OpenCity

The official game design draft

Summary

I.Existing games

Simcity 1 – 4

Everyone knows about the so famous SimCity from Maxis. The first time when I played SimCity 2000, it was in 1994. SimCity 2000 is a great game, and I know that you love it too. However, when you play it for a long time, you get bored because of the monotony of the gameplay. Indeed, it consists of zoning for residential, commercial, industrial and governmental buldings then watch them growing.

However, I don't play Simcity 4, but only read some technical articles about it. There're the pros and the cons about the game. And since Will Wright is not among the designers of Simcity 4, a lot of fans found there a reason to criticize the game.

The sims 1

It's another famous game from Maxis. “The sims” is also a simulation game. However, it's not a SimCity like. In fact, instead of building a city, you build something much smaller: your house. En each house, you control its habitants and their interactions with the environment and each others.

Lincity

Lincity is a popular SimCity like for Linux. However, it's in 2D and lacks of eye-candy graphics. Recently, this game was remade by a great team of fans and released under a new name: “lincity-ng”. The “NG” word stands for “Next Generation”. The problem is that you can get bored rapidly because of the monotony of the gameplay.

PocketCity

This game project is now mature. It's something like SimCity 1 for hand-held devices.

Simutrans

Simutrans is another simulation freeware. In this game, you must manage the transport and traffic between cities.

GTA

In my opinion, GTA is not a simulation game. It's more than that. I've played GTA San Adreas recently and it's really great. You can interact with a lot of environmental structures around you. The objects such as car, bike... can be damaged and explode. I think that if we could integrate the GTA's style in OpenCity it would be fabulous.

The others

Well, there are many other unfinished game project such as: TheCity, CCity ... There are also OpenCity from Canada, and OpenCity from China.

II.What is OpenCity ?

I think that everyone know about Simcity from Maxis. OpenCity is not intended to be a clone of SimCity or any commercial entertainment software. It's just my own city simulator project.

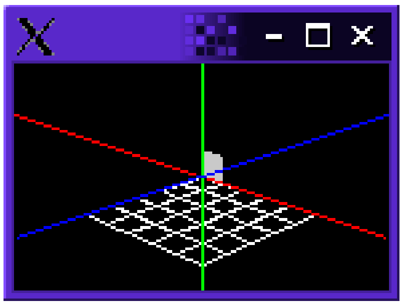
With OpenCity, a player can zone*, build governmental structures and transform the terrain. You have of course some funds at the beginning, and you'll earn money from taxes. Since OpenCity is a city simulation game, I'll try to simulate as many things as I can. By the way, I'd like to give an important role to trees in the game.

The first project's development milestone consists of bringing the SimCity 2000 gameplay into OpenCity. After that, we will add more features like networking, city exploring, and interactivity.

III.The story

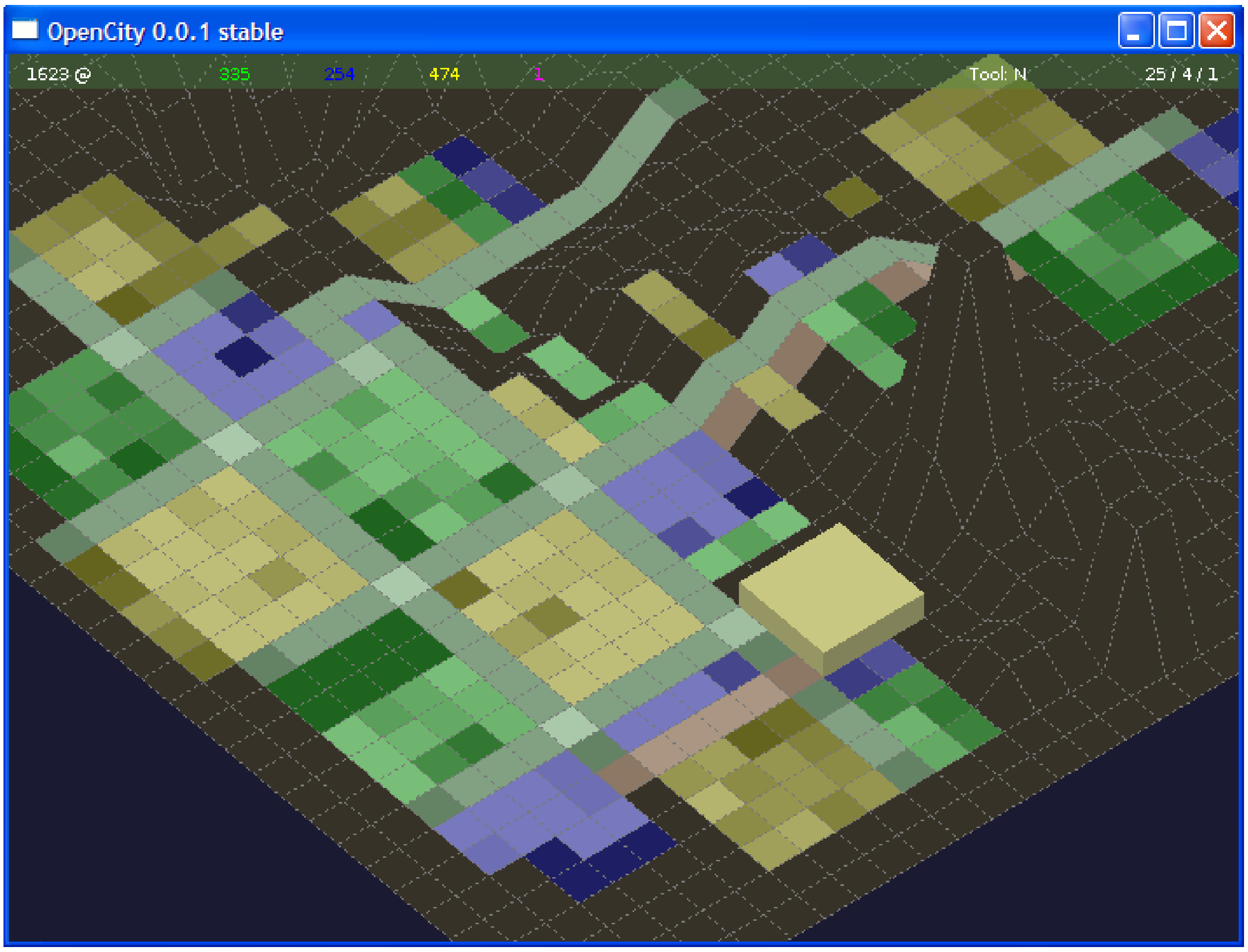
There is another similar game under Linux called FreeReign. However, the development of that project was stopped in 2001. At the beginning, I wanted to continue that project but its codes are weird and there is very few documentation about its design. That's why I decided to start my own project: “OpenCity”.

I started the game's design since the middle of April 2003. It was quite hard to work on the design only without writing any code. Furthermore, I had no experience with OpenGL and Linux game programming.


Screenshot 1 - The simple grid

Few months later, I decided to transform the design work into codes. As shown in the screenshot 1, what I managed to do at the beginning was a little grid displayed in a little window. This piece of codes helped me to understand how OpenGL works. Since I use IRC, sometimes I often see people asking how to write a 3D game engine. However, they don't even mind to start to write something simple just like “the simple grid” above. Honestly, I don't like such a question.

A lot of work and test had been done before the first version of OpenCity. The following screenshot comes from OpenCity-0.0.1stable. I was very proud to present it to the rest of the world. There was no graphics yet in the game but you could build the roads, the three famous types of zone, and the electric plant. That's not all because you can also raise and lower the terrain. Furthermore, the screenshot was taken under win32. Since OpenCity is written with OpenGL and SDL, it was not difficult to port it to win32.



Screenshot 2OpenCity 0.0.1 stable

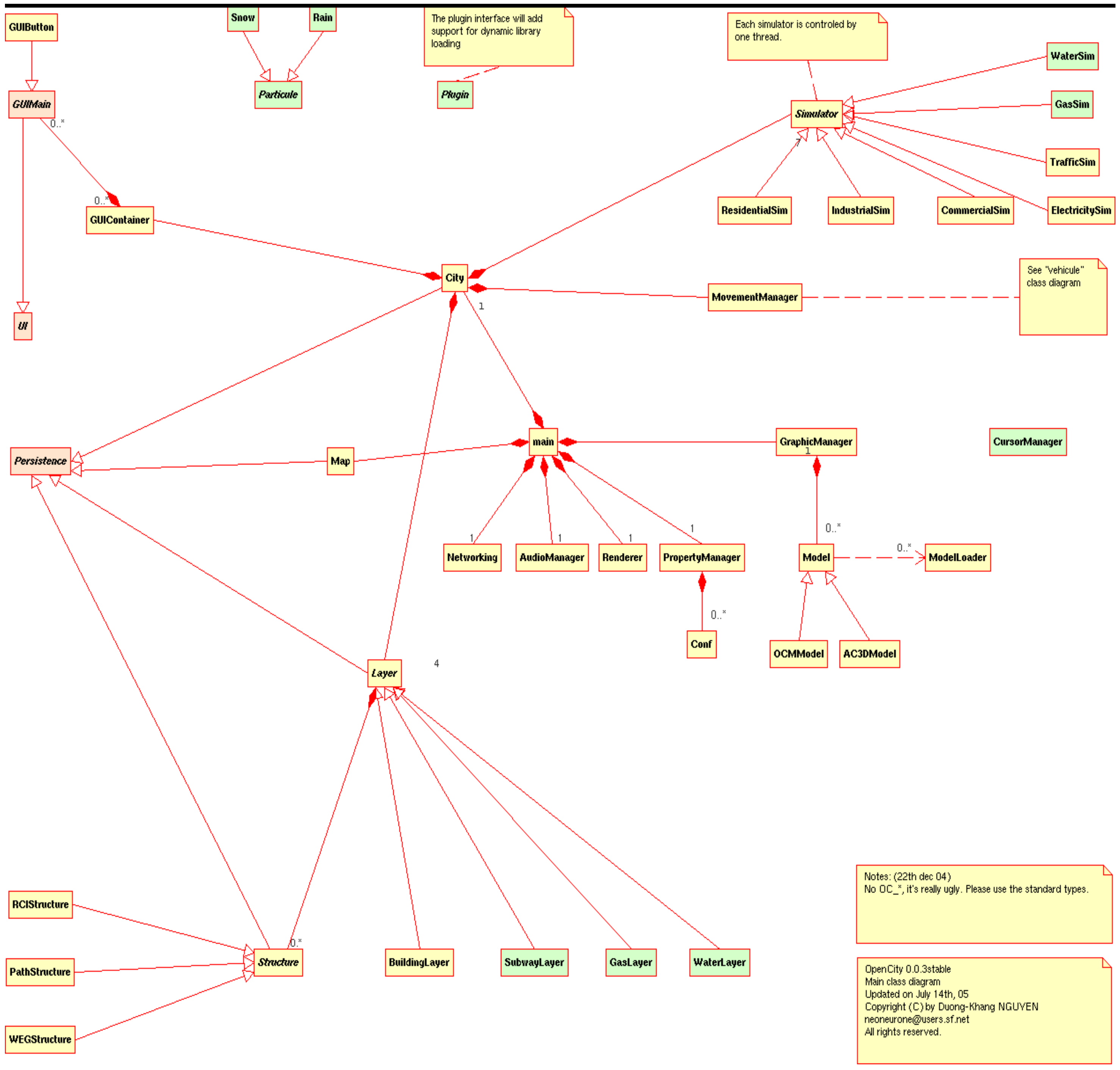
IV.UML design

Before writing any codes, I need to know what my future program will do. There is a lot of people who only know typing codes. And I was one of them. However, in a complex system, working like that will lead to buggy and difficult to maintain codes. In order to produce a high quality home made game, I decided to use UML to design some aspects of the program.

Please refer to the file “UML.xmi” for more details. It was made with Umbrello which can be downloaded for free at:

http://uml.sf.net

You can download the “UML.xmi” file from the CVS. The XMI file format is also a standard XML file which was created to simplify the information export between UML design tools. A tiny screenshot of the global class diagram of OpenCity has been put below.



The plugin interface will add support for dynamic library loading

Each simulator is controled by one thread.

See "vehicule" class diagram

Notes: (22th dec 04)
No OC_*, it's really ugly. Please use the standard types.

OpenCity 0.0.3stable
Main class diagram
Updated on July 14th, 05
Copyright (C) by Duong-Khang NGUYEN
neoneurone@users.sf.net
All rights reserved.

V.Game design

Range table

	W	***E***	***G***	***R***	***C***	***I***	***P***
R	5	2	5		4	7	3
C	5	2		4		8	2
I		2		7	8		4

Cost table

Structure	***Cost***		***Maintenance monthly***	***Products / Income***
	Build	***Remove***		
Residential	5	2*(level+1)		1 x level
Commercial	5	2*(level+1)		1 x level
Industrial	8	3*(level+1)		1 x level
Water				
Electric line	3	1	1	
Electric plant coal	2000	200	20	2000
Gas				
Road	5	4	1	

Traffic simulator

Each zone of path (route, highway, rail ...) is simulated by the « traffic simulator ». The algorithm is:

1. look for a random zone of path
2. reset the level to zero (a byte variable)
3. look for structures in range (2) <=> 25 units square
    1. sum all levels of not path structures = A
    2. sum all path structures (this is > 0 necessarily) = B

4. the current traffic value is given by C = A / B

Model file naming convention

The top level directory must be called "model". Inside that folder, there will be sub-directories listed the table below. Each model must be created in an unique folder inside those sub-directories. Each model needs to have a ".conf" suffixed file which is described below with some informations in it.

Name	***Internal code***	***Description***
residential	10 to 19	Residential models
commercial	20 to 29	Commercial models
industrial	30 to 39	Industrial models
water	190 to 204	Water models: pipe, pump, lake ...
electricity	170 to 184	Electric lines, power plant etc...
gas	210 to 224	Gas pipe
park	320 to 329	Park models
	330 to 339	
	340 to 349	
firedept	350 to 359	Fire department
policedept	360 to 369	Police department
hospitaldept	370 to 379	Hospital department
militarydept	380 to 389	Military department
educationdept	390 to 399	Educational department, school, college, university...
vehicle	400 to 499	Vehicles: fire, police, military, civil car, ...

Tableau 1Model directory

For example, you has just created a "chez-bob" model. Since it's a "**c**ommercial" structure, you must place it inside the "model/commercial/chez-bob/" sub-folder. Any textures referenced by your model must also be created inside "model/commercial/chez-bob/" directory.

Next, you need to create a "chez-bob.conf" file which contains the following parameters:

*internalCode=integer (see main.h file). For "chez-bob" model, put 20 here because it's a "C"

*width=integer

*length=integer

*height=integer

*w=integer

*e=integer

*g=integer

buildCost=integer

destroyCost=integer. Should be something like (buildCost / 8).

maintenanceCost=integer. Should be something like (buildCost / 12) per month.

author=string

copyright=string

license=[GPL|LGPL| blabla ...].

description=string

name=string

url=string. This is used for external reference.

NOTE: Asterisk (*) denotes required parameters. The order of the parameters doesn't matter. The default language for the parameters is English.

If you would like to specify another language then append the country code to the parameter. For example, “description=House” is a description written in English (default) and “description[fr]=Maison” is the same description written in French. You can append the country code to the following parameters:

description

name

VI.OCM file type

It is a plain text file. Everything begins with the character '#' is ignored when OpenCity parses the file. Below is the description of the integer codes that you can file in that file. You can also look into "ocm.h" for more informations.

10) Vertices

The parser will expects 3 float parameters at the next line for the GL x, y, z coordinates of that vertex. The order of the vertices must be clock counter-wise. The y coordinates are modified dynamically by the renderer.

The vertices are displayed in the following order with glBegin(GL_QUAD)

14

23

11) Texture coordinates

Takes 3 float parameters at the next line for the GL s, t, r texture coordinates.

20) Color

It takes 4 float parameters at the next line. This change the current colour value of the renderer. When the light is on, it changes the light's ambient and diffuse value.

30) Texture file path

Indicates to the loader that it must load the texture with the path given at the next line. The first loaded texture will have the index 0. It's relative to the order of the loaded texture. You need to load all the textures you need before indexing them.

Note: Textures should be saved in .png file format. Don't forget that OpenCity _needs_ an alpha channel in your texture file.

31) Bind / Use texture

It takes one integer value: the index of the loaded texture. Don't forget to bind the texture to -1 when you've done with texturing. This tells the renderer to restore the correct OpenGL values for other models.

40) Model's level of detail (not yet implemented)

By default, if no level of detail (LOD) command is specified, OpenCity considers the model as common for all available levels of detail. The command must be used at the beginning of the model's definition.

This command takes one integer value: the LOD of the model. The available LOD are:

0: general model, the model is a general model, it's used whenever OpenCity displays the model regardless the current game's LOD.

1: low quality

2: average quality

3: high quality

4: top quality

VII.Clients-Server or Clients-Servers networking ?

Networking

Nowadays, more and more games offer the possibility to play on-line and share game experience with anyone over the Internet. Why not OpenCity ? The main idea is: in fact you are not alone to build a big city, and you are not alone to decide everything neither. So in the future releases, you can share the management of your city with the others via Internet.

If you want to run a server, there will be 2 modes:

"One-man-show mode"

"Cooperation mode"

As a server, you can limit the number of clients who would like to cooperate or visit your favorite city. Whereas a client will be able to choose between:

"Visitor mode"

"Cooperation mode"

When an OpenCity server is opened to anyone in "Cooperation mode", a client will be able to join it either in "Visitor mode" or in "Cooperation mode". When an OpenCity server is run in "One-man-show mode", the clients can join it with "Visitor mode" only.

Networking interface (c: client, s: server)

IP **GetClientIP**(int index): (s), return the IP of the specified client

uint **GetClientNum**: (s), return the current number of active connections

uint **GetClientMax**: (s), return the maximum number of the connections that the server can handle. If the queried machine is running OpenCity as a client application, the returned value is undefined.

enum **GetMachineRole**: (c/s), return the following enumerations:

OC_SERVER_STANDARD

OC_SERVER_DISTRIBUTED

OC_CLIENT_STANDARD

OC_CLIENT_SERVER

uint **GetPing**: (c/s), return the maximum ping time in millisecond.

uint **GetServerNum**: (c/s), return the number of servers that the machine knows

IP **GetServerIP**(int index): (c/s), return the IP of the specified server string **GetVersion**: (c/s), return the string which describes the OpenCity version of the remote machine. Example: 0.0.2beta

string **GetProtocol**: (c/s), return the string which describes the latest OpenCity networking protocol that the machine can understand. The returned string is in the form: x.y.z like 0.0.1

enum **Open**(IP adr): (c/s), try to open a connection to the host specified by its IP

OC_CONNECTION_ACCEPTED

OC_CONNECTION_REJECTED

OC_CONNECTION_TIMEOUT

enum **Close**(): (c), close the connection to the server

enum **Close**(uint id): (s), close the connection specified by its index.

enum **Accept**(uint id): (s), accept the connection request from a machine.

enum **Reject**(uint id): (s), reject the connection request from a machine.

enum **StartServer**(): (c), create a new socket and begin to listen for incoming connection requests. Return the following enumerations:

OC_SERVER_READY

OC_SERVER_STARTED

enum **StopServer**(): (s), close the server socket and free all server's ressources. Return the following enumerations:

OC_SERVER_STOPED

enum **Send**(void* data, uint len): (c/s), send the data to the server

enum **Send**(void* data, uint len, uint cid): (c/s), send the data to client specified by its index.

enum **Receive**(void* data, uint maxlen): (c/s), read the data sent sent by the server.

enum **Receive**(uint sid, void* data, uint maxlen): (c/s), read the data sent sent by the server specified by its index.

Networking protocol v 0.0.3

The hosts talk together by sending information packages. Those information packages have the following format:

Command	***Command's length***	***Data's length***	***Source IP***	***Dest IP***	***Reserved***	***Data***
16 chars	4 bytes integer	4 bytes integer	6 bytes	6 bytes	28 bytes	512 bytes

The maximum size of the header is 36 + 28 = 64 bytes. The header contains from left to right: the command in capitalized characters, the command's length in integer (greater than 0), the data's length in integer (greater or equal to 0), the source and destination addresses in SDL_net's IPadress type. Like any professional protocol, there's also a reserved space in the message's header for future use. The data part should not be greater than 512 bytes in length in the current protocol version (v 0.0.3).

The commands are described below.

CNT

The program sends this command to an OpenCity server when it wants to establish a connection. The data's part should contain a 4 bytes integer which indicates which version of the protocol the program can understand.

The following formula must be used to translate a protocol's version in a string format into an integer format:

string version: a.b.c

integer version: a*65536 + b*256 + c

Depending on the client's status, the server can respond with the ACK or NCK commands. The server uses ACK for an affirmative answer, and NCK otherwise. An optional reason can be sent back to the host in the data part of the ACK or NCK command.

DNT

A host sends this command to an OpenCity server when it wants to notify its willing to disconnect from the server.

The host should wait for a ACK command from the server before going to disconnect completely. The time-out for this command should not be longer than 2000ms. Even if the server doesn't answer within that period, the host can disconnect and shut down. It's recommended that in such case, the host should display a warning message to the user.

ACK

This command is used by a host (client or server) to reply in an affirmative way. In English, it means: “yes”. The ACK command can contain an optional information in its data part.

NCK

This is the opposite of the ACK command. NCK is used to answer in a negative way. In English, it means: “no”. Like the ACK command, the NCK command can contain an optional information in its data part.

PIN and PON

If there's no activity between a client and a server for more than 15s, the server should send the PIN command to the client. If the client is still active, it must answer with a PON command within 2000ms, otherwise, the server will disconnect it without any further notification.

The PIN and PON commands can also contain optional information in its body (data) part.

MSG

From time to time, a client want to talk with another via the server. It can use this command with the destination address in the header. If there's no address specified in the header (0.0.0.0), the message will be sent to call connected clients known by the server. In order to prevent an abuse use of this command, the data's length is limited to 128 bytes. A client can not send more than 1 broadcast message per second otherwise it will be disconnected without any warning from the server.

VIII.Miscellaneous

Q1: Verify if a realization of an ADT (Abstract Data Type) must have a virtual destructor ?

A1: Every class that derives from an ADT must have a virtual destructor. Since almost methods herited from an ADT are virtual the destructor must be declared as "virtual". When you destroy a polymorphic pointer, the virtual destructor will be called instead of the destructor of the base class, the ADT's one in this case.

IX.What are the typo-rules ?

- Everything I write go rarely beyond of the 80th column.
- Each function or member method should not go beyond one (or two in the worst case) screen page(edited in kwrite with 1024x768 resolution)
- All the "defines" are CAPITALIZED, and prefixed with "OC_*" . However, all the "enum" types are prefixed with "OPENCITY_*"
- There is one horizontal line which separates a function from another
- Each member method's name will be prefixed by the class name. For example: all the methods of the class "City" will have their names beginning with "city" (deprecated, will be removed soon)
- See C++STYLE (from the GCC 3.2.2 sources) for more C++ coding style
- Sometimes, each method may have a name as long as "rendererDisplayLayerHighlight();". It's because I want to make my project easy to read.
- I use Doxygen for source documentation, and you should do the same.
- Each private member method name is prefixed with its class name in low-case. This rule doesn't concern the functions in the “main.cpp” file.
- The tab size is 4 spaces
- I'm an indentation and comment maniac, you should be one too !
- Comments should be written with a lower indentation level than the concerned codes.
- Keep your codes look like the codes you're patching.

X.Credits

Programmer & designer:

Duong-Khang NGUYEN

email: neoneurone at users dot sourceforge dot net

Lead 2D/3D artist and brainstormer:

Frédéric RODRIGO