Rescue! Max

website: http://rescue.sf.net/

The XML structure of the mission File:

I decided to keep the mission separate from the game engine. This also provides the opportunity to have many different missions. The decision to do it in xml was because unlike my previous project each space object has many different tangible features that may be implemented 1 by 1, and the xml format is very good at this.

```
<mission name="Star Trek: The Next Generation">
    <spaceObjects cluster="yes">

        All the space objects go here

    </spaceObjects>
    <guitheme external="no">

        The user interface design goes here

    </guitheme>
    <sound>

        The sounds go here

    </sound>
</mission>
```

Cluster is the default setting for if similar ships are clustered together at the beginning of the game. External indicates if the frames for the GUI are going to be external, both these settings can be changed by the user in the preferences and mission setup.

Space Objects

The class MissionObject is used when creating objects from the mission file when the exact number of these objects is still unknown. Each object is described in the xml file like this:

```
<object name="bird" image="bird.gif" small="s.gif" info="i.gif" number="2" type="ship">
```

The name is the default name to be used for that object if other names are not specified. The image is the default image and will appear on the large map and small is the small image of the ship that will be used on the long range scan. The info image is the image displayed in the ship information box, and is usually a 3D image of the ship, must be 150x150 resolution. Number is the default number of those objects and the type specifies what it is. Possible types are: ship, base, planet, wormhole. The player will be the last object in the XML file, so make sure you put the last one to be a ship. To specify an alternative to the default name and image:

```
<name value="evilone" image="evil.gif"/>
```

This has to be put under the section <names>. This can also be used without the image parameter.

This image will override the info image and will be resized to 150x150 if specified.

Ship Settings

These are all the setting that a ship can have, if a setting is not set for a ship then the default is used.

The defaults may vary in different versions of the game so its best to specify EACH value.

The format of these setting is like this: `<setting value="0"/>`

maxwarp	(default=4) any real value from 1 to 9 This is the maximum warp speed that the ship can use, this is the fastest speed in a game and is used only when flying to far destinations on the map.
maximpulse	(default=1) 0.25 or 0.5 or 0.75 or 1 This speed is used to travel to destinations on the big map, somewhere that is pretty close to the ship (impulse 1 = warp 1) the reason why there is a distinction between the 2 engines is because one is used only on the small map for long distance travel and the other only on the big map to go to close destinations and would be hard to control if going very fast.
jump	(default=1) 1 or 0, sets if the ship can go straight into warp If jump is set to false (0) then the ship has to speed up gradually before it can reach a required speed, and also slow down gradually before it can stop. If this is true then instead of gradual speeding up the ship warms up its engines for a few seconds and then jumps straight into the required speed.
turn	(default=1) 1 or 0, sets if the ship needs to slow down for turning This indicated whether the ship needs to slow down for turning or if it has an engine capable of making instant turns. Ships that have this set to false (0) then it does not need to slow down for turns and also does not rotate on the screen when turning. e.g. The Borg Cube in the “rescue” mission.
maxphaserintensity	(default=1) any value from 0 to 3 (0=none 1=min 2=mid 3=max) This is the maximum amount of damage that you can do with a phaser shot. If its set to 3 you can still fire it at 1 if you want to conserve power, but less damage is done. 0 means that the ship does not have any phaser capabilities.
maxtorpedosalvo	(default=1) any value from 0 to 3 (0=none) The maximum amount of torpedoes that the player can fire at the same time. 0 meaning the ship has no torpedo capabilities.
maxshieldpower	(default=50) any value from 0 to 100 (higher for specific cases) The maximum shield power is the strength that the shields can be set to when under attack from an enemy. When a ship is hit this value is knocked down, if this value is low then the systems on the ship start getting damaged which eventually leads to its destruction. The shields are recharged when not under

	threat.
maxtorpedosleft	(default=18) any value from 0 to 100 (higher for specific cases) This is the maximum number of torpedoes that the ship can carry on-board, when you run out of torpedoes you have to return to a base to stock up on more.
maxtractorpower	(default=50) any value from 0 to 100 (higher for specific cases) The maximum strength of the user's tractor beam. The tractor beam is used when trying to capture another ship.
maxcloakpower	(default=0) 0 or 1 or 2 or 3 or 4 (0=cannot cloak 4=invisible on all scans) The maximum cloak power is how invisible the ship can appear to other ships. Ships are detected using the sensor scan, this can detect ships that have a lower or equal cloak value to the sensor scan value. The strongest scan is 3 so if the ship is cloaked at 4 then there is no way of making it viable. Having the sensor scan on 3 wastes lots of energy so it is not good to keep it on this setting.
maxtotalenergy	(default=10000) any value from ~100 to 1,000,000,000 The maximum energy of the ship determines how long is can fly around and stay in battle for. When energy runs out the ship needs to either find a safe place to stay whilst it recharges or recharge at a base, this is a lot faster.
maxphaserbank	(default=2500) any value from 0 to 2500 (higher for specific cases) The maximum energy that can be stored in the phaser banks, this is recharged slowly so for a long battle it would be best to have a large store in the phaser banks, firing phasers at max drains the phaser banks the fastest but is most effective.

The Like System:

This has been a major change from the original, but will hopefully provide a more interesting game experience. The initial likes are set from the xml file (negative numbers indicate dislike). These numbers do not stay constant throughout the game, and can be influenced with the idea of "my enemy's enemy is my friend". This means that if the user comes into the game and start helping the enemy, they will actually start to like the user, and there will be situations in which the user will need the help of a lot of ships to attack very powerful enemies.

```
<like who="Starbase" amount="5" />
<like who="Borg Cube" amount="-5" />
```

<u>pre-sets and options</u>

each mission file has 4 pre-sets that can be set for each space object, also a maximum and a minimum. This allows the user to manually configure under the custom game option how many of each ship they want in the game.

```
<options min="0" max="25">
    <preset name="cadet" value="5" />
    <preset name="lieutenant" value="12" />
    <preset name="captain" value="15" />
    <preset name="admiral" value="17" />
</options>
```

These options can be adjusted in the Mission set-up dialog where you can use sliders to set the exact number of ships that one would like for a game. The mission set-up dialog is also used to load new missions from xml files.

<u>XML user interface</u>

The user interface is made up of containers, each container can have 5 other panels or containers in it, North, Centre, etc. If the container is not inside another container it has a name that becomes the name of the window and also a default x and y position:

```
<container name="Long Range Scan" x="759" y="10">
```

A container that is inside another container has a constraint that indicates where it will go in its parent container, the constraints are either north, south, east, west or center.

```
<container constraints="Center">
```

Possible panels that can go inside containers and make up the user interface are:
bigmap, smallmap, shipslist, interfacepanel
all these are added with a constraint and a width and height.

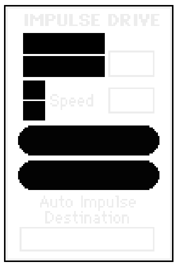
Interface panels are added in the form of:

```
<panel constraints="Center" w="203" h="343">
```

The Interface panels are loaded with 5 images:

```
<image nob="34" on="on.png" off="off.png" up="up.png" down="down.png"
map="map.gif">
```

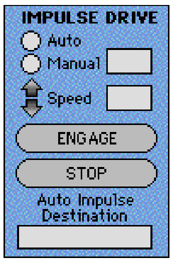
Here is an example of an *image map* file:



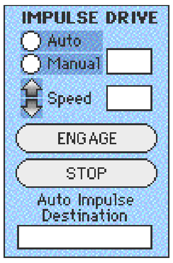
This file HAS to be a 256 colour grey scale GIF file.
Each almost black box in this image indicates a button,
All buttons need a unique colour to indicate where they are on the panel; the buttons can be any shape or size with gaps if needed.

The program reads in this file and constructs firstly a 2D array the size of the grey scale image which contains all the values of the grey pixels from the image, and secondly an images for each button in all its forms (highlighted, pressed down, activated, and greyed out) that it keeps in memory. The number of buttons is indicated by the nob (number of buttons) value.

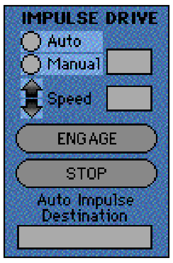
Here are some examples of files that can be used:



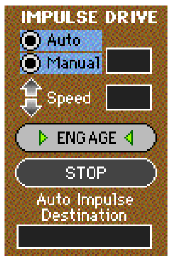
normal



highlighted



pressed down



activated

When the user moves their mouse over the panel an event is triggered that tells the panel the mouse is over a certain x and y position, it then looks up whether or not the mouse is over a button by looking into the 2D array, if it gets a value that's not 255 then it means that the mouse is over a button, then the button is painted onto the panel in its highlighted form, when the button is pressed down it is then painted in its pressed down form, then depending on whether the button should be activated or not either the active or normal button is pained on when the mouse moves off. This method ensures that there is never any unnecessary painting to the panel.

Other things that are on the GUI are; text, this shows ship speed and other things, bars that show things like the amount of ship energy. All these elements need to be updated when the information has changed. There is a method that is called every second that checks if any text or buttons or bars need to be updated on the GUI, and only repaints things that have changed.

The information that says what each button does is stored in the xml file, in the form of:

```
<color value="0" action="impulse_auto" />
```

There is also an option to add a tooltip to the button: `tooltip="go"`

For text information like the impulse speed:

```
<infotext x="64" y="50" w="25" h="13" lx="65" ly="61" type="speed" />
```

the x,y,w and h values indicate the rectangle that must be cleared before the number can painted. And lx and ly indicate where exactly the value is painted.

For progress bars that display things like total energy:

```
<infobar x="64" y="50" w="25" h="13" direction="right" type="energy" />
```

This means that a bar must be shown at x,y with width w and height h that moves in the right direction (direction can be right, left, up or down). The image that is used for bars when they are displayed are taken from the active image out of the four images discussed earlier.

Good luck and send me anything you make yura@yura.net and I will put it up on the site.

Written by Yura Mamyrin. http://yura.net/