

Everything Everywhere

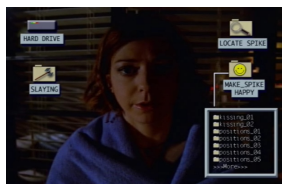
Proposal for a AR Network system based on existing protocols and infrastructure

The following paper is my vision of a open AR Network and potential methods to implement it with existing technologies.

Specifically I'll be focusing on a potential for a global outdoor AR network, although the ideas aren't limited to that.

Of course I call it "my" vision, but I'm obviously not the first to have many of these ideas.

I have been influenced and inspired by many things...



The AR Network.

When I speak of a future AR Network, I mean one as universal and as standard as the internet. One where people can connect from any number of devices, *and without additional downloads*, experience the majority of the content.

Where people can just point their phone, webcam, or pair of AR glass's anywhere were a virtual object should be, and they will see it. The user experience is seamless, AR comes to them without them needing to "prepare" their device for it.

From this point forward, I will refer to this future AR Network simple as the "Arn".

The Arn should be an inclusive, and open platform where any number of devices can connect too, and anyone can make and host their own location-specific models or data. It should allow people to communicate both publicly and privately, and not have their vision constantly cluttered with things they don't want to see.

There's two old, existing paradigms that I think can help reach this goal when they are combined.

The Internet Relay Photoshop.

IRC, or Internet Relay Chat was a chat system designed by Jarkko Oikarinen in the late 80's.

Its a system where people meet on "channels", they can talk in groups, or privately. Channels can be read-only, or open to all to contribute too. There is no restriction to the number of people that can participate in a given discussion, or the number of channels that can be formed. All servers are interconnected and pass messages from user to user over the network.

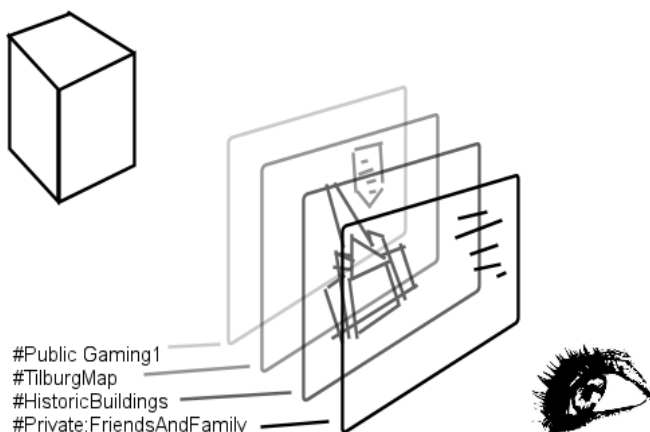
To me, this relatively old internet technology is a great template, or even foundation, for how the Arn could operate. Rather than text being exchanged, it would be mesh data (or links to mesh data), but other than that much of the same principles could apply.

People could join channels of information to view or contribute. Families could leave messages to each other scribbled in mid-air on private channels. Strangers can watch AR games being played between people in parks. People going into a restaurant could see the comments from recent guests hovering by the menu items. None of this would have to be called up specially, if they are on the right channel when it was broadcast, they will see it.

The IRC paradigm becomes particularly powerful when combined with another one common to many computer users; that of a “Layer” in an art program, such as Photoshop or Paint Shop Pro.

As most of us know, layers allow us to separate out different components of a piece of art while editing, either to focus our attention on one piece, or to make future editing easier.

Now what if we simply have each “channel” of information represented as a layer?



Having channels corresponding to layers is an easy and intuitive way for the Arn to operate. The user can login and contribute data to any channel, like IRC as well as adjusting the desired opacity and visual range of each layer, like they would a layer in Photoshop.

In this way they can get a custom view of the world, both with shared and personal AR elements visible at the same time.

They would not have to switch between various overlays to their world view, as they could see many at the same time.

Persistence of Data

While IRC itself has many suitable property's, one obvious downside is its lack of persistence. Data you right into it is transient and can't be recalled.

Its for this reason, XMPP could be a more desirable protocol. Specifically, the WaveFederationProtocol would allow federation, persistence, as well as realtime updating. Their may well be alternatives too that have these desirable property's.

The Data.

Whatever the standard chosen for both server to server, and server to client, communication, the data sent itself would of course also have to be standardised.

For instance, this standard could be as simple as a XML string pointing to a KML file on a sever. This could then be then displayed in the users field of view at the co-ordinates specified within the XML's attributes.

On a more abstract level, the data would essential be a set of key/value pairs, tying virtual data to the real world.

There is also no reason why this shared-space/personal spaces based on channels of data has to be restricted to things given absolute co-ordinates.



If markers are designed with URL data in them, this could even be a prompted or automatic process.

“There is visual data in this area on the following channel; #ABCD would you like to view this channel?”

<http://www.darkflame.co.uk/ArnLinkTest/Church.kml>



(Different ways to access the same mesh)

It could work just as well with Markers and thus relative co-ordinates.

This would be mostly useful for indoor use, letting people logged onto a channel see the same meshes as everyone else on the markers.

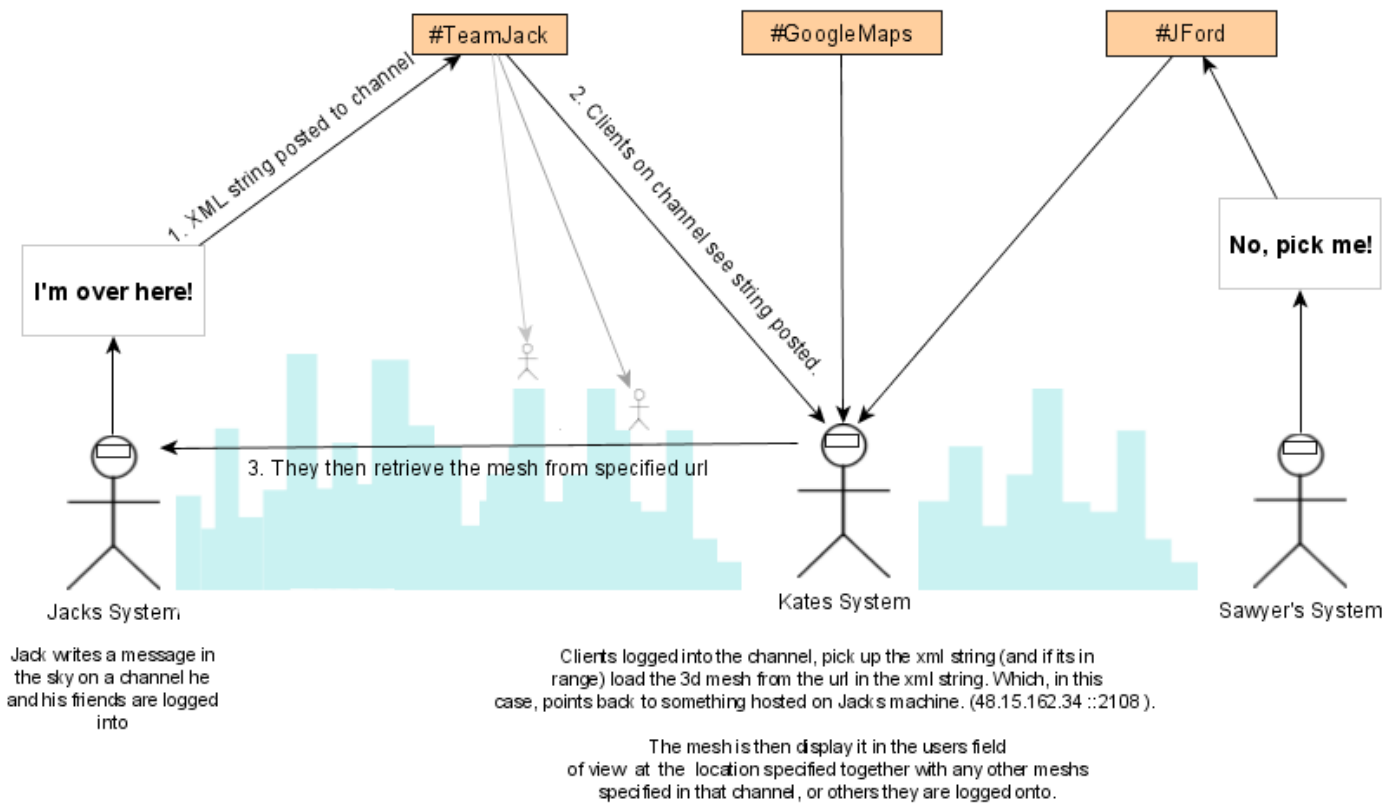
Thus allowing multi-player AR games, or AR games with observers very easily.

For example; games like Chess could be played between people with no additional code needed; You simply have a set of markers for only your own pieces, and as you move them the channel updates with the new positions, which are displayed in place in your opponents field of view.

This sort of game comes “free” with just having a generic system of shared space supporting markers.

It would also allow AR adverts down the street or in magazines to be viewed by simply logging onto the right AR channel.

An example of how collaborative 3D-spaces could be shared;



While in the long run I would hope for a dedicated AR network to be developed, with greater flexibility with persistence of data, there is a lot that can be done even with the existing IRC system to implement the ideas mentioned above.

Below I will show an example of simple, crude, pseudo-protocol that could be fairly easily implemented to create shared AR spaces broadcast across existing IRC channels and servers.

Its important to note, the goal here isn't to exchange the mesh data itself on IRC, its to exchange links to the data. Exchanging the mesh data directly within the 500 character IRC limit would be very hard, and liable to errors.

It's also a waste of network bandwidth, as many people logged onto the channel might not have that object in their field of view, so their clients should not bother downloading it. (it should be up to the client browsers when to anticipate and cache mesh data).

Example Basic link exchange for AR;

Principle;

As user creates or changes an object, the clients software posts a simple set of key value pairs (in this case as xml) to the protocols channel. (in this case IRC) Anyone logged into that channel then sees that mesh displayed in the specified location.

This string could be formatted as follows;

```
<Mesh
ID="DARKFLAME:1"
Obj="http://www.darkflame.co.uk/mesh/church/chuch.kml"
Loc="(49.5000123,-123.5000123)"
Permissions="None"
LastUpdate="12/12/0000,2012:12"
/>
```

This string allows other users client logged into the channel to automatically load the object from the URL and display it at the correct position in their field of view.

If the permissions are set to allow it, they could then move the object themselves, with the update being feeding back seamlessly to other users on the channel.

An example of how collaborative 3D-spaces could be shared;

The objects posted are given an ID, which can be just the posters name, followed by a unique object number for that name. These unique ID's would allow clients to track different instances of the same mesh, as well as making it easy to implement permissions. (if only the poster should be allowed to move this object, then the clients simply check if ID matches the user name posting the update. If its not, they can ignore it).

Next the objects need to be linked to a mesh.

The location of the objects mesh doesn't have to be a fixed remotely-hosted url, it could be an IP address and port number of the user posting the mesh, hosted by the application posting the link to the channel.

```
Obj="www.darkflame.co.uk/mesh/church/chuch.kml"  
Obj="123,223,14,23::3030"
```

The objects co-ordinates, likewise, need not be specified as absolute gps co-ordinates, but instead could refer to generic Marker.

```
Loc="(49.5000123,-123.5000123)"  
Loc="Marker1"  
Or relative to a marker;  
Loc="Marker4 (+0.0023,-0.0023)"  
Or relative to a default plane;  
Loc="Default(+0.213,-0.123)"
```

The ARBrowsers could then handle the association between the Markers pattern and its Name.

This way the markers are reusable, they do need unique markers to be printed for every new bit of AR they want to look at.

Users could just keep a set of generic markers handy, which they could simply assign to be Marker1, Marker2 etc for any AR use.¹

The Default location could be a settable region, or marker, on the clients browser that defines a playable/user-able area in the field of view. Mostly useful for home use, this could typical be

¹ As mentioned above specific makers could also contain a default ID name and channel built into their data, letting the Arn browser simply prompt the user if they want to see the model even if they aren't in the right channel. This set up would be most useful for paper and even billboard advertising.)

a square region on a users desk.

So, in the chess-game example, the client of the person making the moves simply updates the position relative to the Default every time they move their marker.(which is tied to a chess piece mesh).

Then the (non-owners) clients software could automatically display it relative to their Default plane. This would make games like Chess, Checkers, Go or any other game involving merely moving objects about automatically very intuitive and easy to set up.

So by having meshes settable to absolute gps, marker-relative, or default-relative locations, the the bother necessary to experience AR content quite considerably, and makes "non-geo-specific" AR applications and games trivial to implement.

Next is permissions.

Mesh-permissions would be a simple string saying who else can update the data, if anyone.

```
eg;  
Permissions="None"  
Permissions="RandomPerson1, RandomPerson2"  
Permissions="All"
```

By default you could only update or move your own meshes. (identified by the ID of first posting). If you attempt to update anyone else's, their clients would just ignore it.

Thus in a game of chess, you can only move your own pieces. If you attempted to move your opponents (by reassigning your own marker to their pieces Ids), the clients would just ignore that assignment. You'd only be fooling your own system.

Likewise, when pinning a message in mid-air for your friends to read, no one else can change that message without your permission.

(although copying it would be easy).²

Finally, as object data could change within all sorts of time-scales, the easiest way to keep

² It's important to note this sort of object-specific permission system is in addition to the global-permissions, or "user-modes" it's possible to set for the IRC channels and users as a whole.

An example of how collaborative 3D-spaces could be shared;

everyone logged in up to date is to just have a time-stamp of when each model was last updated.

`LastUpdate="12/12/0000,2012:12"`

This would not necessarily be the same as the XML string post date, because the models mesh might not be updated, but merely moved, and in such case the Arn browser shouldn't redownload the mesh.

This sort of arrangement could be used as a standard today, and users wouldn't have to constantly download special AR programs to view single AR mesh.

In the long-term I would hope for more advanced methods to manipulate Arn-content online, analogous to Dom manipulation in web-pages. But for now, we should at least establish standard methods for devices to pull up meshes and overlay them in the correct position.

So, having a layered system could give the user a seamless blend of dynamic and static data with which to paint their world with. I believe this is all relatively easy to achieve using modifications of existing web technology, combined with some basic graphics systems.

Local Data:

However, so far I have only talked about remote data.

What of programs originating on the device itself? This is, after all, how most AR software we have at the moment works.

I think, that just like the remote channels, local software should also be blended into the same list of layers. People shouldn't have to "Alt+Tab" out of one view of the world, to see another.

They should be able to see *both at once*, if they wish.

For instance, if your playing a AR game, why shouldn't your chat window be viewable at the same time?

If you have skinned your environment with a custom view of the world, why shouldn't you also see mapping or restaurant recommendations?

So local data and remote data should be blended in the same view.

How can AR software - of which I hope, there will be thousands - seamlessly be expected to layer their graphics, not only with the real world, but with each other, and with online data too? Will games and software makers need to co-operate to allow their graphics to be integrated together with correct occlusion taken into account? A tall order, no?

I must confess though, my technology knowledge fails me here.

I can only guess special graphics drivers, or 3D APIs, will have to be developed to let programs share their 3D world with that of a Arn browser. Maybe programmes should simply treat themselves as a local-sever which the browser can connect too, and let the Arn handle all the rendering itself (although I imagine many games designers would find this quite limiting).

So I leave it as an exercise to the readers to discuss and propose the best methods by which this vision of a layered world could be realised..

Beyond IRC:

As mentioned before IRC has some drawbacks, which are due to its age or method of working. As such, XMPP based systems might yet prove better alternatives for a open AR network. One example of such a system is Wave Federation Procol.

It shares many of the advantages of IRC (open, anyone can create a channel of data, different permission levels can be set and its free), while avoiding some critical restrictions. (The data can be persistent).

I believe some of the ideas I've mentioned, and possibly even the proposed protocol string could be adapted for WFP or for similar federated systems.

I believe overall the principles are more important than any specific implementation to

An example of how collaborative 3D-spaces could be shared;

get to them;

Summery;

- In order for AR to flourish the user shouldn't need to download a separate application for each mesh they want to see.
- Having url's embedded into QR coded markers which point to standard mesh files like dxf or kml would be a way to do this *right now*. The QR code would only have to be seen precisely in shot once, then its borders could be used like a standard marker.
- An augmented view of the world needs to support visual multitasking, and having layers of information is the best way to do that.
- Methods need to be devised to allow drastically different software to contribute to these layers, without restricting either the software's rendering ability's, or the users ability to pick and choose what layers of information he wants to see.

Last point;

I am absolutely confident in my belief AR will become at least as important as the web has, and probably a lot more so. It will also face much the same hurdles and challenges getting established as that medium did.

But, speaking as a web-developer, can we try to avoid a browser war this time?

Everything Everywhere , updated draft for;
**International AR Standards Workshop-October
11-12, 2010**
by Thomas Wrobel
thomas@lostagain.nl