Unity Heat Map Plug-in

Game Engines E2013

Jakob Melnyk, jmel@itu.dk

December 11, 2013

Contents

1 Introduction

2 Project Overview

2.1 Goal of the project

2.2 Plug-in features

3 How To Use

The heat mapping plug-in consists of two parts: tracking objects and generating the heat map.

To access the plug-in, the custom package must be imported. The package consists of 4 folders. *Scripts/HM_Tracker* and *Prefabs/HM_HeatMap* are the elements used by the developer to create heat maps. The rest of the files should normally not be used.

3.1 Tracking game objects

To track a game object, the *HM_Tracker* script must be attached to the object. Attaching the *HM_Tracker* to an object allows the user to change the tracking interval, save interval and which events that are toggled.
The tracker can record the position of the game object when the following events happen: Breadcrumb (object position at intervals), Awake, onDestroy, OnMouseUp and OnTriggerEnter.

INSERT SCREENSHOT OF TRACKER OPTIONS

The OnMouseUp event is only tracked on every call to Update on the game object. Assuming 30 calls to update per second (30 fps), the tracker will register a maximum of 30 mouse up events per second.

The tracking interval determines the minimum interval between every breadcrumb in seconds. This value cannot be less than one second[1].

The save interval is the minimum time between the positions being saved to file[2]. If a game uses a lot of memory, a lower value here would mean less memory is used, but would also mean that the harddrive is accessed more often.

The tracked data is saved in a file named "HeatMapData DATE TIME.xml" in a folder called "HeatMapData". Tracked data for individual game objects are saved with the file name "HeatMap-DataNameOfGameObject.xml" in folder "HeatMapData/DATE TIME/".

Because we track individual game objects in this manner, the names of tracked game objects must be unique. While this issue can be dealt with by the user of the plug-in, this could certainly be an element to improve in a future version of the plug-in.

[1]If the value was lower, it could become a memory hog and take too long to save event info.

[2]Like the tracking interval, this value cannot be less than 1. We felt that saving event information more often than breadcrumbs can be tracked does not make sense.

3.1.1 Custom events

There are two ways to track custom events.

The first way is to create a separate script that calls the AddEvent method of the *HM_Tracker* component on the game object whenever the event happens. This is how we designed custom events to be tracked.

The second way is to modify the *HM_tracker* script, so that the event is tracked from inside the script itself.

3.2 Generating the heat map

A copy of the *Heatmapping/Prefab/HM_HeatMap* prefab should be part of the unity scene hierarchy. The prefab has a script attached that allows for generating heat map visuals. The *Heat Marker* variable of the script should not be changed.

INSERT SCREENSHOT OF HEATMAP OPTIONS

Heat Marker Scale determines the scale of the sphere surrounding a tracked event. *Allowed Distance* determines how far away two positions can be from each other before they count as increased density.

Session Data lets the user choose from different datasets in the "HeatMapData" folder. Currently, it is only possible to load data files from the "HeatMapData" folder.
Once a data file has been loaded, lists of the objects tracked in the file and their events appear in the editor. Selecting a specific object and/or event type will only heat markers for those objects/events in the scene.

Clicking the *Generate Heatmap* button instantiates heat markers based on the selected data file and the game object / event choices. After the objects are instantiated, the density of the objects are calculated and the colours of the heat markers are set based on their density relative to the highest density.

The *Clear Heatmap Markers* button destroys all objects in the same with the tag "HeatMarker".

3.2.1 Loading data from more files

Currently the only way to load data from more than one file is to load (and generate) the heat map data one file at a time. This is highly inconvenient, if there are large data sets (as there often is with heat map data. I discuss possible solutions to this problem in section 6.1 on page 4.

4 Overview of the Plug-in

5 Test

We tested our plug-in on three different games. An example top-down shooter game called Angry Bots[3], an example 2D-platformer game[4] and a first person exploratory game called Hiraeth we made for a different course. We tested these three types of games to see how our heat map plug-in

[3]Source: https://www.assetstore.unity3d.com/#/content/12175
[4]Source: https://www.assetstore.unity3d.com/#/content/11228

worked in different game types. In addition to testing the three different game types, we also the tested the performance of our plug-in.

The test was performed on a laptop with 8GB memory, Nvidia Geforce GTX 765M and Intel Core i7 processor (2.4 GHz).

5.1 Tested games

5.1.1 Hiraeth

Hiraeth features a relatively large square terrain with different height levels. The game was made for the Game Design E2013 course at IT-University of Copenhagen. The plug-in worked as intended. I have included an example screenshot to show the result of about heat mapping about 1.5 hours of playtime.

INSERT SCREENSHOT OF HIRAETH

5.1.2 Angry Bots

The Angry Bots game is a top-down shooter where the player moves around and shoots robots. When we tested, we successfully registered breadcrumb events and custom death events for the player game object. When we generated the heat map, the heat markers were very hard to see due to the transparency. Giving the the developer control over the transparency of the heat marker material could be an improvement for cases like this.

INSERT SCREENSHOT OF ANGRY BOTS

INSERT SCREENSHOT WITH CLEARER MARKERS

5.1.3 Example 2D-platformer

Tracking the player object worked as expected in the 2D-platformer game, but the rendering of the heat map did not. When we render the heat map, we use a transparent shader to render the 3D-spheres. The 2D-platformer does not support the use of the standard transparent shader of Unity, which means that the objects are not rendered.

5.2 Performance test

We tested the performance of our plug-in on our own Hiraeth game. This game is the one we had collected most data on, so we decided we could better evaluate the results when testing with this game.

5.2.1 Tracking/saving event data

We tested how the much the tracking of events impacted the performance of our game by playing with no objects being tracking and with different amounts of objects being tracked.

Results INSERT TABLE

As shown by the table, even with very many objects tracked at a time (far more than what we would assume is necessary for most games), we could see no impact on the performance of the game. I conclude from that result that the saving/tracking of event data does not need to improved from a performance perspective.

5.2.2 Generating the heat map

Results

6 Issues

In this section I describe the various issues with the implementation of our plug-in.

6.1 Loading data from more files

6.2 Heat map generation performance

6.3 Unique game object names

As I mention in section 3.1, the names of tracked game objects must be unique. If they are not, an input/output exception occurs and data is not properly recorded.
To fix this, I would suggest adding a check to see if there is a similar named object being tracked. If there is, create a new file with the same name except with a number at the end.
Because the render check for all events in a node with a specific name, it does not matter if the objects with the same name end up in different XML nodes when combined on game close.

6.4 Lack of Unity 2D-tool support

With Unity 4.3 came highly improved support for 2D-platformer games[5]. As mentioned in section 5.1.3, our plug-in is not very compatible with the 2D tools provided by Unity. Although the positions were tracked, we were not able to render the semi-transparent heat markers used to render the heat map. A possible solution to this problem could be to make the heat markers completely solid.

7 Conclusion

7.1 Future Work

In addition to the points I describe in section 6, I would consider implementing an option to generate a binned heat map instead of the 3D spherical version we used for this project. Additionally I would implement a method to reduce the amount of elements in the scene when generating the heat map.

Binned heatmap A binned heat map is when the heat markers in the world are predetermined. Every tracked position that falls inside an area (e.g. a 1x1 square) increases the density of that square by 1.

A binned heat map would take less time to generate than our 3D-spherical version. 3D-spherical version runs at O(N^2) and binned heat map would be O(N+M), where *N=number of tracked events* and *M = number of bins.*

Binned heat maps are generally viewed in a top-down 2D-perspective. This makes them quite good for games / levels that do not have multiple floors.

BINNED HEAT MAP EXAMPLE

[5]Source: http://blogs.unity3d.com/2013/08/28/unity-native-2d-tools/

[6]

Reducing number of objects in the scene The current implementation of our heat map plug-in generates one *HeatMarker* per tracked event that is loaded. This means that if there is a lot of data, there will be very many game objects in the scene. This can slow down the Unity engine (if not cause Out of Memory exceptions), which can be problematic.

I would suggest implementing the following method for combining tracked events: If two or more event positions are within allowed range, combine them to one game object and calculate the middle of their positions and use this for the new position.

This would not work in all cases, but it could potentially lead to a very high reduction of the amount of unnecessary objects in the scene.

[6]Source: Game Engines E2013 course Power Point slides: 13_metrics_final.ppt