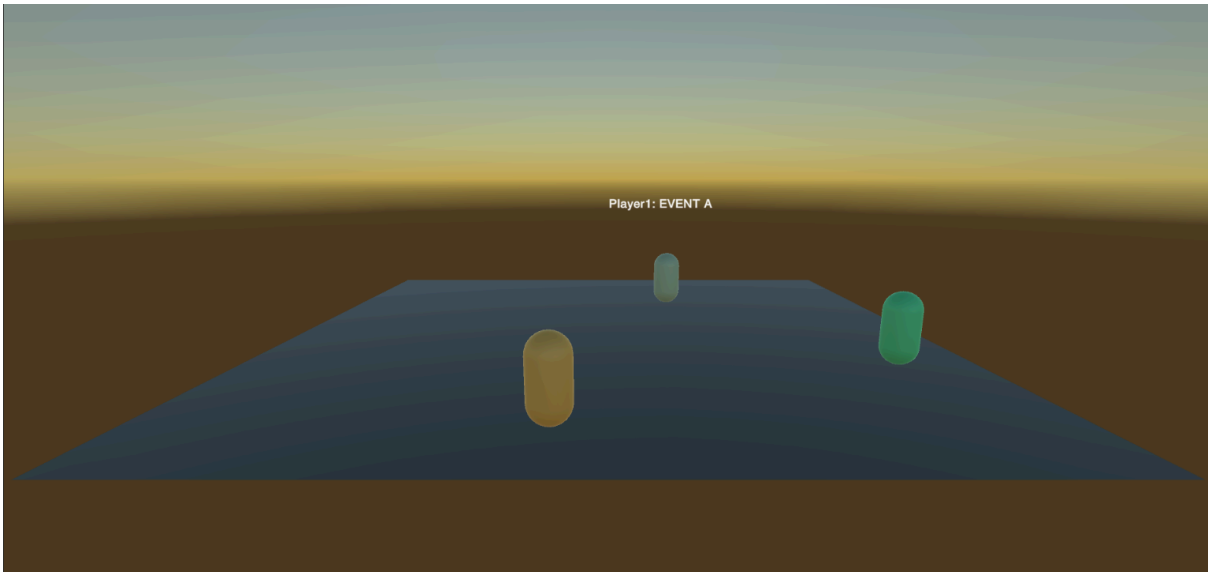
Once the log has been generated, it can simply be loaded in the editor and visualized with various filters.



Getting started

A simple demo scene is provided in the package to showcase how to use and integrate HeatLog in your project.

This demo scene will automatically spawn a few “players” that randomly move around the scene and will randomly fire an event.



By default, every time you start a new play session in the editor, a new log file is created.

Load the demo scene, play it for a few seconds, then stop the execution. A .json or .heatlog file will be automatically saved for that play session.

Now you can simply click the **“Tools” -> “HeatLog” -> “Heatmap Viewer & Analytics”** window. You’ll see the list of the detected sessions.

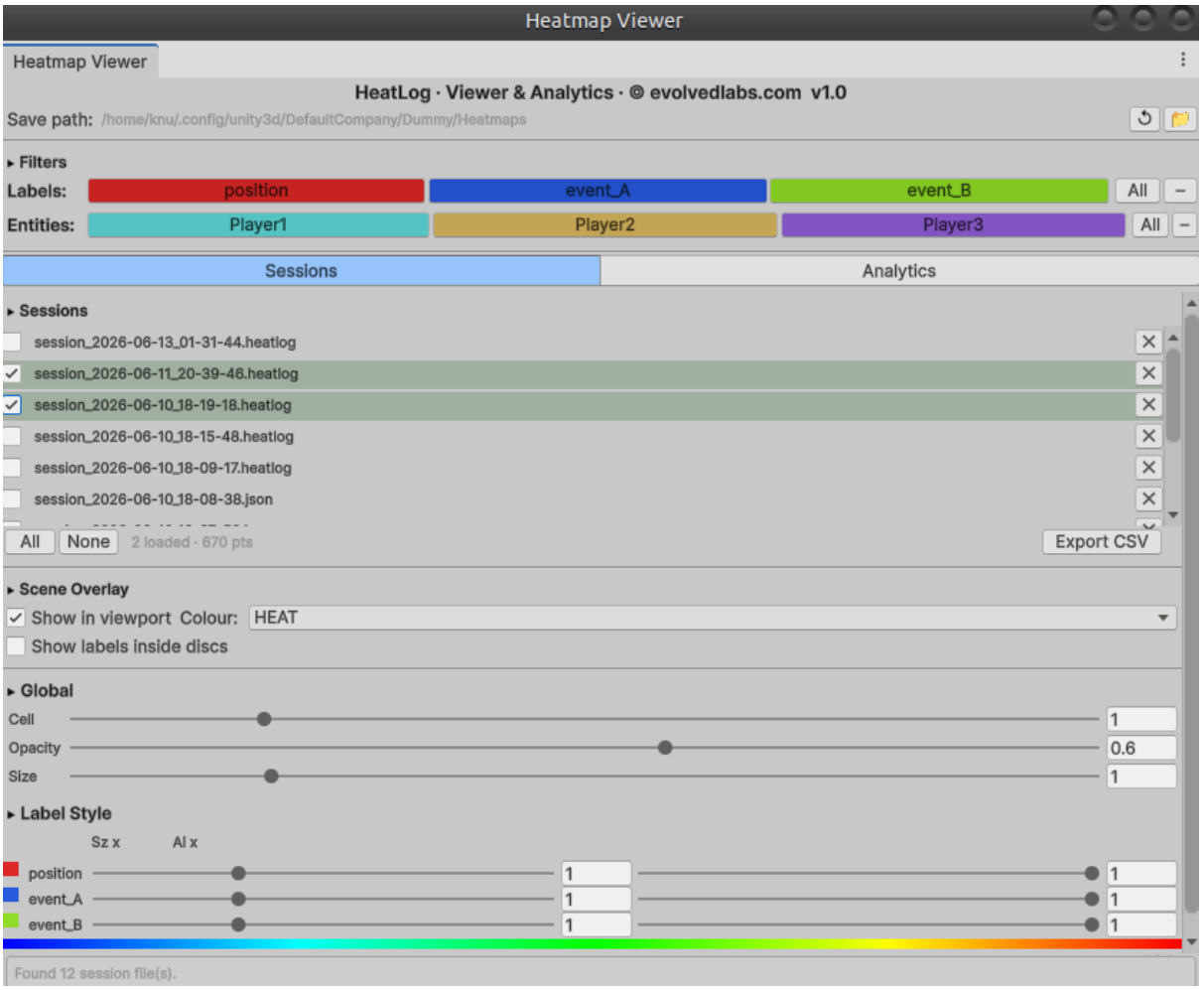
Left click on the desired session, and check “view in editor” to actually visualize in the editor the heatmap. You can use the filters in the user interface to enable/disable the relevant view (for example if you want to see all events from Player 1, or all events of type “EventA”).

The interface is split into two tabs:

Sessions tab: select sessions, set filters, configure overlay.
Analytics tab: event counts and timeline chart.

Note that the “Filters” interface is common and allows to toggle the visualization of any “label” (event you log in your game) and entities (whatever actually logged the event)

The Session Tab



This tab allows you to load any session that has been previously saved. Check whatever session you want to load and it will instantly be visualized in the editor viewport.

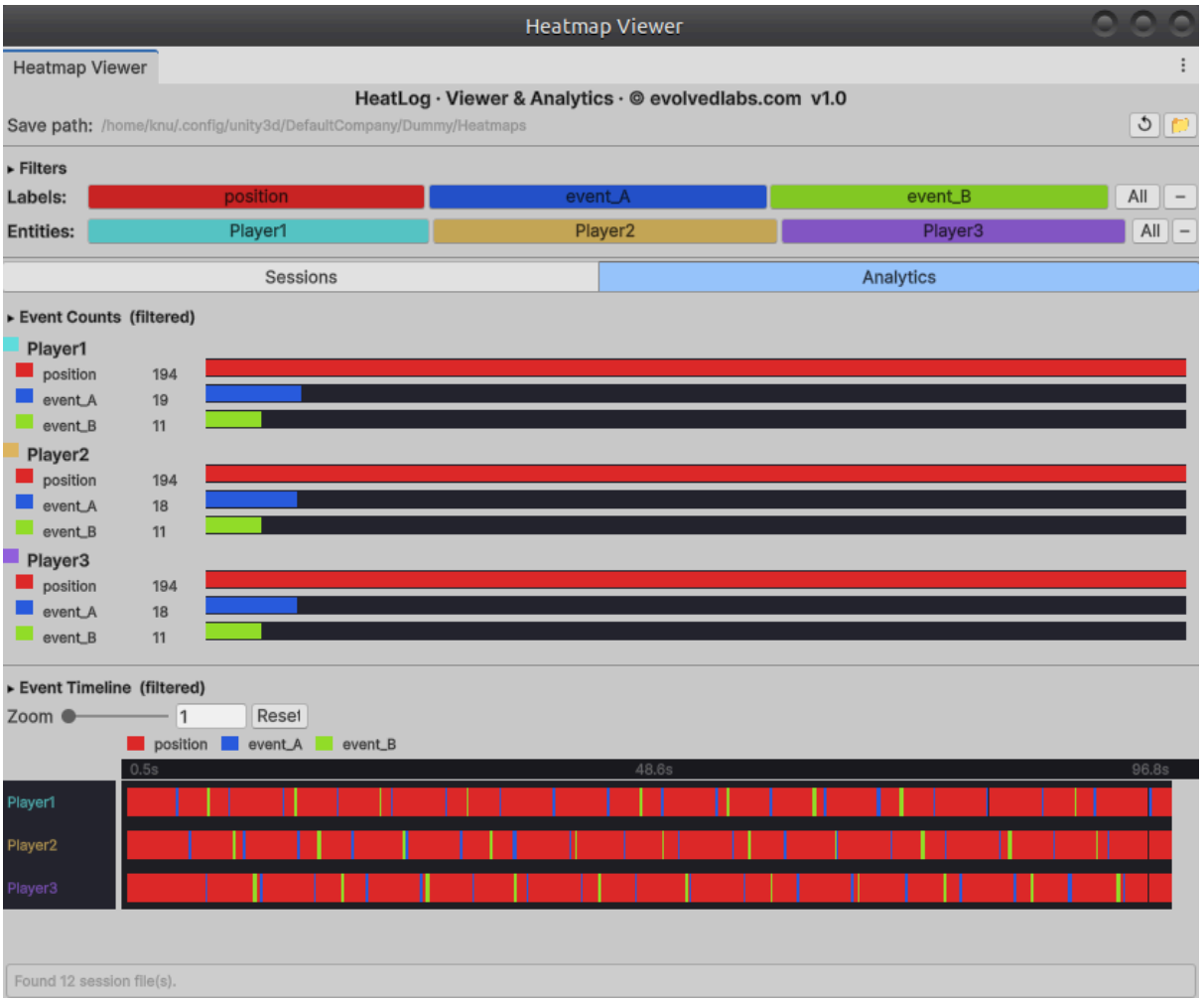
Scene overlay
Toggles the visibility in the viewport with three different options:

- 'Heat': blue → green → red density gradient. Legend shows the gradient strip.
- 'By Entity': each entity gets a fixed colour.
- 'By Label': each event label gets a fixed colour.

The label can be shown in the scene as well, although it can get quite overwhelming in very densely populated scenes.

Global / Label Style
This section allows to define how the distribution and global style of the events are represented. The most crucial one is the “Cell” slider, as it changes how nearby events should be packed together. Similarly, the other options allow to customize the appearance in the viewport.

The Analytics Tab



This tab allows for a high-level view of the events timelines, offering a simple bar chart of the events for each entity, as we all as a proper timeline that shows what-happened-when.

Note that you can zoom and hover and click for more information on that specific event.

Programming Documentation

Logging is done by calling the static method Log of the HeatmapEvent class, like so

HeatmapEvent.Log(string iEventName, Vector3 iPosting, string iString);

For example

```
HeatmapEvent.Log("event_B", player.mGo.transform.position, player.mEntityId);
```

Attach (or spawn programmatically) a HeatmapRecorderBehaviour component on your entities as seen in the demo scene. That's basically it!

Note how the HeatLogManager class allows you to pick if you'd like to save the file session as .json or binary (.heatlog) file. That same class also allows you to choose how to save the data (buffered or streaming) as well as the option to also verbose log in the console.

More importantly, the HeatLogManager class also allows you to specify where the session files are saved.

If you want to replicate the demo scene:

1. Create an empty scene in Unity.
2. Create an *empty GameObject* (right-click Hierarchy → Create Empty).
3. Add the *HeatmapDemo* component to it and *HeatLogManager*
4. That's it! Hit Play.

The demo will:
- Spawn a flat plane as ground.
- Spawn capsules as the "player".
- Attach `HeatmapRecorderBehaviour` to the capsule sautomatically.
- Move the player randomly across the area.
- Log a random events 5 seconds via `HeatmapEvent.Log`.

When you stop Play Mode, the session is saved automatically to:

Application.persistentDataPath/Heatmaps/session_YYYY-MM-DD_HH-mm-ss.json

Credits and acknowledgements

Check out our Unity3D Asset Store page for more assets!

➤➤➤ https://assetstore.unity.com/publishers/116354

Troubleshooting and support

The full source code in C# is available for you to extend functionality. The example provided here is just a quick way to showcase what's possible and you're expected to extend / write your own code in order to trigger the cut or shatter behaviour as you see fit for your game.

If you have questions or you'd like to report a bug, please reach out to staff@evolvedlabs.com