Heat Mapping Unity Plugin

Game Engines E2013

IT-University of Copenhagen

Jacob Claudius Grooss, jcgr@itu.dk

December 11th, 2013

1 Introduction

This report is the result of a project in the Game Engines E2013 (MGAE-E2013) course on the Games line at the IT-University of Copenhagen autumn 2013.

For the project, I have worked with Jakob Melnyk (jmel) to create a plugin for Unity that allows the user to gather data for their games and create heat maps based on this data.

Throughout this report I will be referring to two games, in which we have tested the plugin: AngryBots which is a demo project for Unity and Hiaerth which is a game we have created for the Game Design course[1].

2 The Project

When it comes to video games, there is a lot of data available from when people are playing and all of it can be tracked. Data that involves the positions of objects, where entities died (and what they died to), where certain events were performed, etc. can also be very interesting for the developers. It can be used for statistics, influencing balance, finding bottlenecks and figure out what is interesting for the players.

While the data is important, displaying it in an easy-to-understand manner is at least as important. Using an FPS[2] game as an example, showing what weapons the players prefer to use overall, their accuracy with weapons, which enemies are killed with which weapon, etc. can be visualized properly with a table. It works because the data is numbers and the numbers are not necessarily related to where in the world the even takes place.

When it comes to positioning of any kind in a game, a table is not a good choice. A table is useful for saying how many times an event happened, but it is not very helpful when it comes to stating *where* the event happened. "Event e has happened at location x, y, z so and so many times" gives some information, but where in the game is x, y, z?

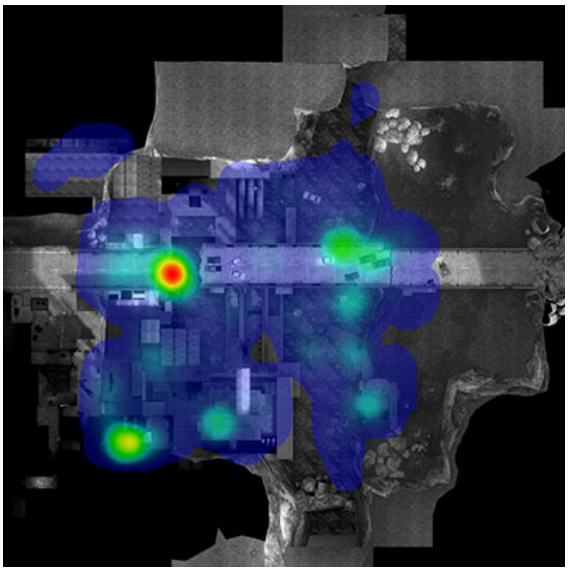

Figure 1: *A heat map representing deaths in a level in Half-Life 2*

[1]The entire Unity project can be found at https://github.com/esfdk/DarkForest
[2]First Person Shooter

This is where heat maps have their place. Using colors, they can show where - and how many times - an event has happened in the game. Heat maps are maps of the different places of the game with colored spots on. The colors of the spots range from an icy blue to a dark red. The darker the color is, the more times a certain event has happened there.

Looking at **figure 1** on page 1, one place is bright red, which means that players have died a lot at that position. This can either indicate that the place is too hard, that the players have missed something that can heighten their survival, or that the place is working as an intended difficult area. Either way, the heat map gives the developers useful information, which is easy to understand and work with.

2.1 Goal of the Project

As both Jakob and I are rather interested in game balance and mechanics, we decided we wanted to make a tool that could assist in generating heat maps. After talking it through, we decided to create the tool for Unity, for a couple of reasons.

Unity works on a set of rules. Every object in the game world has a position and every object has access to methods that are the same for all objects (such as the Update() method, which is called every frame on every object). This means we can assume that every object can do certain things, and base our tool on that.

Another nice things is that plugins in Unity are actually just collections of scripts and prefabricated objects. As such, it is easy to export and import in another game. Copy the necessary scripts and objects, and you have a working heat map tool for another game.

Unity is also fairly easy to work with, when it comes to creating new tools for it. The API used in Unity is solid, and gives access to basically anything one will need. The documentation and support for Unity is also extensive, making it easy to figure out what one can do and how to do it.

2.2 Features

Having decided to write the tool for Unity, we decided on the features there should be in the tool:

- **Track anything** - The tool should be able to track any object, no matter how simple or complex the object is.
- **Generate heat map on event basis** - The tool should be able to generate a heat map based only on certain events. If all the tracked events are shown at the same time, the heat map will practically be useless.
- **Easy to use** - Finding a tool that seems to cover all your needs, just to figure out that it takes ages to set up properly is never fun.
- **Stand-alone** - Gathering and processing of data should not depend on other systems, or even connection to the internet. It should be able to work as long as the game is running.

These features were the must-have of the tool, as without them it would not be able to properly gather and visualize data.

3 Overview of the Plugin

With the features described in section 2.2, we started working on the heat mapping tool. We started with very basic tracking and visualizing, and built it up from there. In the end, the plugin ended up being able to gather data for any event on an object. The gathered data is stored in an XML file and can be used inside Unity to create a basic heat map for the game.

The plugin can, by default, track some default Unity events such as the position of an object. Furthermore, one can track custom events, allowing the developers to gather virtually any kind of data they want. The heat map generation will take the custom events into account, and allow generation of heat maps of the custom events.

As seen in **figure 2**, the heat map overlay generated by the plugin is of a decent quality, and certainly good enough to learn something about the players' behaviour from. As one can see, the player just walked around until they got to something interesting, at which point they stayed for a little while, investigating it.

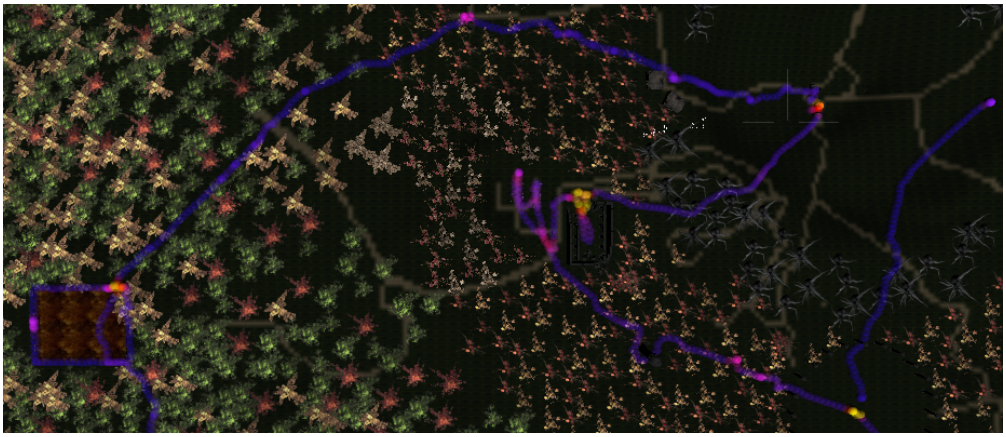

Figure 2: *A heat map showing player movement in the game Hiraeth.*

3.1 How To Use

4 Performance

4.1 Saving data

4.2 Generating the heat map

5 Conclusion

5.1 Issues

Custom events from javascript is not easy.

The transparent spheres do not work with the new 2D tools in 4.3. Transparent objects appear to be totally invisible.