The Aotearoa Digital Arts Reader
Edited by Stella Brennan and Su Ballard
Designed by Jonty Valentine

ISBN: 978-0-9582789-9-7

A catalogue record for this book is available from The National Library of New Zealand

Title: The Aotearoa Digital Arts Reader
Author/Contributor: Brennan, Stella (ed); Ballard, Su (ed)
Publisher: Aotearoa Digital Arts and Clouds

Aotearoa Digital Arts Trust
www.aotearoadigitalarts.org.nz

Clouds
PO Box 68-187, Newton, Auckland 1145
Aotearoa New Zealand
www.clouds.co.nz

Every effort has been made to trace the copyright holders of the illustrations reproduced in this book. Unfortunately, this has not been possible in all cases. The editors and publisher would be pleased to hear from any copyright holders whom they have been unable to contact and to print due acknowledgement in subsequent editions.

Unless otherwise noted, all images are repoduced courtesy of the artists.

Editing a book takes a long time, and many people have helped along the way. Stella and Su would like firstly to thank the authors and artists who have contributed to this book, and the institutions and individuals who shared their image archives with us. We would also like to acknowledge the work and support of the following: Nova Paul, Leoni Schmidt, Col Fay, Khyla Russell, Justine Camp, Letitia Lam, Pam McKinlay, Geoff Noller, Sarah McMillan, Robert Leonard, Melinda Rackham, Mercedes Vincente, and Gwynneth Porter, Deborah Orum and Warren Olds from Clouds. And of course, Jonty Valentine for the hours spent in design. Thanks also to the ADA community, and especially to the other ADA trustees, Janine Randerson, Douglas Bagnall and Zita Joyce.

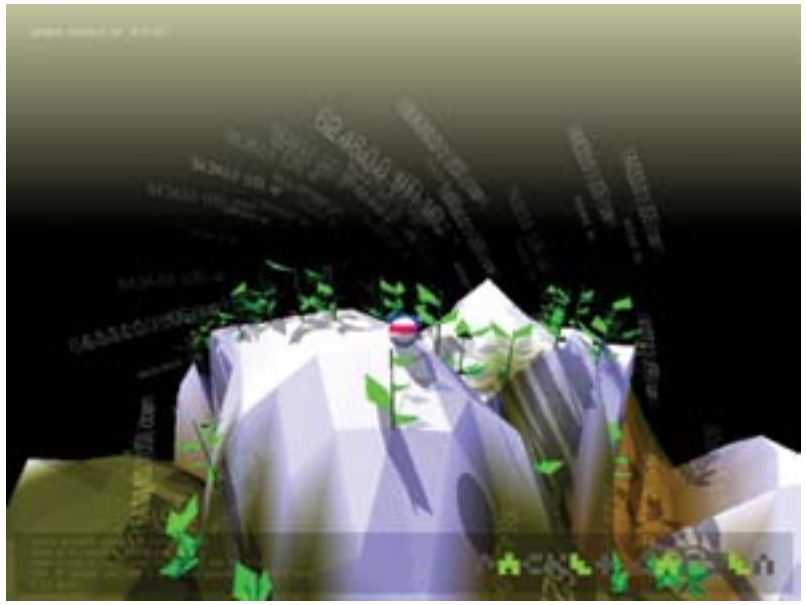
Thanks most of all to our families: Nathan, Moss and David.

The *Aotearoa Digital Arts Reader* would not have been realised without the support of AUT University, Otago Polytechnic and Creative New Zealand.

The Graph as Landscape: Reflections on Making *Packet Garden*

Julian Oliver

A computer with access to the Internet will typically touch hundreds, even thousands of remote computers a day. Even if not in active use, software running on a modern operating system will regularly query other computers for an update, an address or even just a simple 'are you alive?'. Over the last few years peer-to-peer software, instant messaging, mass-mailers, botnets and multi-player computer games have exponentially increased the amount of Internet traffic. *Packet Garden* maps this network contact, forming an accessible 'walk-in' graph, a geography produced through the accumulation of network events.

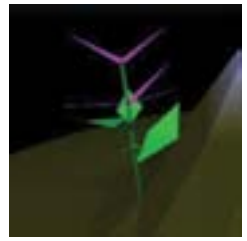


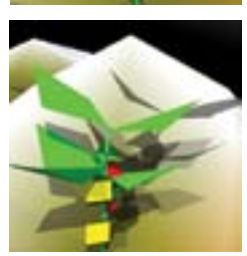
On a personal level, I set out to make *Packet Garden* to learn more about the Internet at the level of network messaging and to explore the metaphor of terrain as graph. Navigation of this terrain is a kind of reading: just as the undulations of the Earth's surface can be read as records of energetic events below and above the surface, I wanted the landscape to both contain and represent information. The key challenge of this world-as-graph experiment was in the design of a heuristic landscape: which data informs which element? The primary information I wanted to use in my graph—and so pass on to the user of the *Packet Garden*—was network address (IP), geographical location and a total of the data transferred.

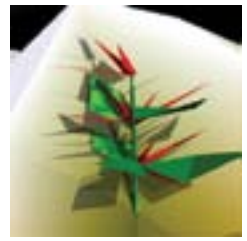
Before design could begin I needed to know which information I could feasibly capture on Linux, OSX and Windows systems using the same base of code. To have access to this information at all I needed to work at a low level, looking at each packet of data going over the network card and dissecting it in turn. To do this I wrote a tiny program to run on the user's machine that logged what went over the network card and collated the results in a way I could parse to generate a three-dimensional scene. In collecting this data I was stepping into a contentious area. The software I was writing was tracking some of the user's most sensitive data—their computer's network footprints.

Once this data was logged it had to be parsed so as to be meaningful within my geographical metaphor. I chose network load as my primary deforming force: uploads would make mountains and downloads valleys. In order to locate these formations and deformations, I needed to design a means to translate Internet addresses (IPs) into a system of three dimensional coordinates. Internet addresses currently use the IPv4 standard; IPs are comprised of four blocks of numbers between 0 and 255, separated by dots (the Aotearoa Digital Arts website has the IP '205.196.213.237', for instance). These numbers form unique numerical addresses for computers, to which we ascribe names known as domains. These domains are made available to querying software such as web browsers by Domain Name Servers. This makes finding data held on a specific


```
while 1:
      try:
        for ts, pkt in p:
          packet = dpkt.ethernet.Ethernet(pkt)
          if type(packet.data) == dpkt.ip.IP:
            source = struct.unpack('4B', packet.data.src)
            dest = struct.unpack('4B', packet.data.dst) # unpack 4 block struct type.
            length = packet.data.len
            source_port = packet.data.data.sport
            dest_port = packet.data.data.dport
          elif type(packet.data.data) == dpkt.ppp.PPP:
            source = struct.unpack('4B', packet.data.data.data.src)
            dest = struct.unpack('4B', packet.data.data.data.dst) # unpack 4 block struct type.
            length = packet.data.data.data.len
            dest_port = packet.data.data.data.data.dport
            source_port = packet.data.data.data.data.sport
          geo_source = self.geo_ip(source)
          geo_dest = self.geo_ip(dest)
          if type(packet.data.data) in (dpkt.tcp.TCP, dpkt.udp.UDP) or \
            type(packet.data.data.data.data) in (dpkt.tcp.TCP, dpkt.udp.UDP):
            count += 1
            if source == ip:
              dir = 'up'
            elif dest == ip:
              dir = 'down'
            for k, v in self.ports.iteritems():
              if source_port in v or dest_port in v:
                proto = k
                break
              else:
                proto = 'unknown'
            if (source[0], source[1], source[2]) == (b1, b2, b3) and source != ip:
              pass
            elif (source[0], source[1], source[2]) == (b1, b2, b3) and dest != ip:
              pass
            if source == ip:
              dest_net = str(dest[0]) +"." +str(dest[1]) +"." +str(dest[2]) +"." +str(dest[3])
              try:
                dict[str(dest_net) +", "  +dir +", " +proto +", " +geo_dest] += length
              except:
                dict[str(dest_net) +", "  +dir +", " +proto +", " +geo_dest] = length
            else:
              source_net = str(source[0]) +"." +str(source[1]) +"." +str(source[2]) +"." +str(source[3])
              try:
                dict[str(source_net) +", " +dir +", " +proto +", " +geo_source] += length
              except:
                dict[str(source_net) +", " +dir +", " +proto +", " +geo_source] = length
            if count > 1000:
              count = 0
              current = time.strftime('%d'+'-'+'%m'+'-'+'%y')
              t = time.time()
              if current == start:
                log = gzip.open(os.path.join(PG_DIR, 'logs', 'pcap' +'_' +hostname +'_'+start +'.log.gz'), 'w')
                for i in dict.items():
                  log.write(str(i) +'\n')
              else:
                dict.clear()
                start = current
                log = gzip.open(os.path.join(PG_DIR, 'logs', 'pcap' +'_' +hostname +'_'+start +'.log.gz'), 'w')
              log.flush()
              log.close()
              total = time.time() - t
              print "processed ", len(dict), "networks in ", total, "seconds"
            else:
              pass
          else:
            print "dead packet"
      except:
        pass
```







computer on a wide network like the Internet easier for humans, with the added benefit that each address contains content of its own—information about location, for instance.

The Internet Assigned Numbers Authority (IANA) has been responsible for the allocation of blocks of IP numbers to countries, while guarding certain restricted ranges for corporate and military interests. Several other ranges (such as the 192.168.0-255.0-255 range) are reserved for Local Area Networks. All such proscriptions considered, of the 4,294,967,296 possible unique addresses, around an eighth of the numeric space of the Internet is out of bounds for public use.

The first two blocks of any computer's IP generally contains enough information to ascertain its geographic region (or that of the computer's ISP at least). I derived this information from a third-party database

```
000      IANA - Reserved
001      IANA - Reserved
002      IANA - Reserved
003      General Electric Company
004      Bolt Beranek and Newman Inc.
005      IANA - Reserved
006      Army Information Systems Center
007      IANA - Reserved
008      Bolt Beranek and Newman Inc.
009      IBM
010      IANA - Private Use                  See [RFC1918]
011      DoD Intel Information Systems
012      AT&T Bell Laboratories
013      Xerox Corporation
014      IANA - Public Data Network
015      Hewlett-Packard Company
016      Digital Equipment Corporation
017      Apple Computer Inc.
018      MIT
019      Ford Motor Company
020      Computer Sciences Corporation
021      DDN-RVN
022      Defense Information Systems Agency
023      IANA - Reserved
024      ARIN - Cable Block                  (Formerly IANA - Jul 95)
025      UK Ministry of Defense              (Updated - Jan 06)
026      Defense Information Systems Agency
027      IANA - Reserved
028      DSI-North
029      Defense Information Systems Agency
030      Defense Information Systems Agency
031      IANA - Reserved
032      Norsk Informasjonsteknology
033      DLA Systems Automation Center
034      Halliburton Company
035      MERIT Computer Network
036      IANA - Reserved                     (Formerly Stanford University - Apr 93)
037      IANA - Reserved
038      Performance Systems International
039      IANA - Reserved
040      Eli Lily and Company
041      AfriNIC                             (whois.afrinic.net)
042      IANA - Reserved
043      Japan Inet
044      Amateur Radio Digital Communications
045      Interop Show Network
046      Bolt Beranek and Newman Inc.
047      Bell-Northern Research
048      Prudential Securities Inc.
049      Joint Technical Command             (Returned to IANA  Mar 98)
050      Joint Technical Command             (Returned to IANA  Mar 98)
051      Deparment of Social Security of UK
052      E.I. duPont de Nemours and Co., Inc.
053      Cap Debis CCS
054      Merck and Co., Inc.
055      Boeing Computer Services
056      U.S. Postal Service
[...]
```

stored locally as part of the *Packet Garden* software installation on the user's machine. I used these first two blocks of each IP as values for my longitude and latitude. Once I had plotted these points onto the 360x360 plane of my graph, I could then map each of them onto the base sphere of my garden. All other values (such as the amount of data collected at that address over the given time) were scaled to a range between 0 and 1. An icosasphere (my feature-less terrain) could be then deformed based on these values. Raw traffic proved to be enough to meaningfully produce a simple terrain, but there was still plenty of other data I could incorporate.

As the computer's activity log was parsed, the ports used by each packet of data were collated. If a match was found between the port used and a known application, a plant form representing that activity was created. There were plants of different types of activities: mail, gaming, HTTP, FTP. I wrote a file pairing ports with applications, but this was later added to by many users interested in detecting traffic from their favourite application. The positions of the plants fit the newly deformed surface so that each rise or dip would be marked with the kind of transaction responsible for it. Features without a detected protocol—sometimes it's impossible to tell a protocol from a port number—were left unplanted. IANA reserved ranges were represented as green crystals or seed-like forms floating above the terrain and marked as 'Private'. The terrain under these seeds would probably never see traffic unless the specific user had access to such addresses via a government, military or corporate intranet.

Packet Garden allows for users to generate a world whenever they desire, with one garden saved each day. Like a kind of a network diary, the user can compare past worlds with their current traffic. Of particular interest however was that some people were changing their browsing habits to generate different sorts of terrains: users were approaching the project in ways I hadn't anticipated, using the graph to shape the data rather than vice versa.

During the latter stages of development, *Packet Garden* got its first major coverage, in *The Guardian Online*. This exposed *Packet Garden* to a wide public days after my first beta release. Almost immediately

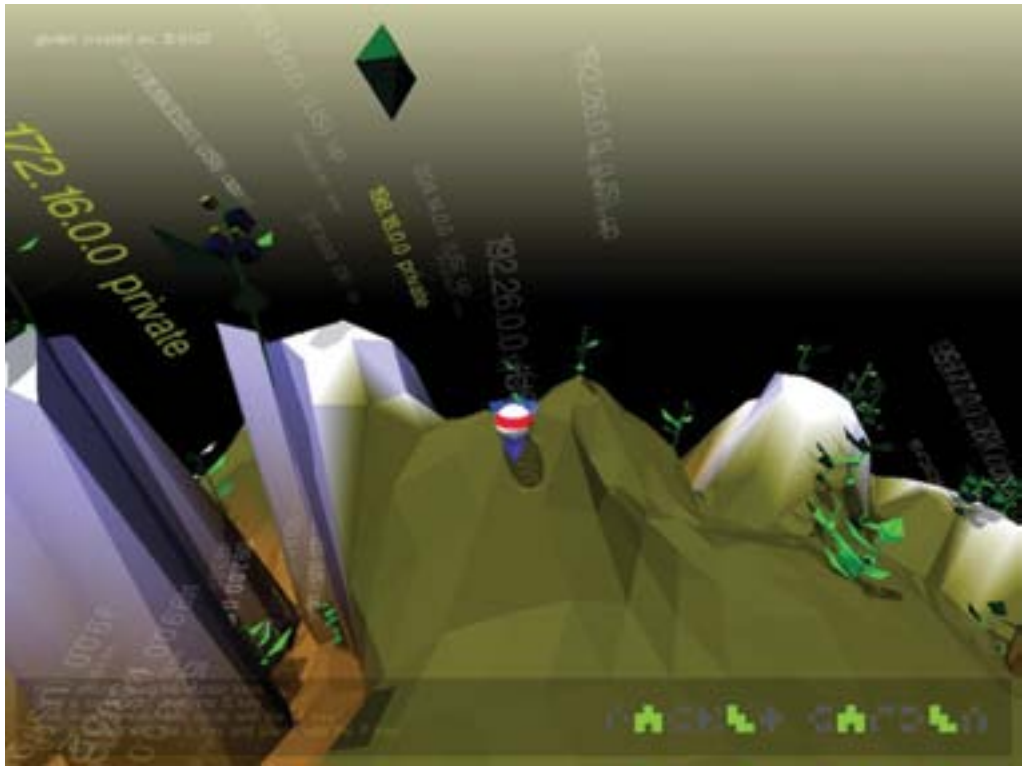

```
for i in self.peak:
        if i[6] == 'home' or (self.peak.index(i)/float(self.peak_len)) > self.filter:
          # progress bar output
          self.eta = 100 - (((self.proc_total - self.count)/float(self.proc_total))*100) \
             # range of progress bar in pg_garden.py
          self.meter.__set_value__(self.eta)
          self.count_label.label = "Processing " +str(self.count) +" of " +str(self.proc_total) \
             +" transactions greater than " +self.peak_limit +" megabytes"
          if i[6] != 'home':
            self.stats_label.label ="Country: "+str(i[6] +'\n')\
                                    +"Address: "      +str(i[0] +'\n')\
                                    +"Protocol: "      +str(i[5] +'\n')\
                                    +"Direction: "      +str(i[4] +'\n')\
                                    +"Transfer amount: " +str(i[3]/1048576.0) +" megabytes" \
                                      # this data feeds the progress bar
          else:
            pass
          pudding.main_loop.MainLoop(scene).update()
          self.count += 1
          angX, angZ = float(i[1]/256.0), float(i[2]/256.0)
          base_height =   self.peak.index(i)/float(self.peak_len) # value created from order in list.
          height = (base_height-self.filter)/(1-self.filter) # re-scale between 0 and 1
          if i[4] == 'up': # switch direction for uploads
            height = height * -1.0
          angX = (2*math.pi) * angX
          angZ = math.pi * angZ
          # spherical coordinate translations
          self.point.set_xyz(radius* (math.cos(angX) * math.sin(angZ)), radius* (math.sin(angX) * \
            math.sin(angZ)), radius* (math.cos(angZ)))
          vec = self.point.position().vector_to(self.world.position()) # vector of self.peak
          for face in self.world:
            for vert in face:
              v = vert.position()
              w = self.world.position()
              dw = v.distance_to(w)
              dv = v.vector_to(w).distance_to(vec)
              p = v.distance_to(self.point.position())
              if dv < .25 and p < radius:
                dist = 0.25 - dv
                vert.add_mul_vector(dist*height, vec) # smoothing
                vecs[dv] = vert
              else:
                vecs[dv] = vert
              v = vert.position()# have to call this again once the vert's moved before colouring it
              dw = v.distance_to(w)
              if dw > 1.6:
                vert.color = (0.6, 0.6, 1.0, 1.0) # mountain tops are blue-white
              elif dw < 1.3:
                vert.color = (0.3, 0.2, 0.0, 1.0) # dirt is brown
              else:
                pass
```

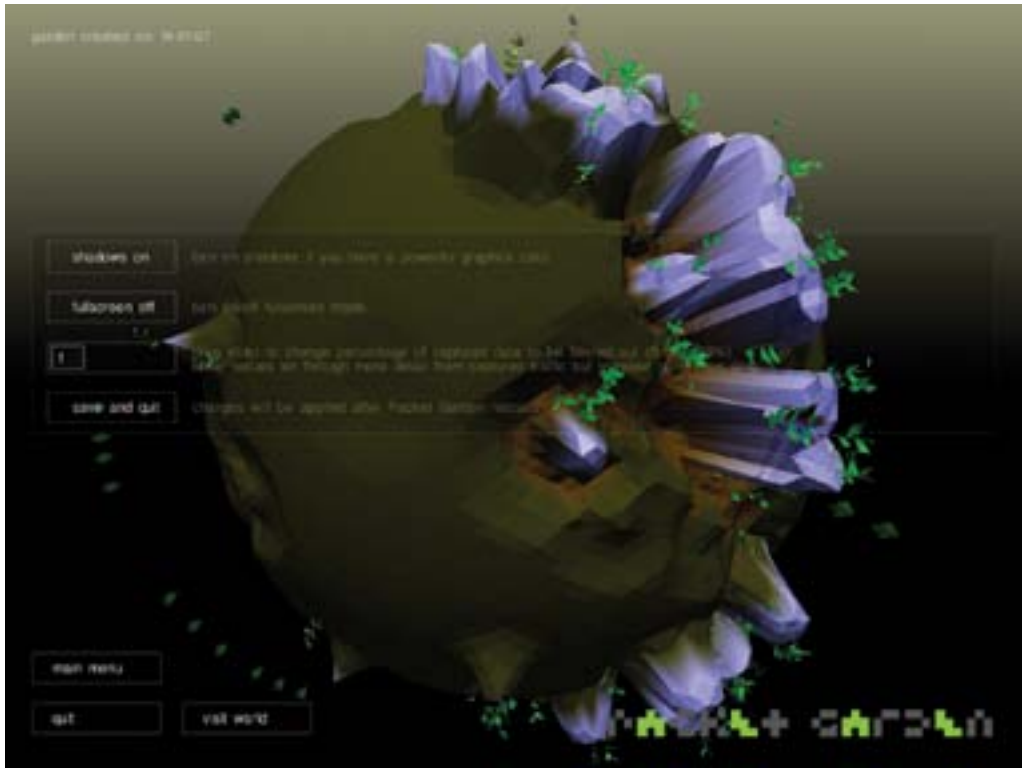
I had thousands of users not expressly interested in 'Information Visualisation' or 'Software Art' but fascinated by what their computer was doing 'behind their back'. Next came paranoia, something I entirely failed to anticipate. Online discussions emerged characterising *Packet Garden* as new breed of spyware from anti-file-sharing organisations, with one user even suggesting that it was created by the NSA. Others were simply concerned that *Packet Garden* was 'phoning home'—a data harvester disguised as a gimmick or toy.

In response to this discussion, two experienced security auditors took the code, screened it thoroughly and reported back. With their finding that my software wasn't strategically breaching their privacy, downloads picked up steadily. Here releasing the project under an open-source license where it could be subjected to such scrutiny proved to be vital to its continued distribution. In spite of this clean bill of health there was some truth to the criticisms: I was writing a kind of perverted spyware—software for the user to spy on their own computer.

As *Packet Garden* was approaching its 100,000th download I saw increasing discussions of the gardens as private worlds, worlds that needed tending and patrolling to protect the sensitive data they contained. More than a raw expression of information, the combination of landscape and personal online history was giving the gardens value as little territories: people wanted to name the worlds and landscape features and to invite friends in. This development of the gardens as places was something I had not anticipated, let alone attempted to produce.

What began for me as way of visualising imperceptible data would emerge as means for creating private places, worlds built from the everyday communications of users' individual machines. Each *Packet Garden* becomes a personalisation of the unseen life of the network.





Internet; Environment[1]

Julian Priest

Line

In 2000 James Stevens and I co-founded a project called Consume with a manifesto.[2] The first line was:

Define a sustainable *network* development.

The emphasis then was on the network and its self-provision; building it collaboratively and distributing its ownership in accordance with the decentralised design embedded in the Internet protocols to secure its long-term operation by users and in users' interests. This intention is still actively pursued around the world with much success, but now I want to re-state that line with a different emphasis:

Define a *sustainable* network development.

This essay is part of a continuing attempt to explore sustainability and the network, to examine the interdependent interfaces of the Internet and the Environment.

Wire

> *As I went under the new telegraph wire, I heard it vibrating like a harp high overhead. It was as the sound of a far-off glorious life, a supernal life, which came down to us, and vibrated the lattice-work of this life of ours.*[3]

> *Cyberspace. A consensual hallucination experienced daily… Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights, receding.*[4]

The Internet has often been characterised as a virtual space—the Cyberspace coined by William Gibson and applied to the Internet by John Perry Barlow. This virtual space of the Internet is a disembodied place where digital information can be experienced and exchanged. It is a space where physical journeys taking hours, days or weeks are replaced with 100 millisecond ping-time round trips.

From the 1990s a generation of computer users dived into this space. They wanted to float free of physical reality, to become telecommuters, netziens. They wanted to live inside the network, within the disembodied collective space of cyberspace, to leave behind the corrupt ways of the physical.

> *We will create a civilization of the Mind in Cyberspace. May it be more humane and fair than the world your governments have made before.*[5]

The desire to escape the materiality of the body can be seen in John Perry Barlow's *Declaration of the Independence of Cyberspace*. The old physical control structures can be avoided by de-materialisation; in his libertarian vision of transcendence and immateriality governments are not invited, and should be left in Gibson's 'meat space'.

Thoreau walking in New England in 1851 notices the telegraph wires overhead and hears music coming from the wind playing in them. He is not moved by the chatter of Morse square waves, sending instantaneous information to the

2. Julian Priest and James Stevens, *Consume the Net*, 25 July 2000. http://informal.org.uk/people/julian/resources/consume/text/
3. Henry Thoreau *The Heart of Thoreau's Journals*, ed. Odell Shepard (New York: Dover Publications, 1961), 57.
4. William Gibson, *Neuromancer: 20th Anniversary Edition* (New York: Ace Books, 2004), 69.
5. John Perry Barlow, *A Declaration of the Independence of Cyberspace*, 8 February 1996. http:/ /homes.eff.org/~barlow/Declaration-Final.html.