**Introduction**

This is a small experimental game. The demo game is for two players that need to build armies and send them into battle against each other. This game is created mostly for the enjoyment of the author

The Tubastga itself implements a very simple and (unfortunately) a boring game.

The intention is to show how that the engine can be used and give code examples on how to use it.

Tubast’ga is in development.

This document should be viewed as a mix of design-ideas and tutorial.

**The Game**

**To Install**

**Socket Variant**

If you have two PCs (from now on called PC-Red and PC-Green) connected via LAN (playing over internet is described in a separate section due to Firewall configurations).

Get ip adress for both of them. Decide which one should run the server logic.

In the following we assume that PC-Red runs the server.

Copy the file Tubastga\_Server.exe to the PC-Red in a designated folder.

Copy the file Tubastga\_Client.exe [more files needed due to Gtk…] to designated folders in both PC-Red and PC-Green.

**PolyORB Variant**

If you have two PCs (from now on called PC-Red and PC-Green) in a LAN network

In the following we assume that PC-Red runs the server.

Get ip adress for both of them. Decide which one should run the server logic.

In the folder Tubastga there is a file called SETUP.cmd.

Open this file in a text editor, and edit the ip adress there so that it has the ip adress of your server (PC-Red).

On both PCs create a folder called C:\Tubastga\Client )

From the folder Tubastga copy the files tubastga\_client.exe and SETUP.cmd to the folder

C:\Tubastga\Client on both PC-Red and PC-Green.

In the folder Tubastga open the file polyorb.conf in a text editor.

Find the line containing "polyorb.protocols.iiop.default\_addr"

Set this variable to the ip adress of your server (PC-Red).

From the folder Tubastga, copy tubastga\_server.exe and polyorb.conf to a folder called C:\Tubastga\Server on PC-Red.

You may need to disable firewalls to make this work. That is not good, but I propose, that in the first few attempts you disable firewalls totally just to see if anything is working.

When you get things to work you should enable firewalls and configure them more restrictive.

**Starting the game**

**Socket Variant**

In the folder C:\Tubastga on PC-Red, open a Command Prompt.

Execute

|  |
| --- |
| Tubastga\_Server ip:aaa.bbb.ccc.ddd port:*NNNNN* |

The server will start and will print out a message indicating the IP address of the server.

The clients you wish to start need to use this IP address and the port indicated by NNNNN.

You can choose any port you like between 1025 and 65535 as long as they are available for use.

You are now ready to start the clients. On PC-Red start another Command Prompt and go to the folder C:\Tubastga\Client. Execute

|  |
| --- |
| Tubastga\_Client ip:aaa.bbb.ccc.ddd port:40000 player:MyName |

String “aaa.bbb.ccc.ddd” stands for the IP address of the server we started earlier. “MyName” is your name in the game. If you don’t enter a name, you will be called “Player 1” or “Player 2”.

On the other PC-Green start a Command Prompt and execute the same as above. Give the name of the other player.

After a few seconds the two clients will start.

**PolyORB Variant**

On PC-Red start two Command Prompts.

In one of the Command Prompts execute

|  |
| --- |
| Tubastga\_Server |

In the other execute

|  |
| --- |
| Tubastga\_Client player:MyName |

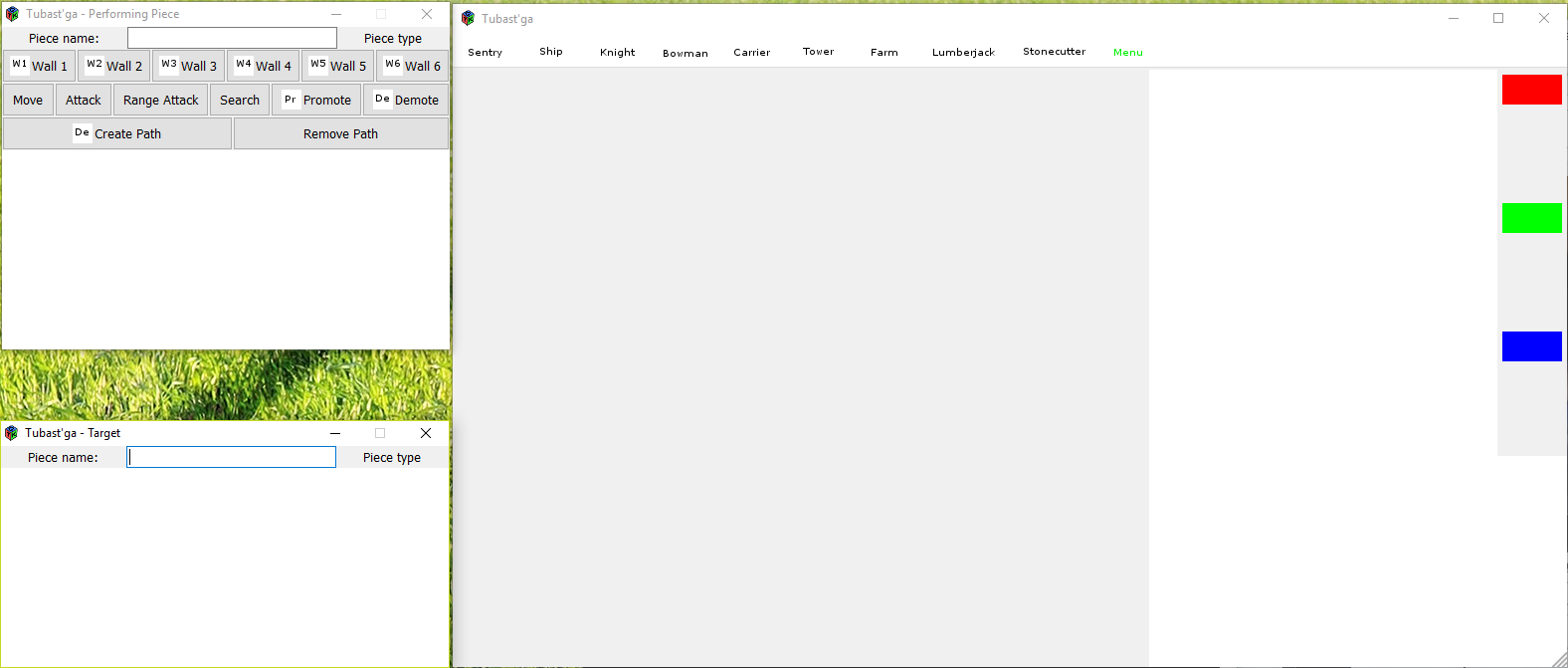
On PC-Green start the tubastga\_client.exe in the same way as on PC-Red.

You should now see the clients of Tubastga running, and the server should have prompted some output showing that it is running.

**Starting a Game**

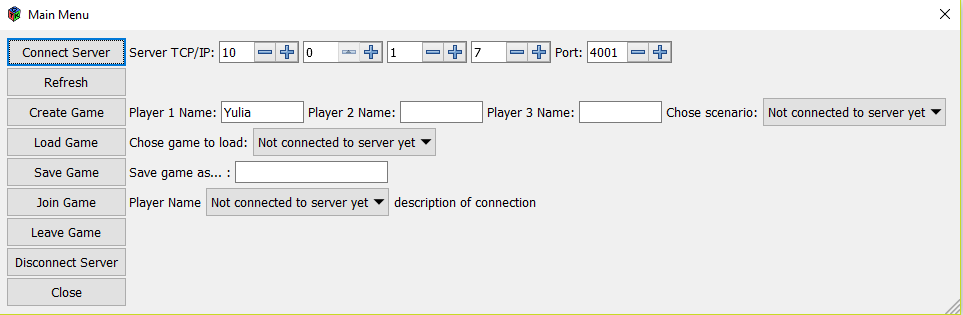
|  |
| --- |
| The screenshots in this tutorial shows a red, green and blue box. The GUI is currently not able to distinguish between scenarios with 2 or 3 players, so it always shows 3 players. The scenario “demo\_1.dat” is configured to be a 2 player game. Further work must be done on this to make it clear. |

The following screen appears.



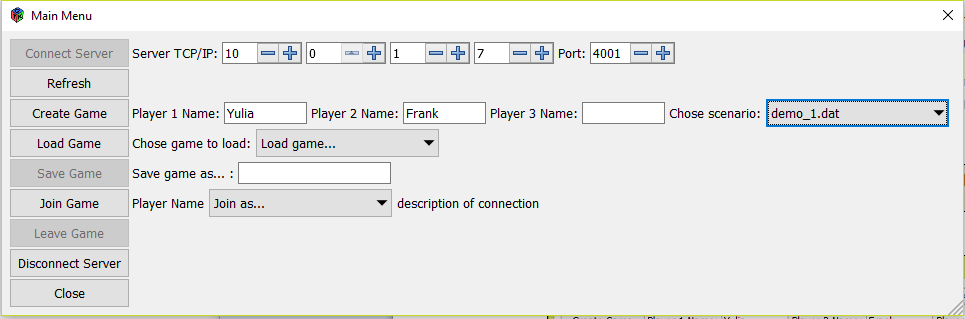
Press then “Menu” button in the toolbar.

The following dialogue appears.

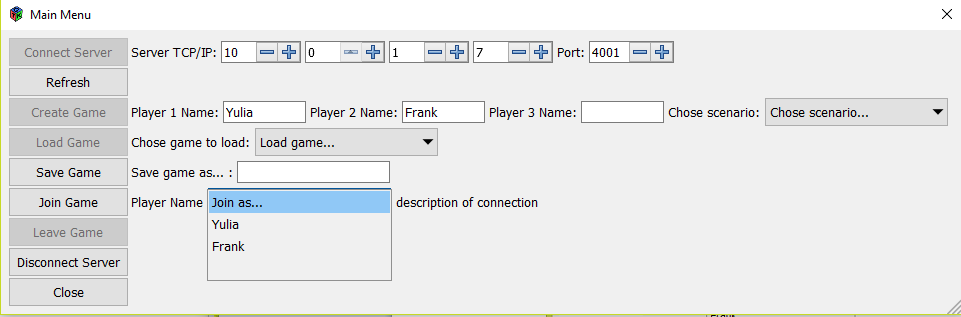


Press “Connect Server”

Enter the two players name in the fields “Player 1 Name” and “Player 2 Name”. Choose a map/scenario. Press “Create Game”.



Now chose which player you are on this particular PC. Then press “Join Game”.



You will be connected to the game in the server.

**The Interface**

When you are connected to the game, the map should be loaded, and the screen should look similar to the screen below.



This screenshot shows the Red players view.

At the top of the screen you see several pushbuttons.

“End Turn” means the player that currently has his turn, can press here and his turn ends and the other player get to do his moves.

The “Menu” button is only used during creation of game, load or save game.

The other pushbuttons are used when you wish to place pieces on the map. First press the pushbutton of the piece you wish to place and then chose the location of the piece.

To the right we see a red and a green box symbolizing the players. When the Red has the circle next to this box it means that it is his turn. When it is Green players turn, the circle will move to his/her box.

The region to the right-bottom of the scree, shows information regarding the area on the map over which the mouse pointer is currently hovering. Also it is displayed information about the towers of this player – and it says how many resources are stored in each of the towers.

**The Map**

The map in the game is 100 \* 100 hexagons. These hexagons are called Patches in the game and in this tutorial.

All players start with one tower each – indicated with a “T” on the map.

One player starts in the bottom-left corner, the other player in the top-right corner.

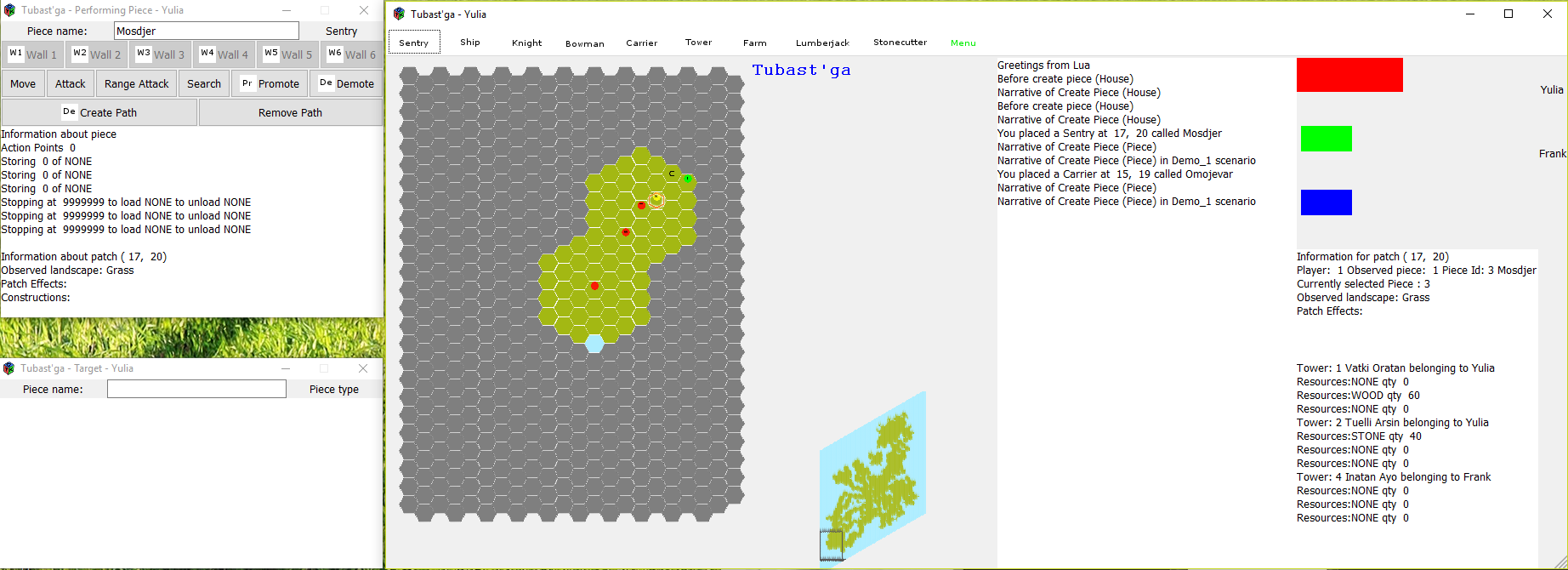
Each player can only see the terrain and pieces on the areas that are within visibility of the pieces they have.

If you hover the mouse over an area on the map, the information field to the right will show the coordinates of the area, which terrain is on that area, any effects that are affecting the area and list all the pieces that are there.

There can be up to 6 pieces on one patch of the map. All pieces in one patch must belong to the same player.

You can only see a part of the map at once. To navigate the view of the map, your left/right/up/down buttons on keyboard. You can also move the map from piece to piece by pressing TAB button to cycle through all pieces. You can select all the pieces that you can see on the map.

You can zoom in/out of the map by using the wheel on your mouse.



**Figur 1**

There is also a minimap so you can see where you are on the total map. You can navigate on the minimap by clicking on it.

A player can only place a piece in the visible area of one of his Towers.

**The Houses**

Houses can’t move and can’t perform attack or defense. However, a house can provide benefits for other pieces. Some houses are used to perform production of material that you need to produce other pieces or houses.

**Tower**

A tower has several functions.

It reveals a big area of the map so that the player can see any attacking pieces and terrain, and it is in this area that any new piece or building can be placed.

The towers are also the storage for the resources. Other houses nearby, that are producing goods, will deliver their goods to a nearby tower. A tower can only contain a particular number of different goods and a particular quantity of each goods.

A tower is costly and you need to build stonecutters and lumberjacks to be able to build any big number of them.

**Lumberjack**

A lumberjack is a house that produces lumber. It should be placed near or in a forest area to have maximum effect. The quantity of lumber that it produces is depending on the number of forest areas nearby.

The lumber produced by the lumber will be stored in a nearby tower.

**Farm**

A farm is a house – it should be placed near a grassland area to have maximum effect. The quantity of food that it produces is depending on the number of grassland areas nearby.

The food produced by the lumber will be stored in a nearby tower.

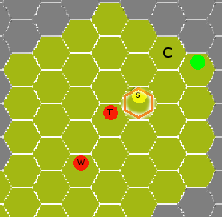
**Stonecutter**

A stonecutter is a house – it should be placed near a mountain area to have maximum effect. The quantity of stones that it produces is depending on the number of mountain areas nearby.

The stones produced by the stonecutter will be stored in a nearby tower.

**The Fighting Pieces**

The fighting pieces are able to move around within certain terrain constraints.



**Figur 2**

**Pieces Movements**

To move a piece, **left-click** on it. Right click on that patch to which you wish to move. Press “Move” in the “Performing Piece” – dialogue.

**Pieces Attacks**

To be able to attack your piece need to be at attack distance from the enemy. E.g a sentry needs to be on the patch next to the enemy.

**Left-click** on the piece you wish to attack with, and **right-click** piece you wish to attack. Press “Attack” in the “Performing Piece” – dialogue. If the attacking piece lose, it will be removed. If the attacking piece wins, the attacked piece will be removed. If the patch of the attacked piece is empty, the attacking piece will enter it. If not it will stay where it is.

**Pieces Range Attacks**

To be able to attack your piece from a range it needs to be at range attack distance from the enemy. E.g a bowman needs to ?? from the the enemy.

**Left-click** on the piece you wish to attack with, and **right-click** piece you wish to attack. Press “Range Attack” in the “Performing Piece” – dialogue. If the attacking piece lose, no pieces are removed or moved. If the attacking piece wins, the attacked piece will be removed.

**Sentry**

The sentry is a soldier that have good defense, but poor movement and attack capabilities.

**Ship**

The ship can move on water.

**Knight**

The knight has a longer range of movement and attack.

**Bowman**

The bowman can perform ranged attack on other pieces. It has very poor defense.

**Action - Defence**

Design idea where a player can attack another piece – and the other player can have a chance to respond.

1. Perform\_Attack ( P\_Player\_Id, Action Type, P\_Attacking\_Piece\_Id, P\_Attacked\_Piece\_Id)
2. Before\_Perform\_Attack (P\_Player\_Id, P\_Action\_Type, P\_Attacking\_Piece, P\_Attacked\_Piece, P\_From\_Pos, P\_To\_Pos)
   1. Grant\_Piece\_Effect (P\_Player\_Id, P\_Action\_Type, P\_Piece\_Id, P\_Effect)  
      Effect: “Attacked\_By” Piece\_Id: NN
   2. Return: Failed
3. The Game loop proceeds….
4. The opponent’s client receives this effect on his/her piece.
5. The client informs the player about the attack.
6. Player can respond by:
   1. Adding effect to his / her piece
   2. Adding effect to patches
   3. Player can perform any attack also
   4. When the defending player is done – he / she can add the effect “Defended”
7. …..
8. When Before\_Perform\_Attack finds “Defended” the attack can be performed.

Intern logikk i Before\_Perform\_Attack:

1. If no Attacked\_By then
   1. add Attacked\_By
   2. returner Failed
2. If Attacked\_By and no Defended then
   1. Returner Failed
3. If Attacked By and Defended then
   1. Do the attack

**Action Point**

Idea 1: The players can assign their commands simultaneously. The number of commands that is limited by the number of messengers. One command must be brought by a messenger from you to the piece that is supposed to have it. The messengers are not shown, but you see them in your panel to the right. When you all messengers are out, you cannot give more commands but you need to wait for some messenger to return. The time that the messenger needs is depening on the distance from your main tower to the piece.

You can move main tower.

Idea 2: As idea 1 but use pigeons - they fly much faster.

Idea 3: Towers works by semaphor. They need visual contact. When the piece is in the rang of a tower it receives the message. How to put a limitation to the messages in this?

Idea 4: The limitation on Action Points can be related to supplies.

Usually your Action Points are a resource that is valid for some period of time or during a turn.

Idea 5: The player receive a number of AP per *day*. They can spend up to this number of AP during the day. At the end of the day, they get new Action Points.

Idea 6: When a piece tries to e.g move from one patch to another, it has a number of Action Points in order to do this. If the move require more AP then the piece has, the "Before Perform Move" returns a failure code. Every "Tubastga Update" it is checked if the piece should have more Action Points. The command is in the command queue until it is done. We would like to avoid that one player can use very long time on his actions, and make the other players wait.

**Effects**

There are effects that influence pieces positively or negatively.

**Action Points**

The first effect you get to know is the *Action Point* (AP). This indicates how many action points you have left this turn.

**Promotions**

In some situations several pieces can be eligible to have an effect, but only one piece can get it. In these situations, the piece that was created first gets the priority to get the effect.

The player adjusts this a little, by promoting some of his/her pieces. You do this by pressing the button “Promote” and then select the piece. You can only promote to “CAPTAIN”. Pieces with the rank CAPTAIN will be assigned all effects before other pieces. This means that if several pieces have the possibility of getting an effect, then CAPTAN pieces will be chosen first. If there are several pieces of rank CAPTAIN that can get an effect then the one that was created first gets the priority of getting effects first.

**First\_Defence**

If four Sentry Pieces are standing on adjacent patches at the beginning of a turn and at the end of the same turn, they get the effect “FIRST\_DEFENCE”. The pieces must stand on adjacent patches. They don’t get the effect by standing on the same patch.

This effect gives a defensive bonus added to each of the Sentry Pieces in the formation.

When one or more of the Sentry Pieces breaks out of the formation it keeps the effect during next turn – but looses the effect at the end of next turn.

**First\_Attack**

If you have five Knight Pieces standing in a line at the beginning of a turn and at the end of the same turn, they get the effect “FIRST\_ATTACK”. The pieces must stand on adjacent patches, they don’t get the the effect by standing on the same patch.

This effect gives an attack bonus added to each of the Knight Pieces in the formation.

When one or more of the Knight Pieces breaks out of the formation it keeps the effect during next turn – but loses the effect at the end of next turn.

**Raining Arrows**

If you have a group of Bowmen Pieces standing in such that one of the Bowmen Pieces has four neighbor Bowmen, then this Bowman Piece gets the effect “RAINING\_ARROWS”. Thus effects disappear after an attack and re-appears again during another turn. *(How do I avoid this being too powerfull… only one piece can do the attack? How can player decide which one to be captain…? Do I know when a piece attacks? Calculate\_Attack? One attack spends all AP for this piece?)*

**Versatile**

A group of FIRST\_ATTACK standing together with a group of FIRST\_DEFENCE.

All pieces involved get the effect VERSATILE.

Pieces that have FIRST\_ATTACK get –N in attack and +M in defence. Pieces with FIRST\_DEFENCE get –M defence and +N in attack. This makes both types of pieces better in general, but not as good as a “clean” group. (*Also here player does not have possibility to choose… How can I solve this? Another controlling effect?*)

**Fatigue**

Fatigue is an effect that appears on some pieces when they have repeated the same operation too many times. The effect will set the ability to attack to 1 and the ability to defend to 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Piece | Becomes Fatigue when… | Consequence | Wears off when… |
| Knight | Moves more than 15 steps during 3 turns | Attack set to 1 and Defence set to 2 | It does no action for one turn |
| Sentry | Moves more than 8 steps during 3 turns | Attack set to 0 and Defence set to 2 | It does no action for one turn |
| Bowman | Moves more than 8 steps during 3 turns | Attack set to 0 and Defence set to 0 | It does no action for one turn |
| Knight | Attacks more than 2 times during 2 turns | Attack set to 1 and Defence set to 2 | It does no action for one turn |

If several reasons are causing the fatigue, then it will wear off when the last of the causes are invalid.

**Design**

This is a chapter containing ideas that I could put into the game.

Add a parameter to all attack, move, search operations that indicates the type. E.g Perform\_Attack type Physical/Fire/Ice/Black/White and so on.

A general Perform\_Command method.

Make it possible to use cards in the game. E.g Perform\_Card – the player has a card in his hand and wants to play it. The engine needs a way to communicate list of cards to the player. If we introduce cards as a concept in the engine I think it should be introduced fully so that card is really something in the engine. E.g Reports for cards on the players hand, reports for cards on the table etc.

Perform\_Siege\_Attack – solution to my problem that I had to be able to attack buildings. Simply give this operation another name Parameters are :

P\_Attacking\_Piece : Piece.Server.Fighting\_Piece.Type\_Piece\_Access\_Class;

P\_Attacked\_House : Piece.House\_Piece.Type\_House\_Access\_Class;

Etc…

Temples – let the player build temple to collect magic strength etc.

Player could grow and harvest food, magic stength etc on the ground.

Pattern of seeds on the ground could be used to improve crops.

**Development**

This game is my hobby-project for several years. It has taken me long time to get to this point it is now, because I started out in one way and have almost constantly changed my mind and turned into another design.

This is the current state as I see it:

This is a turn-based game engine – but I have started to create a real game on top of the engine. The real game is not very interesting. It is more like a proof of concept at the moment.

First of all the game engine is an experiment with the programming language Ada (Ada95/Ada2012) and its Distributed System Annex (DSA), implemented through PolyORB. I have been interested in the programming language of Ada for several years, mostly because of its built-in tasking support. It looks very much like Pascal. Also included in Ada95 is Distributed System Annex – this is a language standard for how to make Ada programs communicate with each other via network.

An introduction to what PolyORB is look at <http://libre.adacore.com/tools/polyorb>

I got some problems however while using PolyORB – I was not able to run a game server in my own LAN and have it presented to the internet. There seems to be an obstacle of some sort in PolyORB.

I have realized that DSA/PolyORB made this project so odd, that I have now developed a variant of the solution that uses plain sockets and TCP.

For the moment you are able to build both PolyORB version and socket version. If you wish to start tinkering with Sisyfos, Ada95/2012 and PolyORB you should not do it because it is simple :)

To be able to install Ada95/2012 compiler and development environment look in the doc folder in the Sisyfos folder.

There is a PDF there that I put together some years ago to help myself remember how to install Ada compiler, PolyORB and GtkAda.

I have not done any good job on memory management in the game. I have focused on making things playable.