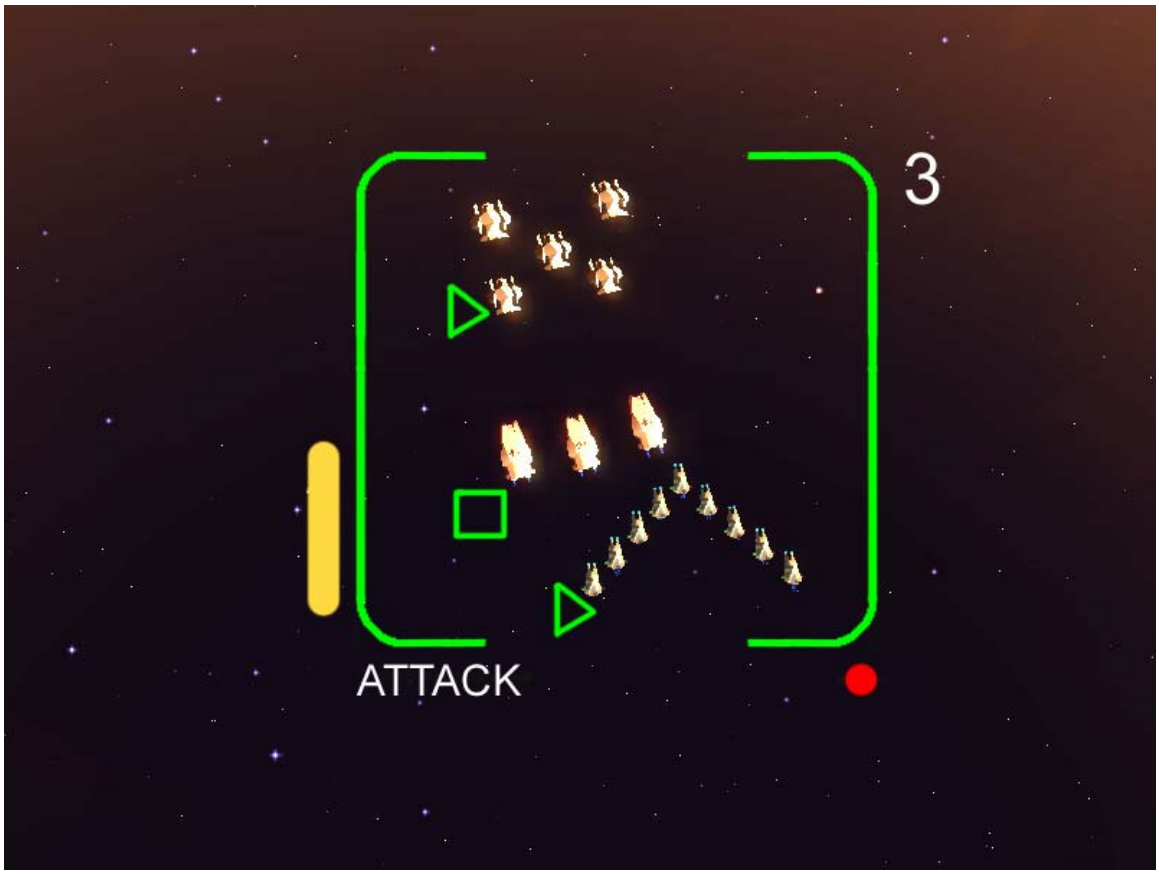
HOMEWORLD2

Advanced Tactical Interface (ATI)

Introduction:

The ATI is meant to integrate the functionality of Homeworld2's selection mechanism and Tactical Overlay (TO). This is readily apparent from the following comp:



Now compare this to a screenshot from the final game:

The ATI is also used for Sensors Manager world graphics, Sensors manager blobs and pings. This document covers the implementation of the ATI system and its use for the in-game selection/TO system as illustrated above. Please refer to SensorsManager.doc for more information on rendering of blobs, world graphics and pings.

Implementation:

The ATI is tightly integrated with the squadron system. The ATI basically renders the on-screen bounds of squadrons and displays information about them. The information for a given squadron ATI is described in Data/UI/ATI/ATI.lua.

ATI.lua

Data/UI/ATI/ATI.lua contains descriptions for the ATI for selections, mouse-over, Tactical Overlay etc. It contains a set of ATI templates, each with one or more ATIItems and optional tunable variables. There are also a number of definitions and helper functions to make setup of the items and placements simpler.

Most lines in ATI.lua end in a comma. This is because most lines describe an entry in a table or sub-table. It's safest to assume that any line should end in a comma.

Templates:

A template is a description of how a particular squadron's ATI will be displayed. For example, we would expect the ATI for unselected enemy squadrons to be quite different from that used for selected friendly squadrons. To that end, we have the following ATI templates:

Template	Description

friendly	Template for player-owned squadrons
allied	Template for squadrons of players allied with the player
enemy	Template for enemy players' squadrons
subsystemFriendly	A subsystem on a friendly ship
subsystemAllied	A subsystem on an allied ship
subsystemEnemy	A subsystem on an enemy ship

In addition, we have the following special templates, which are not strictly for squadrons:

Template	Description
resource	ATI for a resource. Just for mouse-over information.
tacticalOverlay	Template for rendering Tactical Overlays, both for individual ships and for "Meta-TOs", which are TOs for groups of ships too far from the camera to be rendered with individual TOs.
missilefriendly	Template for a friendly missile.
missileallied	Template for an allied missile.
missileenemy	Template for an enemy missile.
sphereRangeModifierAllied	For displaying the bounds of spherical modifiers (such as a cloak generator).
rallyPoint	For displaying a rally point.
multiSquad	For multiple selected player squadrons.

ATI Templates for **friendly**, **allied**, **enemy** and **resource** can have the following tunables:

Tunable	Description
tacticalIconColour	The colour used for the Tactical Overlay icon for this squad.
tacticalIconColourFar	The colour when squad far from camera.
clampATIWidth	The on-screen size of a squadron is not to exceed this width. Useful for preventing parts of the ATI for going off-screen when focussed on a unit or squadron.
clampATIHeight	The on-screen size of a squadron is not to exceed this height.
RUFormatString	wsprintf() format string for RU count (resource)
totalRUFormatString	wsprintf() format string for resource group RU total (resource)
unknownRUFormatString	wsprintf() format string for unknown RU (out of sensors range)
unknownTotalRUFormatString	wsprintf() format string for unknown resource group RU total (out of sensors range)
gateCostFormatString	wsprintf() format string for printing the cost to form a hyperspace gate
commandColourScalar	Modifies command colour for non-moused-over mousover feedback.

cloakColour	Colour of the cloak bar.
defenseFieldColour	Colour of the defense field strength bar.
EMPColour	Colour of the EMP shield bar
EMPLevelIgnore	EMP shield value above which ships have no EMP healthbar displayed.

Items:

Each template is composed of a number of items. Examples from the above image would include the brackets and the health bar. Each item has a placement and may have a graphic and one or more parameter indices.

Item placement:

The purpose of an item placement is to allow the designers full control over the position and size of items within an ATI template. These are entitled **placement2D** and **placement3D**. If an item contains a placement2D, it's a 2D item. 2D items are not Z-buffered with the rest of the screen. If an item contains a placement3D, it's a 3D item. 3D items are z-buffered with the rest of the world. Since the ATI for squadrons is inherintly 2D, placement3D will not be described here. Please refer to SensorsManager.doc for more details on placement3D.

placement2D:

placement2D is used to determine the location of an ATI item WRT the rectangle that encompasses the squadron.

The following table lists the possible members of placement2D. "Normalized screen coordinates" refers to screen coordinates that range from 0,0 (lower left of screen) to 1,1 (upper right of screen). "Normalized screen area" refers to an area that ranges from 0 (0 x 0, infinitly small) to 1 (1 x 1, full-screen). In ATI.lua, normalized screen coordinates are usually specified with the helper functions WIDTH, HEIGHT and AREA. There are also pre-defined sizes HUGE, LOD0 to LOD3 and TINY. If you need resolution-specific pixel-perfect placement, you can use the PixelWidth/PixelHeight functions.

Member	Type/Range	Default	Description
factorX factorY	float: 0<1	0	Multiplied by the ATI width/height to place the centre of the item within the ATI rectangle.
plusX plusY	float: -1<+1	0	Added to previous position regardless of size of ATI. In normalized screen coords.
factorWidth factorHeight	float: 0<1	0	Multiple of total width/height of ATI (0.0 .. 1.0). 1,1 will be the full size of the ATI rectangle.
plusWidth plusHeight	float:0<1	0	Added to the width. Normalized screen coords.
minWidth minHeight	float 0<	0	Minimum size of element (normalized screen coords).
maxWidth maxHeight	float: <1	1	Maximum size of element (normalized screen coords)

minATIArea	float: <1	0	Minimum area of the ATI that contains this element where this element is visible. Normalized screen area.
maxATIArea	float: <1	1	Maximum area of the ATI that contains this element where this element is visible. Normalized screen area.
visibility visibility0 visibility1 visibility2 visibility3	table of strings	0	Strings must be defined in the code to correspond to bit masks. Multiple bit masks will be or'd together to get a final mas. This number will be ANDed with the squadron's ATI visibility mask and must equal itself. {} will always be visible. There are 4 different possible visibility masks. visibility is an alias for visibility0.
invisibility invisibility0 invisibility1 invisibility2 invisibility3	table of strings	0	If the visibility test with the corresponding number described above passed, will be ANDed with the ATI visibility mask and if the result is nonzero, item will not be drawn. Useful for excluding visibility in special cases. Like visibility, there are 4 optional masks. invisibility is an alias for invisibility0.
placementFlags	table of one or more: "clampCentre"	nothing	Prevent plusX/Y from moving centre of element past centre of ATI. Useful if you have a negative factorX/Y and a positive plusX/Y or vice versa.
	"orthogonal"		Same height as width, corrected for aspect ratio.
	"attachToMouse"		Placement flags are relative to a rectangle that surrounds the mouse rather than the squadron's bounds.
	scaleToGraphic		Placement will be scaled to the size of the graphic.

Please refer to the following example:

```
placement2D =
{
    factorX = 1,
    factorY = 1,
    plusX = WIDTH(-8),
    plusY = 0,
    factorWidth = 0,
    factorHeight = 0,
    minWidth = WIDTH(8),
    minHeight = HEIGHT(8),
    maxWidth = HUGE,
    maxHeight = HUGE,
    minATIArea = TINY,
    maxATIArea = HUGE,
    placementFlags = {"clampCentre"},
    visibility = {"AVF_GroupNumber"},
},
```

This will place the specified item to the left of the upper-right of the squadron bounds. Its minimum size will be 8 x 8 pixels (at 1024 x 768) and its max size will be full-screen. It will be displayed irrespective of the ATI rectangle size. The “clampCentre” flag will prevent the item from being placed to the left of the ATI rectangle’s centre. It will be visibility only if the ATI is being rendered with AVF_GroupNumber(4) flag set.

Item graphics:

Most elements will have a graphic attached to them. A graphic is actually a container for an LOD sequence of sub-graphics and various other members. The available types of graphics are **mesh**, **text** and **texture**.

mesh

A mesh can be a solid polygonal mesh or a wireframe mesh. For solid meshes, export them from Maya as a “particle”. Their size should be 1x1 if they are rendered with “scale” renderflags. Otherwise, they should be modeled in normalized screen coordinates. The following members are available for mesh:

Member	Type/range	Default	Description
colour	vector4 {0,0,0,0} < {1,1,1,1}	{0,0,0,0}	The colour the mesh will be rendered in. Not needed if a colour parameter is set. If left default, the colour will come from Maya.
lineWeight	float: 1<	1	Thickness, in pixels, wireframe meshes will be rendered at. Only needed for wireframe meshes.
renderFlags	table of the following: “scaleX” “scaleY”	false	Mesh will be scaled to fit its placement rectangle. Mesh should be modelled at 1x1.
	“stretchX” “stretchY”	false	Mesh will be stretched to fit placement rectangle. Mesh should be modelled at its smallest possible size.
	“fromLeft” “fromRight”	false	If rendered with “stretchX”, mesh will be stretched from its left or right, rather than stretched about its horizontal space.
	“fromTop” “fromBottom”	false	If rendered with “stretchY”, mesh will be stretched from its top or bottom, rather than about its vertical centre.
	“clipTop” “clipBottom” “clipLeft” “clipBottom”	false	For items with a floatParam, mesh will be clipped off the specified direction according to the parameter. See parameter section.
	“justifyLeft”	true	Non-scaled, non-stretched meshes will have their left side lined up with left side of the placement rectangle.
	“justifyRight”	false	Non-scaled, non-stretched meshes will have their right side lined up with right side of the placement rectangle.

	"justifyTop"	false	Non-scaled, non-stretched meshes will have their top side lined up with top side of the placement rectangle.
	"justifyBottom"	true	Non-scaled, non-stretched meshes will have their bottom side lined up with bottom side of the placement rectangle.
LODs	Table of pairs of LOD value and mesh name.	none	Describes the LOD progression to use. Ordered from highest to lowest detail meshes. LOD value is based on squadron rectangle's size in normalized screen size.

text

A text item is a text string that is rendered in the placement rectangle. It is not clipped to the placement rectangle, so the size of the placement rectangle usually does not matter. In fact, because text is always justify WRT the left/top of the placement rectangle, it's best to make the placement to have a min/max size of 0,0. The actual text is always procedural, and so you need a string parameter in order for the item to be visible. The following members are available:

Member	Type/range	Default	Description
colour	vector4 {0,0,0,0} < {1,1,1,1}	{1,1,1,1}	The colour to render the text. Not needed if you have a colour parameter is set.
renderFlags	table of the following: "justifyLeft"	true	Justify such that the left of the text lines up with the left of the placement rectangle.
	"justifyRight"	false	Justify text such that the right of the text lines up with the left of the placement rectangle.
	"justifyBottom"	true	Justify text such that the botton of the text lines up with the top of the placement rectangle.
	"justifyTop"	false	Justify text such that the top of the rectangle lines up with the top of the placement rectangle.
	"justifyHorizCentre"	false	Justify text such that horizontal centre lines up with the left of the placement rectangle.
	"justifyVertCentre"	false	Justify text such that the vertical centre lines up with the top of the placement rectangle.
LODs	Table of pairs of LOD value and font filename.	none	Describes the LOD progression to use. Ordered from largest to smallest font. LOD value is based on squadron rectangle's size in normalized screen size.

texture

A texture item is basically a 2D sprite. The following members are available:

Member	Type/range	Default	Description
renderFlags	Table of the following: "justifyLeft"	true	Default, no effect.
	"justifyRight"	false	Texture will be horizontally mirrored.
	"justifyTop"	true	Default, no effect.
	"justifyBottom"	false	Texture will be vertically mirrored.
LODs	Table of pairs of LOD value/image names.	none	Describes the LOD progression to use. Ordered from largest to smallest image. LOD value is based on squadron rectangle's size in normalized screen size.

Parameters

Parameters allow us to apply procedural modifications to ATI items. Text items, for instance, need a string parameter to be visible. Some items, such as the tacticalOverlay TO item, have no mesh and so need to have a mesh parameter. These parameters are indicated by a number which is an index into an array of parameters created by the caller. They are usually identified with a defintition in ATI.lua which corresponds to a #define in the source code.

Following is a list of the various parameter types:

Type	Description
floatParam	A parameter passed to rendering of a mesh with one of "clipLeft", "clipRight", "clipTop" or "clipBottom" set. This flag controls how much of the mesh will be clipped: 0 means all of the mesh will be clipped, 1 means none of the mesh will be clipped.
colourParam	This controls the colour an item will be rendered at. Overrides item colour or mesh colour.
stringParam	The actual text a text item renders.
scaleParam	The scale to render mesh items at. This is a vector3, but only x and z are used.
graphicParam	The actual graphic (mesh, text or texture) to render an item with.

Following is a table indicating how the various parameters are interpreted:

Type	mesh	text	texture
floatParam	Use with clipLeft	N/A	N/A
colourParam	Overrides mesh colour and item colour	Overrides item colour	N/A
stringParam	N/A	String to display	N/A
scaleParam	Controls scale of	Controls scale of	Controls scale of

	placement rectangle	placement rectangle	placement rectangle
graphicParam	Specifies mesh item to render	Specifies text item to render	Specifies texture item to render

Efficiency issues:

Rendering the ATI consists of rendering many small sets of polygons. This is nearly a worst-case scenario for modern video cards. To mitigate this, all ATI rendering goes through a system that buffers renders of like types (textures, wireframes etc) and renders them all at the end of the normal 2D game render.