Cell-Forum:
Collaborative multi-user virtual world applications

Author: S.M. Wolff
Project: Cell-Forum Project
Student ID: 0333816

Supervisor: F.J. Verbeek
Group: Imaging & BioInformatics

Table of Contents

Abstract

Cell-Forum is a project for the creation of a 3D environment in which users can easily share and discuss 3D models and images. In particular, a 3D environment representing the Cyttron database (Cf. §1.3), developed at Leiden University, was considered. Collaborative virtual environments are used for collaboration and interaction of possibly many participants that may be spread over large distances. OpenCroquet, or just Croquet, is an open-source project that implements such a virtual environment. Croquet makes it possible for programmers to create and deploy such collaborative multi-user environments using the Croquet SDK (Software Development Kit), without having to know a lot about how the synchronization and consistency of data between the (large amount of) users is accomplished. This report provides a closer look into Croquet to see if this could be a good environment for projects such as Cell-Forum.

Chapter 1 – Introduction

In this section Croquet is introduced in the subsection “What is Croquet?”. The Cyttron Database is introduced in the subsection “Cyttron Database”, followed by a detailed description of the Cell-Forum Project and the connection with Croquet in the subsection “Cell-Forum Project”.

1.1 What is Croquet?

Croquet is a computer software architecture which was built with a focus on 3-dimensional collaborative environments for a large amount of users. It compares to the current incarnation of the World Wide Web in several ways, where users have the possibility to create and manage websites of their own to share text and images with other users and create hyperlinks to other websites. In Croquet users can create a 3D environment comparable to a website, so not a ‘homepage’, but a ‘home world’. Also it’s possible to create some sort of hyperlinks to other places of your ‘home world’ or even to other existing worlds. However, in addition, those worlds are fully dynamic and everything in it is an object that can be part of collaboration. The worlds are fully modifiable at all times, in collaboration with other users inside the same world. One way to think of the Croquet environment is as a high bandwidth conference call. Once a connection is made, users don’t only have text/voice communication, but can also easily exchange documents, images and other multimedia files, collaboratively perform complex simulations, design project plans etc.

Croquet is open-source and written in the Squeak programming environment (Cf. §2.1), which makes it highly portable between the different Operating Systems. Croquet is a development and delivery platform in one, making no distinction between the user environment and the development environment.

The focus of Croquet is on interactions inside a 3D shared space which is used for context based collaboration. Each user can see all of the other participants and also see what they are doing and what their current focus is. A new collaboration protocol called TeaTime has been developed to enable this functionality. The rendering engine is built on top of OpenGL.

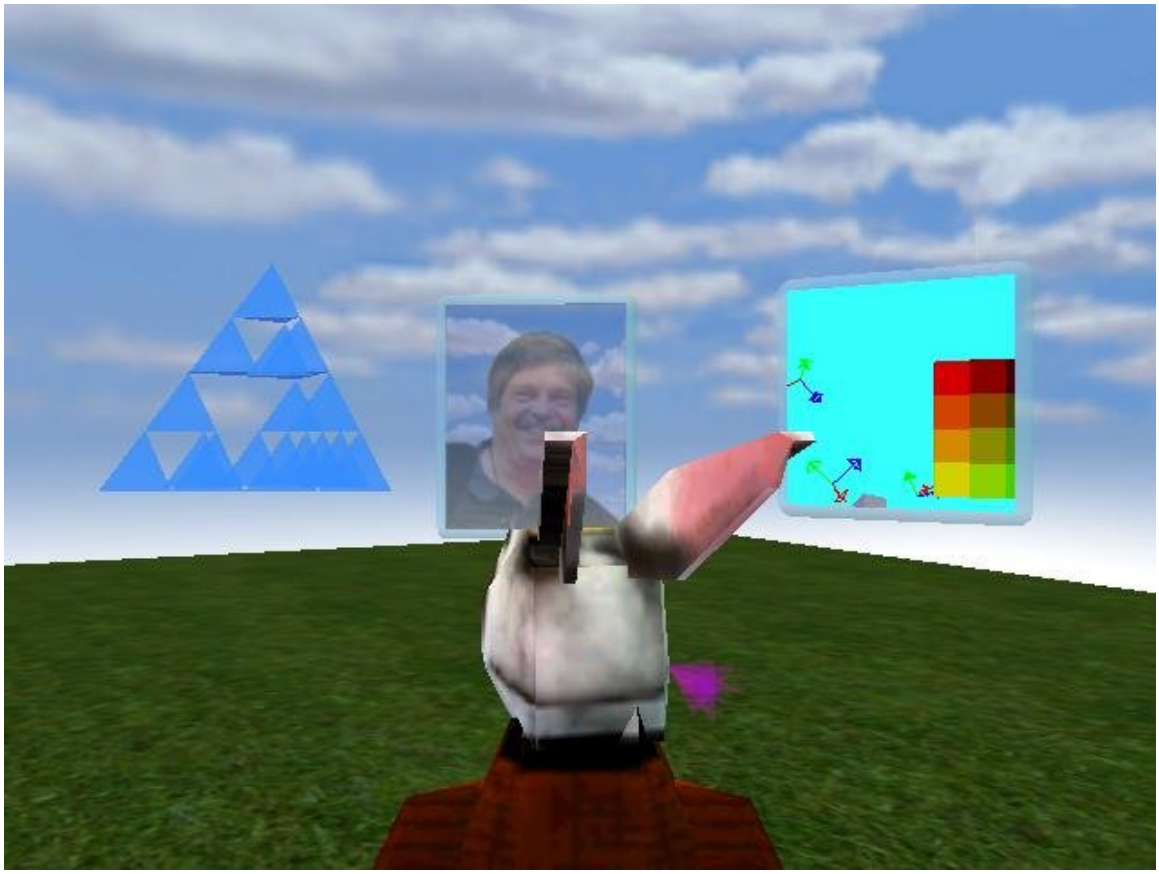

Figure 1. A Croquet Space containing several objects.

In Fig.1 a typical Croquet space is shown. These spaces may contain all kinds of objects, including ‘links’ to other spaces. Such thing is realized by creating a ‘Portal’ which connects to spaces to each other. These portals can be compared to URL’s in a website. They redirect users to other parts of the website or even to other websites. These portals behave exactly the same. In Croquet however, there is a big difference in how linking to another part of the ‘website’ and linking to another ‘website’ is being handled, more about this later on in this report.

1.2 Cyttron Database

The Cyttron Database is an image database of microscopic images built to support scientific research in the field of molecular cell biology. The images in the database, which are acquired through a range of different modalities, can be retrieved at different detail levels ranging from the detail level of cells to the detail level of viruses and atoms. The goal of the Cyttron consortium is to deliver a virtualized microscope covering the entire resolution range, where users can actually zoom in on the image through the different resolutions. To certain extend comparable with the zooming function in “Google Earth”.

Beside of offering the images in more than one resolution, Cyttron saves metadata along with the image, making it possible to retrieve related images to a given query.

1.3 Cell-Forum Project

The Cell-Forum project deals with the following problem; how can users collaboratively discuss and annotate images within the Cyttron database? This report should answer the question if the Croquet environment provides a suitable solution to the Cell-Forum Project.

The Cell-Forum Project deals with creating a 3D environment for the Cyttron database, a scientific image database created at LIACS. The project however will not only aim for 2D image displaying, but will also include 3D model viewing, annotation of items and collaborative modification and discussing of all in-world data. The result aimed for is a Croquet World which communicates with external resources for the images and objects to be viewed inside the world. Simple forms will be used to insert queries and the images and models will be located in dynamically created spaces within the World. These spaces will also be interconnected to related search results which may be interesting for the user(s). This way, search in the database by a user should result in several newly created spaces, connected in a way that represents the relation of the contents of these spaces to each other.

This way we hope to create a collaborative experience within a 3D virtual world to really give users the ability to ‘walk through’ the database and discuss and change/annotate the items they visit.

Chapter 2 - Material & Methods

In this section the programming language and environment of the Croquet project is described in the subsection “Squeak/Smalltalk”. In the subsection “Croquet Architectural overview”, the architecture and inner workings of the several components of Croquet are described.

2.1 Squeak / Smalltalk

Smalltalk was the product of research by a group of researchers at Xerox Palo Alto Research Center (PARC). The first version was Smalltalk-71 and the following versions were only used for research purposes inside of PARC. Until the release of Smalltalk-80 which was the first language variant that was made available outside of PARC. First as Smalltalk-80 Version 1, given to a small number of companies and universities and later (1983, Smalltalk-80 Version 2) a general availability implementation was released.

The Squeak programming language is a Smalltalk implementation, derived directly form Smalltalk-80 Version 1 by a group at Apple Computer. The group which implemented Squeak included some of the original Smalltalk-80 developers. It is object-oriented, class-based and reflective. Programs produced with Squeak run bit-identical on all other platforms. The Squeak system includes code for generating a new version of the virtual machine (VM) on which it runs. It also includes a VM simulator written in Squeak as well. For this reason, it is easy to port to other systems.

Development Environment Interface

The Croquet/Squeak environment, comes with some graphical tools which will be explained later on in this report, in the ‘How to program in Croquet’ section. (Cf. §2.3)

2.2 Croquet Architectural Overview

2.2.1 Islands

The Croquet architecture has been build upon the concept of replicated computation, rather than replicated data. Thus Croquet sends requests for computation to the data, rather than the other way around. Though it is necessary to transfer the current state of a world to a newly joined user, the rest of the synchronization is done by a message passing model where the messages themselves take care of the synchronization and consistency between the set of participating machines. More about the messages further in this chapter, first we'll take a look at the main objects of replication, the data where the computation is actually being sent to.

The main objects of replication are the so-called Croquet Islands. A Croquet Island is a secure container of other objects. Croquet Islands are simple save to disk and exchange with other users to initiate a collaborative session with its contents. Every user participating in a collaborative session has a replica of the Island on their machine, so an Island can be thought of as a collection of item containers all in identical states but operating on different machines all connected by a network.

Objects inside of an Island can send messages directly to each other or even to themselves. The spaces (Fig.1) we have seen earlier contain all kinds of objects which do or do not communicate with each other. These spaces however are actually also objects themselves, thus an island can contain several spaces, containing objects intercommunicating through the set of spaces.

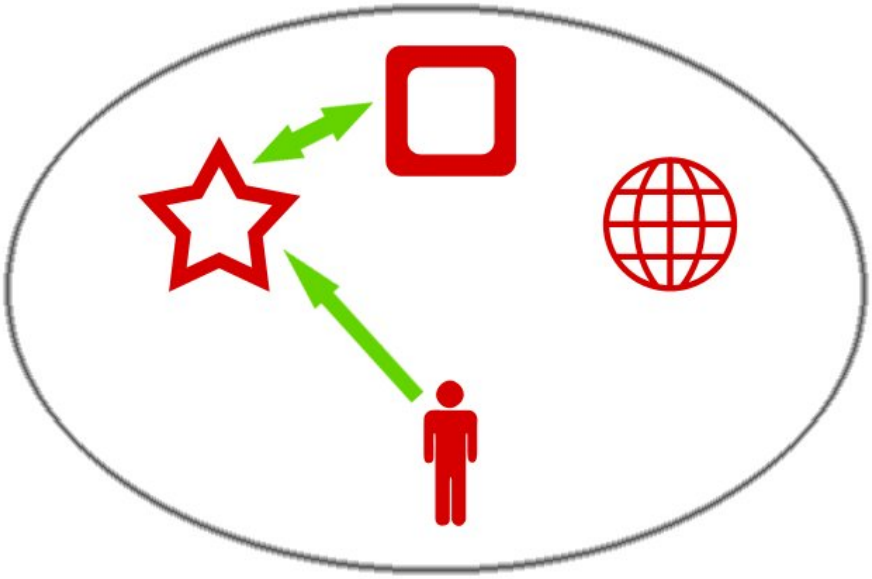

Figure 2. A Croquet Space containing several objects communicating.

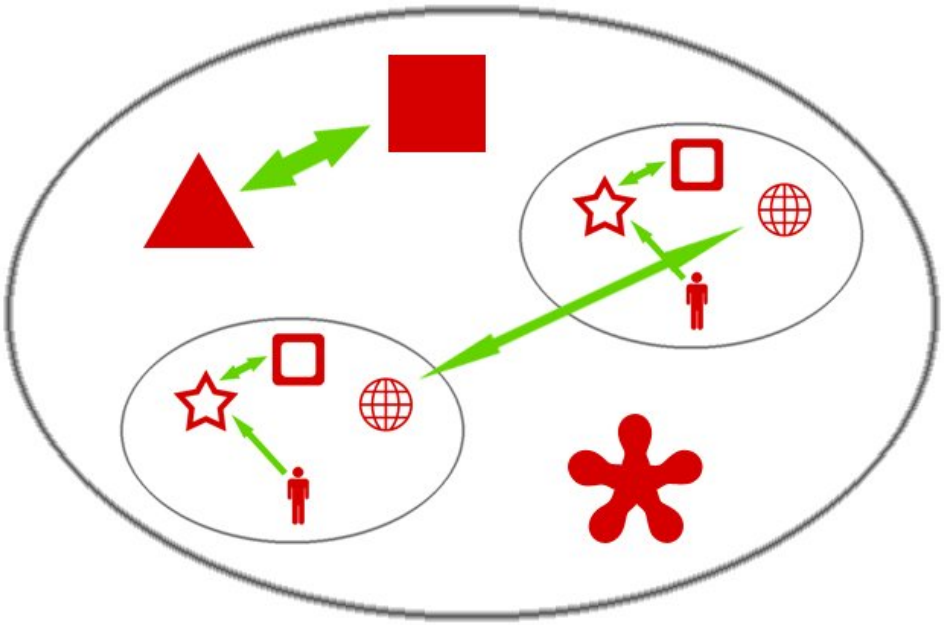

Figure 3. A Croquet Island containing several objects, including two spaces

However, it is not possible for objects inside of the Island, to send messages to objects outside of the Island's scope – nor can objects outside directly send messages to objects inside of the Island. Whenever a link to another Island is established, the Islands will be linked to each other, but the objects in the several spaces of the current Island will need a special 'TFarRef' object (Cf. §2.2.2) to communicate with objects in the additional Island.

As stated, Islands are the units of replication in the system. For Islands to function properly, they must be deterministically equivalent. Meaning they must retain an identical state given an initial state and exactly the same inputs, while not peeking or communication with each other. The idea behind this is that whenever objects inside of the scope of an Island communicate with each other, these messages won't have to be distributed between the several machines; they all have to be able to generate the exact same behavior. Whenever an external event occurs, for example a user changes the size of a window; the message will have to be passed throughout the participating machines. A further detailed explanation is covered in the sections (Cf. §2.2.2 - §2.2.3).

2.2.2 Messages

Objects within Croquet can only perform actions (rotate, increase in size, etc.) if they receive a message that contains the information about the action to be performed. Objects can send messages to themselves or to other objects within Croquet. The messages that are sent to objects outside of the scope of the Croquet Island are treated a little different than those that are sent to objects inside of the scope of the Croquet Island. From the point of view of the Object they are the same, but they are treated differently by Croquet. This will be described in more detail in section (Cf. §2.2.3).

Croquet messages consist of four components.

- **Target:** The object that will receive and execute the message.
- **Actual message:** The actual message, for example; "Move 'X' units in 'Y' direction".
- **Arguments:** The arguments to the message, as in the example above this would be the 'X' and 'Y'.
- **Execution time:** The time at which the message is to be executed, is always in the form of Now + Time offset. This time part of the message is also used to sort the Islands Message Queue.

Message Queue

There are two types of messages for the queue, external and internal generated messages. As the names already suggest, the internal messages are generated in the island itself and the external messages are generated outside of the Island-scope, most of the time through a user performing a particular action or task.

The Island does not distinguish between internally and externally generated messages, where external messages are usually generated by user inputs. The externally generated messages however, do play a major role in the synchronization part of the Croquet Architecture, which make them a little bit different from the viewpoint of Croquet. The external messages are used by the queue to indicate the upper bound to which the Island can compute its messages, without the danger of computing beyond any pending messages. This will be discussed in more detail in the next section (Cf. §2.2.3).

TFarRef

Sending messages to objects outside of the scope of the Croquet Island cannot be done directly, but are sent using a so-called ‘TFarRef’. A TFarRef is an object that exists outside of the scope of the Croquet Island, but acts as a proxy for the object that is inside of the Croquet Island’s scope. The TFarRef then forwards the message to the object intended.

The TFarRefs are generated by having the Croquet Island register a particular object as being externally accessible. An external name is then generated and a TFarRef for that object is created by the Croquet Island. The Croquet Island holds some dictionary that maps the TFarRef to the actual object.

2.2.3 Croquet Time and synchronization

The synchronization of the running copies of a Croquet world running on several machines is crucial for Croquet to function properly. It is of major importance, that whenever a user performs an action, for example increase the size of or moving an object, all the other participants can see this happen in their copy of the world as well. All of the participants must have their worlds fully synchronized at all time, or else collaboration between the users and discussing/annotating of on-screen objects would not be possible.

The definition of time plays a central role in the synchronization methods in the Croquet Architecture. Islands are deterministically equivalent (Cf. §2.2.1), meaning that the internal messages generated within the Islands are to be executed in exactly the same order and time, so the states will be exactly the same at all times. Externally generated messages have to be properly interleaved in the queue with the internal messages at exactly the right time and order.

The order of messages in the internal message queue is the only view of time an Island has. These messages literally define the Island clock. Although Islands have internally generated time based messages that can be queued up to a certain future point, these cannot be released for computation until an externally generated time based message has been received and inserted in the queue to indicate the outer temporal bound to which the Island can compute the messages to. This is a key point in the architecture to ensure the synchronization between the set of Islands participating in a collaborative session. So whenever a huge number of internally generated messages are eagerly waiting to be executed, they remain pending until an externally generated message comes in setting the messages free to be computed, up to and including the newly received message.

During the execution of a message from the queue, the time does not advance during the execution, it remains atomic. Whenever future messages are generated during the execution of a message, it's time of execution will always be defined in terms of "now" plus an offset value, where the offset-part has to be bigger than zero. Meaning that all the messages in the Island queue are "future" messages (to prevent Croquet from getting in a loop where to whole world appears to "freeze"). The generated messages are obviously sorted in the queue by the time they're meant to be executed.

Internally generated messages are implicitly replicated, meaning that whenever two copies of an island are running on different machines and are in fully synchronized state, the messages generated by objects that are inside of this island are automatically generated exactly the same on each running copy of the island. Internal messages involve messages generated and processed within each Island replica separately, so no network traffic is involved for this.

Externally generated messages are explicitly replicated, meaning that these messages will not be automatically generated on each running copy of the Croquet world and have to be send to all of the participating machines to ensure all the running copies of the world will all remain synchronized at all times. The name already indicates that these messages are generated outside of the scope of an Island, typically by one of the users participating in the Island. The replication of external messages is handled by an object which will be explained in the next section and is called a Router. The Router replicates the externally generated messages and determines when the message will be executed.

As time is atomic and the external messages are the actual Island clock, network latency has no impact on the synchronization. It does mean however, that users have a degraded feedback experience.

2.2.4 Controller

The role of the Croquet Controller is to act as the data-interface between the Island and the Router and between the user (Cf. §2.2.6). The main job of the controller is to ship messages around between all the parts of the system.

The Controller also manages the Islands message queue, by sorting the incoming messages based on their timestamp and by determining when messages will get executed.

Interestingly, a Croquet Controller can exist without an Island, acting as a proto-Island until the real island is either created or duplicated. In this case it is used to maintain the message queue until either a new Island is created or until an existing Island is replicated.

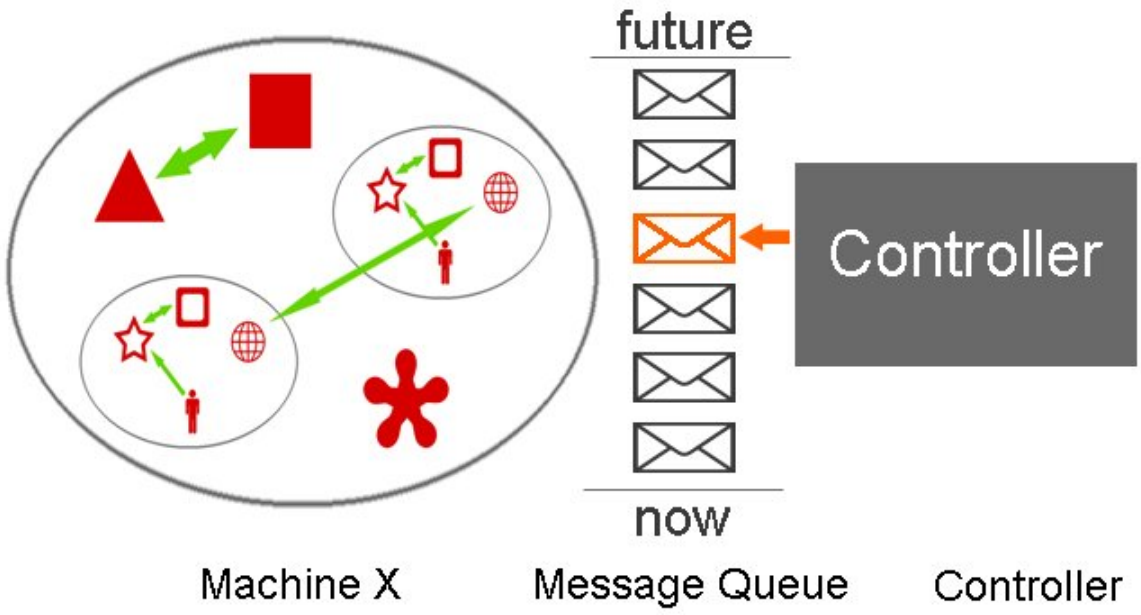

Figure 4. Croquet Controller inserting a new message

2.2.5 Router / Sequencer

The Croquet Router plays two major roles. First, it acts as the conductor for the replicated Islands in that it determines when an external event will be executed. These external events are the only information that an Island has about the actual progression of time, so the Island simply cannot execute any pending messages in its message queue until it receives one of these time-stamped external messages. The second critical role played by the Router is to forward any messages it receives from a particular Croquet Controller to all of the currently registered Islands.

Given that Islands cannot execute beyond these external messages, it is usually necessary to manufacture new messages simply for the sake of moving time forward. These messages are created by the Router and are called heartbeat messages. They are basically message-free and only contain a time-stamp that generates an end-point of the message queue allow the island to execute any pending messages, to prevent the time from standing still when no 'normal' external messages are being generated.

Routers can be located almost anywhere on the network and need not be collocated with a particular Island. Typically, the creator of the Island will own the Router by default.

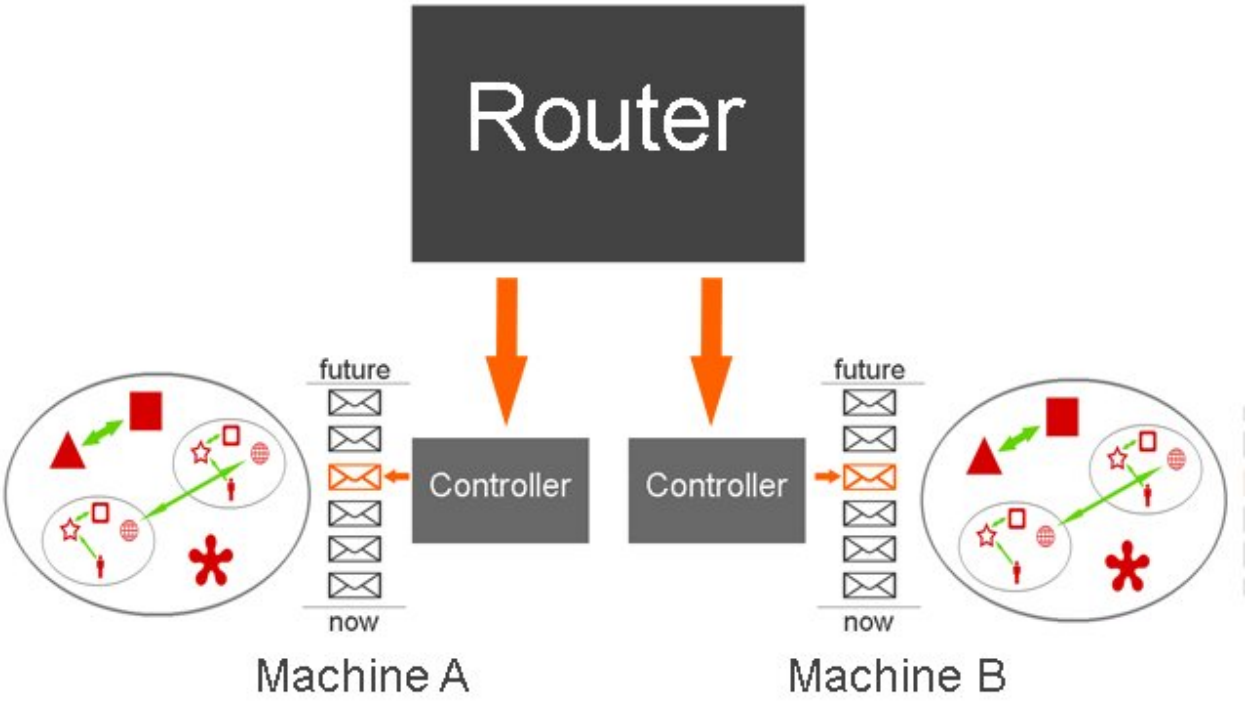

Figure 5. Croquet Router forwarding a message to all the Controllers

2.2.6 The user

Every user that is participating within a Croquet Island is represented by an avatar within the 3D virtual world. This makes it possible for other users to see where and what a collaboration partner within the Croquet Island is focusing on. A user is actually an external object to the Croquet Island, meaning that every action performed by the users (walking around, manipulating items, etc) has to be explicitly replicated on all the running copies of the Croquet Island. The HUD (Head-up Display) / UI, is a property of the object representing the user and therefore can easily be changed as well.

2.2.7 Overview TeaTime

Croquet is designed to operate as a peer-to-peer architecture, this to ensure the greatest level of flexibility in the design of the system and its ultimate usability. The key part of the architecture making up Croquet to enable this peer-to-peer interaction is TeaTime, which is the basis for component object-object communication and world/object synchronization between the set of Croquet worlds, including the initial content synchronization when a user is joining an existing world.

It is important to notice that everything inside Croquet is an object. Even the user interface is more a property of the object that represents a user in the space and consequently it can easily be modified.

Some items that will be used a lot while working with Croquet are TObject and TFrame:

TObject
A Tea object acts with the property that messages sent to it are redirected to replicate copies of itself on other users participating machines in the peer-to-peer network. All of the interesting objects inside of Croquet are constructed out of subclasses of TObject.

TFrame
The component base class, also a subclass of the TObject class, is called TFrame. The subclasses of TFrame act as frames in an OpenGL rendering hierarchy, as event handles, and as time based simulation objects as described above as part of TeaTime.

2.3 Tutorial

The first few steps of programming in croquet were very hard for me, as there was some serious lacking of decent documentation and tutorials. With some trial-and-error and some practice however, the basic idea of how to create some simpler Croquet Worlds and applications within those worlds became clear. Once you get to know the Squeak environment and the Croquet architecture, creating objects and giving them certain functionalities was actually quite straightforward.

Hopefully starting Croquet application developers will find the (novice) tutorial located in the appendix of this report useful.

Chapter 3 – Implementation / Results

In this section the construction toes for the Cell-Forum Project are stated and described.

3.1 Cell Forum construction toes

Due to time constraints, the development/implementation of a complete 'skeleton' for the Cell-Forum environment was not possible. Therefore, only some of the required construction toes were selected and their functionality implemented within a Croquet World.

Without doubt not at all a complete code, but it provides a good start. The following functionality is implemented in a test World:

- Dynamic creation and linking of spaces and their contents
- Remote server communication
- Virtual Network Computing (VNC)

3.1.1 2D Interface inside of 3D World

While in the 3D World, you will still need some text-based interfaces which will eventually be in 2D. Squeak provides some tools to create such 2D interfaces with the ability to load them in a window inside the 3D World. How these 2D interfaces can be created will not be described inside this report, but links to online tutorials will be provided for further details on this subject.

3.1.2 Dynamical creation and linking of spaces

The idea was to create a new 'room' for every item that has been searched and requested to load. The new 'room' will then be added to the Island, linked to with a portal and the requested image / model will be loaded from a remote server and placed inside this newly created 3D space.

For the input of important parameters for the creation of these rooms, 2D windows with text fields and buttons were created. Just for testing reasons you can decide on the color of the room and you can decide on the item that is to be loaded inside of it. The pressing of the execute button results in the appearance of a portal to the new space.

Pointers, or the addresses to the data structures, to the dynamically created spaces are stored in an Ordered Collection (similar to an array, but then named differently), so the content of the spaces can still be altered after some time has passed. For testing reasons

some starting code is written to link several created spaces together, to create some "graph-like" structures.

The most crucial parts of the code for this are included inside the appendixes section.

3.1.3 Remote server communication

To load the requested data into the Croquet World, external server communication will be required. In this case I have only tested the simplest case of visiting a web server running a PHP script, giving this PHP script some input and returning the outputs of this script to variables within the Croquet application for further usage. The reason I chose to test it with a PHP script was that I am familiar with it, but it could have been anything else as well.

3.1.4 Virtual Network Computing

A very useful feature that enables users to set up a connection to a remote PC and actually see its desktop and be able to use it as if you would actually sit behind the PC itself. This isn't something new, but creating such a connection to a desktop inside of the collaborative croquet world does open some new possibilities. As now some programs that were not originally created for collaborative use, are now all of a sudden viewable and accessible for a group of users simultaneously inside of the Croquet World.

A link to a step by step tutorial for setting up such a VNC server and opening it inside of the croquet world is included inside of the appendix section.

Chapter 4 – Discussion / Conclusions

Similar/Related projects
Croquet is not the only environment that offers developers to create collaborative 3D virtual worlds. The Wonderland Project (ref. 10), formerly know as the Looking Glass project, is another one of the several projects that aim for the same kind of functionality as Croquet does. At first glance the Wonderland Project looks a little further in the development phase than Croquet, but for now I cannot give my opinion on which one is the better. Another project that is somewhat related is the Win3D project (ref. 12). Although Win3D does not really aim for the collaborative aspect, it does focus on 3D GUIs in general, which can be very important to Croquet, or more to Croquet application developers as well.

Own experience
The status of development which Croquet retains in while I am completing this report, is that it is just not finished yet. Although the software package has already been updated several times and has improved significantly in terms of stability and functionality within the past several months, it is currently still lacking some promised functionality and/or some functionality is not really functioning properly yet. It is expected however that Croquet will continue to improve rapidly for the time coming, as the Croquet community is growing and very active at the moment.

Within the time I spent working with Croquet, I learned a lot about how Croquet is designed and how to effectively create applications with it. Unfortunately it took me little too much time to really feel accustomed with it and therefore have not been able to actually create a working application for the Cell-Forum project. I have been able however, to look into and develop some of the construction toes required for the Cell-Forum application.

For me it was the first time that I jumped in the middle of such a large open-source program to try and understand it, so my approach would be a little different next time. A better approach would be to first read some papers about the architecture of a project and why certain implementation choices were made. Especially when the programming environment is really lacking good tutorials, knowing what and why you are doing is incredibly much faster than the classical ‘trial and error’ approach.

Future Development
A ‘road map’ for future development goals for the Croquet project is defined at (ref. 1) and contains some key improvements that will undoubtedly take Croquet to a higher level. Some points that I would like to highlight are; (1) the text/menu’s inside of the 3D world. They really need some improvements to present an attractive GUI to the end-user and to reflect the quality of its underlying technologies. (2) Movement of your avatar inside the 3D world. At the moment this is done by positioning the mouse on certain positions on the screen and clicking it to move in a certain direction. Although it works

fine, a more general method which is also widely used in all modern 3D gaming software would definitely fit more and improve the usability of Croquet.

Conclusions

Croquet gives designers a fairly straightforward development environment to create and publish powerful collaborative applications. The developers do not need to know a lot about how the collaboration works. Furthermore the system is designed in such a way that it is easy to take an existing object and modify it by making a subclass for it to fit your own needs. This way it is possible to create functional applications in a reasonable time period.

Whether Croquet will eventually be widely used or not is hard to say at this point. The project is clearly still in the development phase and therefore it is difficult to foresee. The future plans however are really promising and as the community is very active at the moment, we might just see more of this project in the near future.

References

1. http://www.OpenCroquet.org
2. http://www.duke.edu/~julian/Cobalt/Home.html
3. http://www.dmu.com/croquet/
4. http://xaverse.blogspot.com/
5. Smith D.A, Kay A, Raab A, Reed D.P (jaartal)
   Croquet – A collaboration System Architecture
6. Smith D.A, Kay A, Raab A, Reed D.P (jaartal)
   Croquet Programming, SDK guide
7. http://www.smalltalk.org
8. http://www.squeak.org
9. http://www.scribd.com/doc/2913988/-technische-uiteenzetting-Edusim-3D
   Technische uiteenzetting Edusim-3D
10. https://lg3d-wonderland.dev.java.net/
11. Kallergi, A., Bei, Y. Kok, P., Abrahams, J.P. ,Dijkstra, J., Verbeek, F. (2009)
    Cyttron – A virtualized microscope supporting image integration and knowledge discovery.