Interim Report

ConceptTop - Concept Map Based Desktop

Alistair Steele - 0930435

Supervisor – Dr Frank Langbein

Moderator – Dr Xianfang Sun

Contents

Abstract

This document details the research carried out (including history of document managers, 3D rendering and self-organising graphs) and methods and tools to be used for the creation of a 3D concept-map based document manager. The tools needed to create the project were discussed and selected (including C++, OpenGL, SQLite3 and the Qt framework) based on their appropriateness for the topic. Advanced features were also identified such as a self-organising option for the underlying graph structure, however this was deemed an optional extra if time permitted. The first stages of the project design including the core database tables (Nodes and Edges) and a top-level component diagram (detailing the Window, Renderer, Graph and Database components) were also included.

Introduction

This project aims to design and develop a document management system that utilizes concept-maps to describe the relationship between documents. The concept-map will be visualised in three dimensions (3D) and will be fully navigable by the user. In addition the user of the software will also be able to alter the concept-map to include the documents they wish to organise and create relationships between said documents.

Concept-maps are graph-like data structures where each document or element is known as a node and the relationships or connections between nodes are known as edges. Unlike a standard graph however, in a concept-map these edges can signify a relationship between the nodes on either end. These relationships can take many forms and they can usually be represented by a textual label. In addition to standard connections, this project may also seek to provide support for hyper-edges. These are a special form of edge which can connect more than two nodes and as such can act as a 'folder' or 'directory' style listing within the concept-map where all nodes on the hyper-edge share a certain property or meaning.

Aims

The aims of the project are varied, however the primary goal is to assess whether a 3D visualisation of a concept-map can be used as a method to organise documents on a personal computer and describe their relationships. Part of this core aim is to examine whether a 3D representation is preferably over a 2D or 1D variant. A focal point of this project's aims and objectives is to ensure that any control and representation system devised for use by average computer users will be as simple as possible and remove much cause for confusion.

This report will seek to explore the functional improvements to document management that can be achieved with a 3D concept-map based structure, whether or not the ability to organise documents in a fully 3D environment is practical and useful. I will also be exploring any advanced features that may make worthwhile improvements to the project, but may be not provided due to time constraints.

Background

Context

The main goal of this project is to create a user interface that provides document organising tools within a 3D environment. Document organisation has long been a core component of a user's experience with computers. Over the years of software development and evolution, this base aspect has remained largely the same with very little revolution changes occurring. While an important aspect of this project is to explore new alternatives to document management, an equally important part is to examine and evaluate the use of 3D representation. Computer hardware has been rapidly and steadily evolving and most machines are now capable of creating 3D visuals relatively easily. However widespread software development has not taken great advantage of this increase in graphical processing power and indeed, most applications still continue to use basic menus for user interaction.

Research

This section will detail some of the research that I have carried out in order to assess the current evolution of document managers and 3D applications. In addition, there was some research done into the use of self-organising graphs and how this could be used within the project.

Document Managers

Operating systems began with simple command line user interaction, with no visual listing of documents and files beyond textual output. In fact most console commands that sought to list files only did so for the current directory as it was seen on the underlying file system. This lack of visual representation was mainly due to limited computer hardware during the period.

Most modern day document and file management systems consist of a window which displays the files either in a list or tile format. While these thumbnails aid the user in acknowledging the file type and therefore the application it will open with, their context is still limited to the files in the current directory. A tree pane is often included as part of the file browsing window, this displays the currently viewed directory's position in the file system and helps the user to navigate among parent and sibling directories. This can be helpful when dealing with file duplication and transfer. The frequently used operating systems of today all use examples of this form of management, these include Windows 8 (Appendix Figure 1), OSX 10.8 (Mac) and the most widely known Linux distributions such as Ubuntu 12 and Linux Mint 14.

While list and tile form file managers are assuredly here to stay for the time being, there have been some attempts over the years to create a 3D visualisation for use in file management. One such example is BumpTop (Appendix Figure 2).[1] Originally a master's thesis for a student at the University of Toronto, this software sought to create a 3D space with which a user could collect all the important items on their machine and organise them as they saw fit. One of the highlighted features of BumpTop was that each item on the virtual desktop was affected by physics, the user could 'pile up' documents or files and then 'throw' them across the surface of the 'desk'. Although this feature is certainly impressive and led BumpTop to have quite a following, it did not actually change the interaction or method by which users organised their files.

In my research I attempted to find examples of concept-map or graph based document organisers, however I failed to find any popular examples. This may be due to a lack of interest in the research area or the idea may have been attempted before but stopped early on because of poor user reception.

3D Graphics

There are many examples of fully 3D desktop applications in the world today. The majority of these are games of some sort which render a player or players in a game world and then continually render the actions carried out by the user as they affect the game world around them. While these do bear some similarity to the software described in this project, this project will be significantly less complex. In addition, the control and navigation system that will be present in the final prototype of this project will have greater similarity to genres of 3D modelling programs such as Computer-aided design (CAD). For example, the user will be presented with a view and have the ability to rotate around the scene or concept-map as well as pan up and down and zoom in and out. Restricting movement of the camera to these actions will help to prevent user confusion and maintain a productive environment within the application. An alternative approach to visual navigation would be to allow the user to directly control the camera position, as if they were 'flying' through the 3D space. In my opinion this would lead to confusion as it would reduce the user's perspective and require them to find a suitable position every time they wished to view a different area of the concept-map.

Octrees

There are many things to consider when creating a 3D application, one of the most pressing is performance issues. Drawing a large collection of polygons and visual effects onto the screen takes a significant amount of processing power. Due to this a lot of research has been conducted into optimizing techniques to reduce the amount of the visual data that needs to be processed and rendered to the screen.

One such technique I have been looking into is by use of a spatial data structure known as an octree. An octree is a recursive structure where each node stores a three-point coordinate and is then subdivided into eight children. When a view is then rendered, the camera angle is projected from its origin across the octree structure in all three dimensions. When a child node is found to be within the viewing angle it is subdivided and then each child is checked for visibility. This process continues for a set depth level (as the process could continue indefinitely). Any node that is outside of the viewing pane is discarded and therefore never subdivided (Appendix Figure 3 is an image of a quadtree which is similar to an octree but deals with four children per node and not eight. The method is demonstrated in 2D and the image was chosen for clarity regarding the implementation.).

This means that entire sections of the visual data can be discarded for that draw as no part of it will be visible from the camera. For my project I would start by storing the positions of each node within the octree structure, which itself will need to be created by myself. There are many obstacles to implementing a functional octree optimization method and I feel this will be a significant challenge for my project. However I feel it will be worth it as performance would greatly increase when viewing very large concept-maps.

Self-Organising Graphs

An initial idea for this project was to allow the program to organise the documents stored within the concept-map into the most efficient order based on the edges or relationships connecting them. I explored methods of doing this as part of my background research.

The general algorithmic idea behind computationally organising a graph-like structure consisting of nodes and edges is to simulate physics within the structure. Each node is considered a physical body and each edge is treated as a spring or constraint between two nodes. The algorithm then carries out a number of iterations to the point where the graph changes little. Within each iteration the distance between each connected pair of nodes is reduced slightly, this is done rather than snapping them to a fixed distance apart so that the graph does not flip between two extremes. The gradual reduction in distance between connected nodes means that after the final iteration the graph will be in a relatively stable state and should have an efficient layout.

A large graph may have many nodes and edges described in it and as such an organising algorithm that executes a significant number of iterations may take a large amount of processing time. This may require the

UI to indicate the progress of such an algorithm. In addition, automatically re-arranging the documents displayed in the concept-map is likely to cause significant confusion to the user and as such this may be an option rather than an automatic feature of the software. I was unable to find any established examples for an organising algorithm and as such would be required to create my own, due to this and other issues mentioned above, this feature will be considered advanced and attempted if and when the basic functionality has been completed.

Methods and tools

Software Language - C++

There are many programming languages that are capable of creating desktop applications, and many of them are able to draw in 3D space or have core or third-party libraries that add this functionality. C++ is a highly established and well supported compiled language. It can be used on multiple platforms and operating systems including Windows and Linux. While C++ is not a language that I have a high amount of experience with, it is one I am capable of using to create software. The majority of my previous experience with the language focused on internal mechanics of software development such as implementing various data structures and searching or sorting algorithms. Nevertheless I feel that I can adequately expand my knowledge during the course of this project to the level that will enable me to build the graphical and user interface elements required by this project.

Database - SQLite3

The proposed software requires a component capable of storing the concept-map data structure, for this task I decided to use a Structured Query Language (SQL) database. An SQL database will provide a great degree of flexibility in both the stored representation of the data and the way in which it is retrieved from within the C++ program. While there are many different implementations of the SQL database design, a large proportion of them are designed to be used in web-based systems. Integration of a web-based database, while not impossible, is largely unnecessary and instead I will use a local variant called SQLite3.[2] The database created by this library is simply a flat-file which is structured in such a way that it resembles a database. There is a large advantage to this form as it is highly self-contained. There is a small drawback to this SQL variant in that the local file that stores the data is not encrypted or protected in any way and as such can be read by directly opening the file with a text editor. I believe that this will not be an issue for this project as I had not intended to implement password protected concept-maps, however it has the potential to be a feature in any future versions of the software.

Renderer - OpenGL/GLUT

Open Graphics Library (OpenGL) is a cross-platform, open source graphical library that provides a wide range of rendering capabilities. Originally developed by Silicon Graphics Inc. in 1992, with the goal of moving the responsibility of creating interface layers between the hardware and software to the hardware manufacturers.[3] This allowed application developers to greatly increase efficiency as they could work using a standard Application Programming Interface (API) rather than tailoring their code to specific hardware. While the algorithms and features defined in its standard can be implemented purely in software, it is intended to be highly supported at the hardware level. This aims to enable a great degree of speed and efficiency in its routines, which a crucial requirement of applications that generate 2D or 3D visuals.

The OpenGL Utility Toolkit is a small OpenGL library for C++ that adds methods for rendering basic geometric shapes to OpenGL.[4] In addition to this, the library also provides helper methods for capturing keyboard and mouse input from the user. As the Qt library (described below) already provides this functionality as part of its OpenGL Widget I will not be using these sections of GLUT's API. As I currently plan to only use GLUT's helpful primitive shape rendering methods within my program it may be the case that I can find other ways of drawing the representation for each node. In this case it may not be necessary to use GLUT within the application at all.

User Interface - Qt

Qt is a User Interface library that is heavily used for a variety of applications, initially developed by the mobile phone provider Nokia before being sold to Digia in March 2011.[5] The library is built in standard C++, however its capabilities are greatly expanded by the use of a Meta Object Compiler which converts Qt macros in the program source code into C++. The added features include the signal and slot system which allows for UI elements to be connected, this is similar to a dynamic callback design, as well as increasing the introspective abilities of the C++ language. When I initially looked into using Qt for my user interface I had concerns that it would be difficult to integrate or use alongside an OpenGL window. However I discovered that one of the main widgets of Qt is called QGLWidget which directly allows the developer to place an OpenGL implementation within the Qt application window.[6] This widget means I can easily separate the standard user interface elements such as buttons and menus from the OpenGL code within the program source. This will improve the clarity of my source code.

Research Question

Explore software currently available for the use of organising documents and to assess the viability and usability of a system that employs a 3D representation. In addition, implement a method by which to store a concept-map data structure while utilizing minimal storage space and processing power when accessing the data. The aim of this software is to explore concept-map and graph-like organisation for documents, and assess whether or not this is a useful means of document management.

Approach to Solution

Usability

The primary goal of this project is to assess the viability of using a 3D representation to organise documents on a desktop computer. As such, one of the core requirements of the software is its usability and ease of interaction. There are several aspects to be considered, the greatest of which is how to navigate across the concept-map in a 3D environment. Allowing the user full control over the navigation of a 3D environment has often led to confusion and inefficiency in completed projects. Due to this, I may instead implement a more controlled form of navigation, limiting the user to certain angles and perspectives.

Planned Features

Limited and Easy Navigation

I plan to create a simple and limited navigation system for the project. The controls will allow the user to rotate around the concept-map, zoom in to look at a small area of the node display and change the camera's elevation to look at nodes at different levels on the z axis. It is also a possibility that I will allow a user to select one or more nodes and focus the camera on those points of the concept-map, this should allow a user to limit their current view to their active work.

Sub-graphs or Folders
An important feature to be implemented is the ability for the user to group relevant documents into a 'folder' of sorts. On a graph structure level this will be achieved through the use of hyper-edges, these are edges that can connect any number of nodes and as such removes the two-node restriction placed on standard edges. The technical details associated with hyper-edges have not been fully explored as of yet, however it seems likely that hyper-edges will need to be treated as special cases within the program which will add to its complexity.

Mouse Interaction with Rendered Objects (Nodes/Edges)
A key aspect of this project is user interaction, part of this is the ability to directly select and operate on the objects that are rendered by the 3D library. When a mouse click is registered on the OpenGL widget (QGL-Widget) the renderer class will redraw the concept-map as it is seen but with colour coded objects. It will then be possible to identify the object clicked by decoding the red, green and blue (RGB) of the pixels under the mouse at the time of the click. Once this identification has occurred, appropriate options can be displayed based on what type of object was selected.

Saves/Loads without user input
A document manager forms an integral part of a user's computing experience. As such, I believe it is important that once a user has spent the necessary time selecting which documents will be organised by the software they should not have to perform any explicit action following the starting of the program. To this end the program should automatically load the stored concept-map upon start up and save the structure when closed. This should aid the program in providing a seamless experience and suitable environment in which to work.

Initial Component Layout and Design

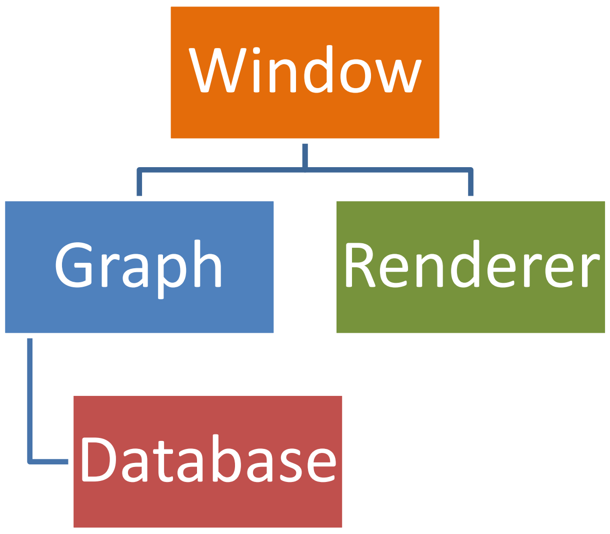


Figure 1. An initial top-level component view of the proposed software.

Database

The following is a brief description of the two main database tables that will store the data for the concept-map. These will likely be created through an external SQLite3 GUI (Graphical User Interface) but could also be created from within the C++ program using the database library and "CREATE TABLE" queries.

Nodes

Type	Name
Integer	id
String	label
String	path
Float	x
Float	y
Float	z

The nodes of the structure will be stored inside a table that consists of six columns. The first of these is an automatically incremented and unique integer that will act as an identifier for each node. The second column will contain the user-accepted label which will be act as part or all of the documents representation within the graphical application, this can simply be the document name as seen on the file system and then altered by the user at a later date. The path column will contain the location of the document as found on the file system, this may either be an absolute reference (including the volume containing the item) or a relative path. The latter option would be appropriate if the application was to be made portable. Finally, the three float-type columns will describe the coordinate location of the node within 3D visual representation.

Edges

Type	Name
Integer	id
String	label
Int	start
Int	end

The database table that will store the edge data will be similar in its structure. It will contain a column storing a unique integer for each edge, this will be used for identification purposes. There will also be a label value which will be displayed in proximity to the edge when it is rendered in the application. The last two columns within the edges table will store identifier values for the nodes at the start and end of the edge. These will correspond to records within the nodes table. It is important to note that the concept-map has many of the same properties of a directed graph and as such, the relationship described by each edge of the concept-map may not transition in both directions. A simple example of such a case is where a document is described as the "parent" of another. This is a unidirectional relationship and does not translate to both documents. It will be important to maintain the integrity of the database through these two columns. If a node is deleted and any edges that reference that node are not deleted or altered accordingly then the program will experience undefined behaviour as it attempts to represent the relationship between two nodes, one or more of which may no longer be present in the concept-map.

Graph

The graph component will deal with the logical operations of the concept-map data structure as well as maintaining pointers to each of the Node and Edge objects. This class will provide access methods to retrieve data as well as change the structure of the nodes and edges. The Graph component will be the only section of the program that has access to the Database and as such will be responsible for loading and saving the data when the program is opened and closed. The Graph component will also be responsible for providing data to the Renderer in order for it to be capable of drawing each of the concept-map objects.

Renderer

The OpenGL renderer will primarily be tasked with drawing each of the elements that make up the concept-map, including the nodes and edges. It will do this by retrieving data from the Graph component during the paint cycle and appropriately representing that data on the screen. The actual appearance of the nodes and edges is still undecided. One idea is to have each node displayed using the icon of the document it's representing. These are the same icons that would be used in a traditional file browser and as such should be very familiar to the user. The Renderer will also handle mouse clicks that occur within the OpenGL context. When a click occurs, the Renderer will redraw the current view with colour coded objects and then read the pixels beneath the mouse to determine the exact object that was clicked. This will then be passed to the Graph component to carry out the operation.

Window

The highest level component is the Window, this is where the general UI creation and interaction will be handled. There will be several options outside of the OpenGL context, including those to add documents to the concept-map either one at a time or by selecting a directory and adding all documents contained within. The latter process may or may not be preceded by an option to disregard one or more of the documents found by the search. In addition the user will be able to enter terms into a search box which will then find relevant objects within the concept-map. This component may also provide a small section detailing the specifics of the selected document as it may be impractical to display all of the information within the OpenGL context. It should also be noted that the Window component will be charged with the creation and ownership of the OpenGL context pane.

Flow of data through the system

The main information held by the program during the time in which it is running will be the concept-map itself. Although there will be large amounts of context-based data relating to the 3D visualisation and user interface, this will be mostly handled by the OpenGL and Qt libraries respectively. The concept-map itself will be represented by the Graph class which among other variables contains C++ classes that will describe the Nodes and Edges of the map. Before this data can be stored in memory as C++ objects it must be retrieved from the database, the Database class will be responsible for connecting to the SQLite3 file using its library and extracting the data. At this point it will then be passed to the Graph class via the use of a Result object which provides a means for iterating over the data safely.

Algorithms

The core of this project does not centre on a specific kind of algorithm or any great amount of processing. The focus is instead spread across several different areas, including storage, 3D rendering and user interaction. However as graphical rendering is a highly complex field requiring a considerable amount of mathematical calculation and data processing it is likely that specialist algorithms will need to be employed in certain areas of the program. One such area is the advanced feature of self-organising the graph structure to an efficient format. I currently have no examples of such an algorithm, however the basic concept is to simulate a physics environment with each edge acting as a spring between the nodes. This feature was described earlier and, if implemented would require a specific algorithm to function correctly.

Evaluation Process

As my project deals with the creation of a user interface and extensively with complex interactions, it will be necessary to evaluate the effectiveness of my design choices throughout the creation of the program. As this project is highly time-limited it will not be possible to gather a group of testers frequently throughout its development and this type of testing will be reserved for closer to the end of the project. However I personally will test the user interface design and ease of use thoroughly over the course of its creation to ensure that each option and planned user interaction is intuitive and fluid. The simplest way for me to carry out this testing is to use the program and make changes to its design and source code based on my experiences.

As mentioned above, toward the end of the project I will convene a small group of potential users to assess the effectiveness of the user interface and to see if there are any changes that can be made to greatly improve its usability. The results of this testing will form part of my final report.

Conclusions

This report has allowed me to collate the research I have carried out and assess each section for use within my project. In addition I have identified the tools I will use to create my piece of software (C++, OpenGL/GLUT, Qt and SQLite3). I have shown an initial component layout and explained how I feel each section will work to create the needed prototype. This document has helped me to identify the main features I will attempt to create and include in my design of a 3D concept-map based desktop. I have also been able to explore several features that, while impressive and potential useful in a program such as this, may need to be considered advanced features and only attempted if and when the standard functionality is achieved.

Future Work

The next stage of this project is to continue building the application, creating the standard features and exploring the navigation and UI design. I believe these design choices will be decided through careful experimentation and self-evaluation. If time permits I will attempt the advanced functionality described in this document in order to further explore the usefulness of a 3D concept-map based document manager.

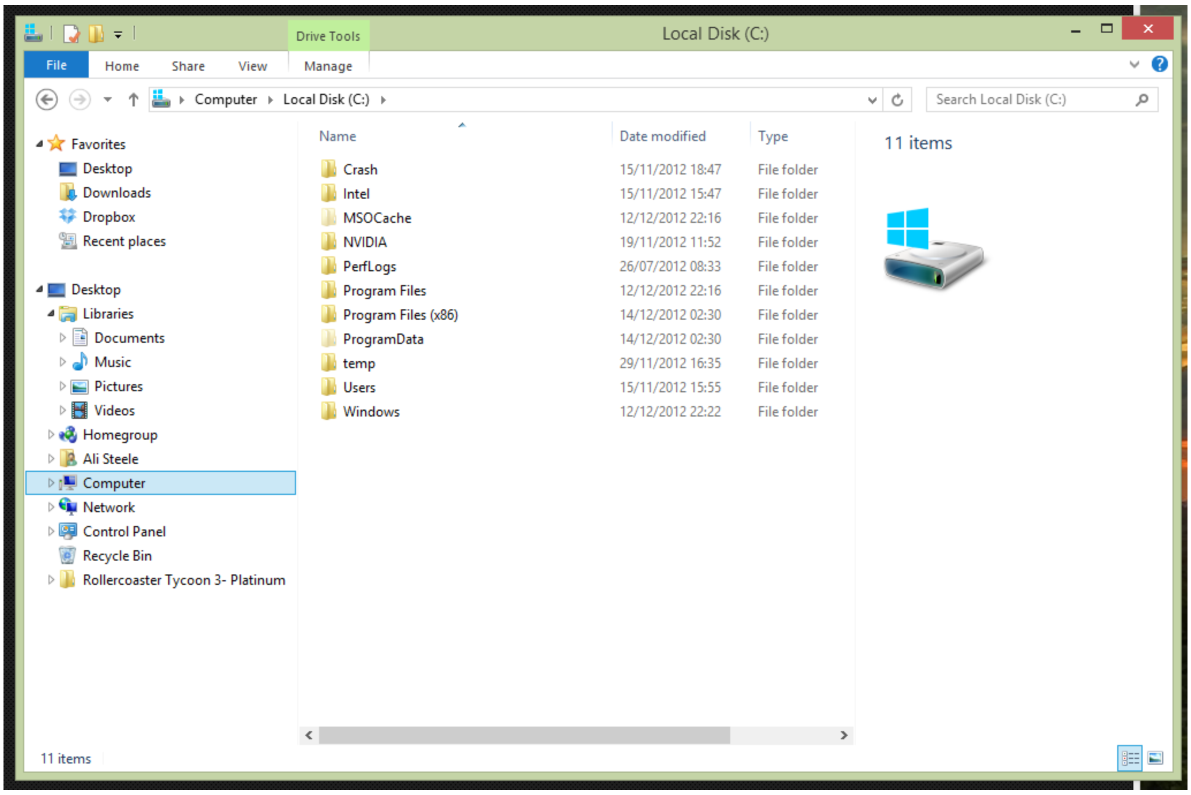
Once the prototype has been completed and all standard functionality has been created, I will carry out limited user testing with a small group in order to assess the viability of this kind of application, the results from this testing will be collected and analysed within my final report.

The final report will also contain a detailed description of the software development process, including the issues encountered and the solutions found

Appendices

Figure 1: Windows 8 File Explorer

Source: Self Taken



Name	Date modified	Type
Crash	15/11/2012 18:47	File folder
Intel	15/11/2012 15:47	File folder
MSOCache	12/12/2012 22:16	File folder
NVIDIA	19/11/2012 11:52	File folder
PerfLogs	26/07/2012 08:33	File folder
Program Files	12/12/2012 22:16	File folder
Program Files (x86)	14/12/2012 02:30	File folder
ProgramData	14/12/2012 02:30	File folder
temp	29/11/2012 16:35	File folder
Users	15/11/2012 15:55	File folder
Windows	12/12/2012 22:22	File folder

Figure 2: BumpTop

Source: http://upload.wikimedia.org/wikipedia/en/4/4b/Bumptop-desk2_600.jpg

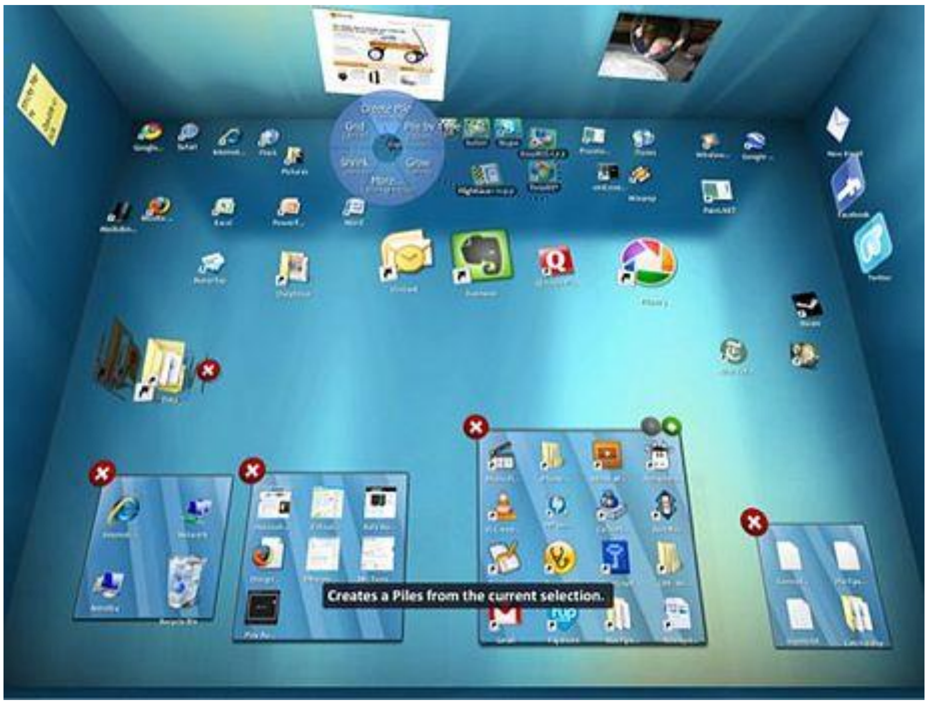
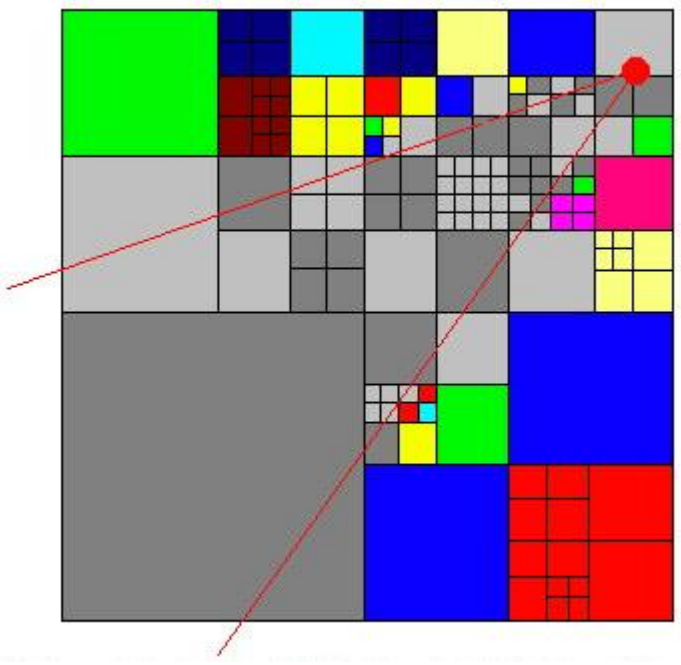


Figure 3: Partial 2D Quadtree subdivision, coloured nodes are discarded as they are not visible

Source: http://www.flipcode.com/archives/article_introtooctrees04.jpg



References

[1] – "Google buys Canadian 3-D desktop startup" – Retrieved 14/12/12

http://www.cbc.ca/news/technology/story/2010/05/03/tech-google-bumptop-computer.html

[2] – "SQLite Home Page" – Retrieved 14/12/12

http://www.sqlite.org/index.html

[3] – "OpenGL Overview" – Retrieved 14/12/12

http://www.opengl.org/about/

[4] – "GLUT – The OpenGL Utility Toolkit" – Retrieved 14/12/12

http://www.opengl.org/resources/libraries/glut/

[5] – "About us – Digia Plc" – Retrieved 14/12/12

http://qt.digia.com/About-us/

[6] – "Qt 4.7: QGLWidget Class Reference" – Retrieved 14/12/12

http://doc.qt.digia.com/qt/qglwidget.html