Proposal For Google Summer of Code 2014

JavaScript Implementation of liquid-galaxy

Name: Ashraya Bhat K S
Mobile: +918722871967
Email Address: ashray1993@gmail.com
Freenode IRC Nick: _ashray_
Location: Bangalore, India UTC +0530
Primary Language: English

About Liquid-Galaxy:

Liquid-galaxy is a cluster of computers running Google earth and creates an immersive experience. Liquid-Galaxy allows you to fly all over the globe with instant zoom-in out and around. It lets the users to experience the rich imagery of Google earth in different views. With Google Street view,it brings a totally new experience and it has the potential to change the way the people use Google Maps/Google earth.

Motivation for Proposal/Goal:

The project aims to implement the existing liquid-galaxy in JavaScript. The current implementation follows master-slave architecture. All the computers in the local network have the same broadcast address. In master's drivers.ini file if the ViewSync/hostname is set to the broadcast address of the local network. Now the packets get delivered to all the systems in the network and all the systems are configured to get the view synchronisation.

This project implements the same functionalities in JavaScript. The fundamental question that arises here is why it should be implemented with JavaScript. The advantage of JavaScript implementation is, it doesn't require any external (Other than js) libraries, add-ons or frameworks for its functionality. Once if the liquid-galaxy is implemented in JavaScript , installation and configuration becomes easier so that even-non technical people can also configure and use liquid-galaxy with minimum effort.

One other advantage is JavaScript implementation makes it platform independent. All that is required is a browser! Users can have great experience of immersive Google earth without needing to install anything that is specific to their operating system. These advantages that we have in JavaScript implementation are a strong driving force for the project.

Use Cases:

1. Mr.Ramesh is real-estate businessman. He deals with "high-end" clients. Mr.Ramesh wants save time without compromising in the experience his users used to get before. Here comes the Liquid-Galaxy, providing them the best experience with 3D building models and complete Google earth data. This is one thing that can Mr.Ramesh could relay on for providing experience closest to real-life experience. If Mr.Ramesh is unaware of the technical details of implementation, "JavaScript implementation of the Liquid-Galaxy" would come in handy for him. All he needs to do is to type the URL generated in the browser!

2. Ms.Saritha works for a multinational company in United States of America. Everyone in her family is curious about the country they never been to; The United States of America. Her family could get a best experience with Liquid-Galaxy. People like them want a simple but efficient interface. ”JavaScript implementation of Liquid-Galaxy” serves that purpose.

Expected Results:

· The implementation should be simple and easy with respect to the end user. The computer in which the application first started should generate an URL. Upon entering that URL in slave machines, the machines should be ready to transfer data between them. This acts like a handshake. This URL can be shared by any other means of communication like Gmail or Google hangouts.

· Master computer is one in which an event of click, drag, zoom-in or zoom-out occurs. Once the connection is established, any computer in the connected network can be a master. Every computer in the network should respond to the event occurred in any other computer in the network.

· The computers in the network should operate with as minimum latency as possible.

· As the number of computers added to the network, it should automatically adjust the views. The views in the two computers in the same network shouldn’t overlap. At the same time views in the different computers should be properly related to each other and continuous.

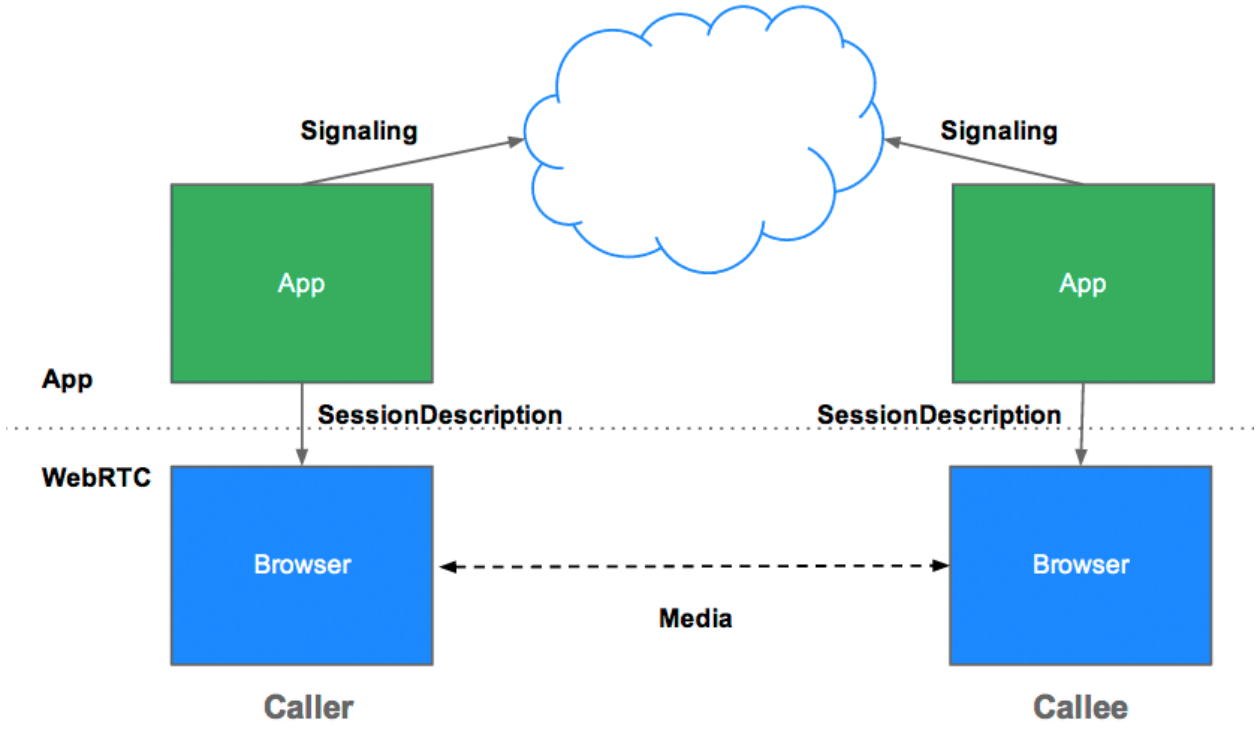
Challenges:

1. Creating a handshake between all computers in the network
2. Detecting click, drag, zoom-in and out.
3. Sending the Latitude and Longitude that corresponds to the event
4. Achieving minimum latency
5. Dealing with NATs. What if the computers are behind a NAT
6. Incorporating intelligence in creating the views in different machines. (Avoiding overlap and maintaining continuity)
7. Dealing with the situations where 1 computer suddenly disconnected.
8. Dealing with browser differences

Possible Ideas for Implementation:

For this application the latency is critical. So we must look into something with minimum latency. Web RTC enables the communication possible with lowest possible latency. And in future Web RTC looks very promising for these kinds of applications.

1. In the first place all the computers should agree to conduct the session. In web RTC it’s called signalling. It’s the exchange of session description objects. These session description objects contain all information like network information to set up data exchange. It can be embedded in the URL that is generated .This URL is to be exchanged and it can be done using other messaging services. It can also be exchanged as JSON objects.



Signalling

2. Google earth API provides number of events, which can be used with google.earth.addEventListener to provide additional functionalities. When click or drag event is occurred, new latitude and longitude should be fetched using getLatitude() and getLogitude() methods and zoom event is also captured.
3. The data that are captured are put into a JSON object and sent to the other computers. These computers use these data and using functions like setLatitude() and setLongitude() the Google earth is set to focus on that location. This is to be done for each event that occurs. These computers can implement some architecture for sharing data between them:
Mesh:
There is no server running and every computer has to send a copy of the data to all other computers. This is inefficient especially if one of the devices is a mobile device.
Star:
Here the most capable system amount the computers in the network are chosen and all the data is sent to this computer and then it is forwarded to other computers. This computer is called focus computer.
4. Web RTC enables applications to exchange data with minimum latency.
5. NATs handle private IP address and there is no way that we can use that for peer to peer data transfer. We somehow need to get the public IP address of the peers. The client asks it STUN server for public IP address. STUN server puts that information in a packet and sends it the clients that asked the public IP address.

This may not work all the time. For that we have technology called TURN. This provides a cloud fall back whenever direct peer to peer data transfer is impossible. It is like a relay. It gives a new IP address to the clients and since the relay is on the cloud, any computer can make connection. But it adds some cost to it. So we need to choose the best out of two. That is where technology called ICE comes in. It chooses the best of the two and makes connection always possible.

Testing servers:

STUN Server:

1.stun.google.com:19302

TURN Server:

1.rfc5766-turn-server :Has Amezon VM images

2.restund

6. Synchronising the view is also very important. The application should be intelligent enough to adjust the views so that it's continuous. It involves mathematical calculations to set correct offset for every new computer joining the Liquid-Galaxy network.
7. In the case of sudden and unexpected connection interruptions the system should be capable of handling these conditions.
8. Web RTC is compatible with all the browser. There is something called adopter.js which implements the latest specifications and then functions down to whatever it supports.

Roadmap:

I am available for the entire GSoC period.I have my semester end-exams between 20th -29th May. I have no other commitments / plans. I am prepared to work for 40 hours a week, and more, if that is what is required to finish the task.

22nd April – 05th may:

· Bond with the community. Tweak with the code. Build a close relationship with the community.
· Understanding the current implementation of Liquid-Galaxy.
· Also working with the required Google earth APIs.

06th May – 19th May:

· Understanding STUN, TURN servers and getting familiarized with ICE.

20th May – 29th May:

· Discussing with mentors.Implementing the synchronisation with 2 systems.

30th may – 7th June:

· Working out about the Mathematical aspects of synchronisation. It includes the decision making regarding views when new computer is suddenly added etc.

13th June – 25nd June:

· Implementing view synchronisation with multiple computers within local network. Fixing bugs of the code written so far.

26th June – 7th July:

· Implementing Liquid-Galaxy through NATs using STUN, TURN Servers. Analysing performance of the implementation.

8th July – 17 July:

· Implementing intelligence into the system in intelligently expand views when new system is joined.

18th July – 23rd July:

· Checking for unhandled errors.

· Handling conditions when a system in the liquid-Galaxy is disconnected.

24th July – 02nd August:

· Building a testing Unit and testing the application thoroughly for errors.

· Asking mentors suggestion for finalizing the application

03rd August – End:

· Improving the efficiency of the implementation. Looking at the other projects that the community is undertaking and contributing for them.

About me:

I am a student pursuing an undergraduate degree (currently 3rd year) with Information Science and Engineering as major in PES University (previously known as PES Institute of Technology), Bangalore.

I was introduced to open-source philosophies 2 years back. Since then I have been a keen follower of FOSS philosophies. I haven't attended any major open-source event but I am regular to my college's Open Source community.

Recently I have developed a video chatting application using Web RTC as an academic project. This helped me to get a good hold on Web RTC. Also I have done couple of web related projects as academic projects. The following are the courses I have taken so far.

1.Physics and Chemistry	2.Mechanics (Statics)
3.Mathematics I and II	4.Methods of Applied Mathematics
5Linear Algebra	6.Discrete Mathematics and Combinatorics
7.Introduction to Computer Programming	8.Finite Automata and Formal Languages
9.Data Structures	10.Principles of Programming Languages
11.Computer Architecture and Organization	12.Analysis and Design of Algorithms
13.Programming through C++	14.Logic Design and Analysis
15.Database Management System	16.Introduction to Microprocessors
17.Compiler Design	18.Advanced Java
19.Computer Networks	20.Unix System Programming

21.Web Technologies	22.Software Engineering

Other than academics I have participated in PES-SoC program (by the PESOS community) under which I developed a portal for collaborative placement training for college students. I have been participating in various hackathons conducted at various colleges and companies in Bangalore .I have volunteered for hackathons conducted in our college.

Why me?

1. A good grasp over HTML,HTML5,JavaScript,Python,PHP
2. Capability of hacking into the existing feature of the system and getting the job done.
3. Creative in implementing ideas.
4. Dedicated and Hardworking
5. Passionate about web development.
6. Lastly, a pure unadulterated love for coding.

I assure that I will give around 40 hours per week for my work Other than my examination time which is during 20th May to 29th May. During this time may not be able to invest more than 4 hours a day. If any part of my application is not clear, I will be very happy to clarify it.

References:

1. https://developers.google.com/web-search/docs/
2. https://developers.google.com/earth/documentation/events
3. http://liquidgalaxy.endpoint.com/
4. https://developers.google.com/maps/documentation/android/interactivity
5. http://www.html5rocks.com/en/tutorials/webrtc/basics/