

CT Vector

Introduction

In 1977 Game Designer's Workshop published the science fiction role playing game known as Traveller. As a young role player and inveterate fan of science fiction, it was only natural that I adopted the system of little black books (LBBs) as a favorite system for role playing adventures in deep space. The original game was laid out in three books. The first (LBB1) detailing player character development and combat mechanics, the second (LBB2) concerned itself with ship creation, mechanics and combat, and the third (LBB3) detailed worlds, systems, trade, etc. It is primarily the ship combat system detailed in LBB2 that this application was created to facilitate. Traveller presented a system of vector movement, plotted on paper, to represent the interactions of ships maneuvering in an interplanetary environment.

While I was intrigued by this approach for its realistic treatment of the space environment, it quickly became apparent that it was difficult to employ with ease due to the time involved in plotting vectors and the restraints of scale involved. Scale being 100kms per mm on paper, to draw Sol would still require a sheet of paper over 9 meters across! It also had an unfortunate tendency to change a roleplaying game into a ponderously slow wargame, which many gamers would have had little patience with. I ended up fudging the space combat stuff in my games. Future editions of Traveller largely abandoned this combat system, likely for the same reasons, but I always preferred the idea of the old system's approach in spite of its drawbacks.

The following years saw the advent of home computing, and it seemed a natural thing that this system could be adapted to a computer that could accept the input of navigation data and display the turn based results of that input in a more user friendly manner. Even so it was more than a decade before personal computing power reached the point where I deigned to make an abortive attempt at such a tool. My first attempts failed due to my own limitations in mathematics and programming, but even given those limitations, it was the inability of my chosen programming language (Qbasic) to handle the range of numbers generated that doomed that attempt. I'm not presently, nor was I then, of an inclination to learn entirely new programming languages. The project got shelved indefinitely.

Modern computing, and the face lift given to Qbasic by the folks at QB64.ORG, have now made it possible for me to advance the project into something that might actually be useful. CT Vector is the result, CT being a reference to "Classic Traveller".

Just what do we have here?

CT Vector brings a 3D cartesian coordinate system model of a local star system to your PC desktop. The system's primary star being the origin at (0, 0, 0). Coordinate resolution is 1 km and has a range that can easily accommodate large star systems. Ships may be placed anywhere within the loaded star system (*or even far outside of it*) and maneuvered through it. Any ship on the board can be designated as the "active" unit, and given navigational parameters which are then executed when committed to a "game turn". All other ships will display bearing and distance from the "active" unit, unless sensors are blocked by planetary bodies or extreme distance. Target locks may be established between combatants and/or lost from distance and planetary occlusions.

A number of visual aids are included, among them are dynamic zooming, jump diameter shading, range bands, 3D tilt, orientation and directional displays, scale grids and orbital tracks. These may be toggled on and off as needed to reduce display clutter.

The star system details are dynamic, with planets and satellites moving with each turn. Since I am not an astrophysicist by trade, the celestial bodies themselves do not move according to gravity; rather moving by rote data, but they will exert gravitational influence upon the ships in play as well as presenting potential dangers from impact. There is no orbital eccentricity and all planets follow the same ecliptic plane, but the ships themselves are not limited to that plane. You may take your battles to 3 dimensions if you wish. Because there are no gravitational mechanics for the planets, there are also no provisions for barycentric phenomena, though it may be possible for a clever GM to set up a system data file that would mimic something like that.

Star systems, when loaded, will query for a date reference which will determine the relative orbital positions of planetary bodies on that date. If player characters leave the system and return at some point in the future, the planet and satellite positions will have changed accordingly, adding a subtle layer of realism to game play...or so it is hoped...

CT Vector does not handle combat details other than range, position data and presence or absence of target locks. It is felt that such things as the firing of weapons and details of damage control are best left to the players, their character skills and their dice rolls. It is not envisioned as a game in itself, but rather as a game aid. It is designed primarily to get rid of paper, compasses and protractors as well as provide some additional visual feedback to given situations. How far is it to the jump point and how fast is that pirate vessel closing with you?

Basic work flow and file system

System loading: Appropriate star system files for the campaign/scenario should be prepared by the GM ahead of time and these should be stored in the “systems” subdirectory. The user may specify additional subdirectories within the “systems” folder, such as sector/subsector directories, but the user will be responsible for providing the path to those directories. It will be possible to enter the path and filename at the initial splash screen, or the program will default to Sol, which is provided for practice purposes. After entering the main program it will still be possible to load other systems, though only one star system can be loaded during a session. Star system files are identified their “.tss” extension.

Ship saving/loading: The program starts with a predetermined group of ships, which can be added to, edited or deleted as necessary. Once the desired details are entered/edited, the vessel group can be saved for future use, whereupon the group is saved in the “ships” subdirectory under a desired name with a “.tvlg” file extension.

Scenario saving/loading: Specific scenarios can be saved, and are placed by default in the “scenarios” subdirectory and recalled from there later. System, ship and other files are saved under a common scenario name in that directory and are not to be confused with system and ship files in the “system” and “ships” subdirectories as these scenario files save the exact scenario position and turn data for picking up where the players left off. CT Vector will also auto save to provide some crash protection.

Navigation: Each participant vessel requiring movement for the turn is chosen as active, whereupon it’s azimuth heading, angle of inclination and thrust can be entered. These inputs are not executed until a game turn is committed. Once a turn is initiated the ships move according to these navigation orders and any pre-existing vectors. The ships will continue to execute the same orders on subsequent turns unless they are altered. During this phase it is possible to add, edit, move and delete ships.

Turn: All ships move according to their navigation orders, all planets and satellites move according to their orbital periods, turn and time counters are updated. Display data is updated to reflect the new

positions, and certain old data is stored for vector computations or to undo a mistaken turn. Only one turn can be undone, there is not sufficient data storage for more than one undo operation.

Shall we begin?

When opening the program we are greeted by the title screen and queried to provide a system file. If the user knows the name and path of a specific file, it can be entered here, or pressing <enter> will default to a demo of the Sol system. After this, the program queries for a year and day input.



All system files stored in the “systems” directory are stored with randomized “zero year” planetary positions. Date input insures that the planetary ‘ephemeris’ is corrected from one visitation to another. This is not the same as a saved scenario, which is solely for picking up a game where it left off. Rather it is designed to give a degree of continuity to the game universe. It may be safely ignored, if not needed, by simply pressing <enter>, when queried for the date, to bypass the positional computations.

Astrogator

Once we are through the splash screen phase, the main display appears. This consists of several elements, a ship data display, orientation display, planetary distances and directions, controls block, Z-panning bar and an “**astrogator**” display. The astrogator, on the right half of the screen, is the visual centerpiece of the program and displays the relative positions of the units in play and the planets and satellites moving nearby. The active unit is displayed at the center of the astrogator and changing the active unit updates all other positions relative to it. The display may be zoomed in or out to include as little or as much of the star system displayed as desired. A default zoom factor of “1” is whatever field is necessary to include all units in play, making the maximum use of the astrogator display.

Zooms: Displayed below the lower left corner of the astrogator are the zoom controls and the present zoom factor. The user may either left click on the blue button fields to actuate the desired zooms or use the hotkeys “+”, “-”, or “x”. The Zoom Extents function “x” always returns to zoom factor 1. All planetary details will resize to the new scale when zooms are used. When moving an active ship via a right



mouse click, the zoom will reset to the default of 1, which may require additional zooming to bring in the desired display details and/or fine tuning ship positions.

Display toggles

Range: The range toggle displays combat range

information in the form of shaded ranging bands and circular targeting limits. It is toggled on and off via this button or hotkey “r”. These are based upon the LBB2 rules for combat ranges and sensor detection. This display is off by default.



Orbit: The orbit toggle is an on/off toggle that displays or hides the orbit tracks of planets and satellites that are present in the astrogator screen. The body need not be within the astrogator screen so long as it's orbit track is. The hotkey for this toggle is “o”. This display is on by default.

Grid: This, along with hotkey “g”, toggles the scale grid on and off. When present the grid dynamically resizes, and also rescales by a factor of 10, according to the zoom factor and display limits. When the grid is on, it's size is displayed in the lower left corner of the astrogator. This display is on by default.

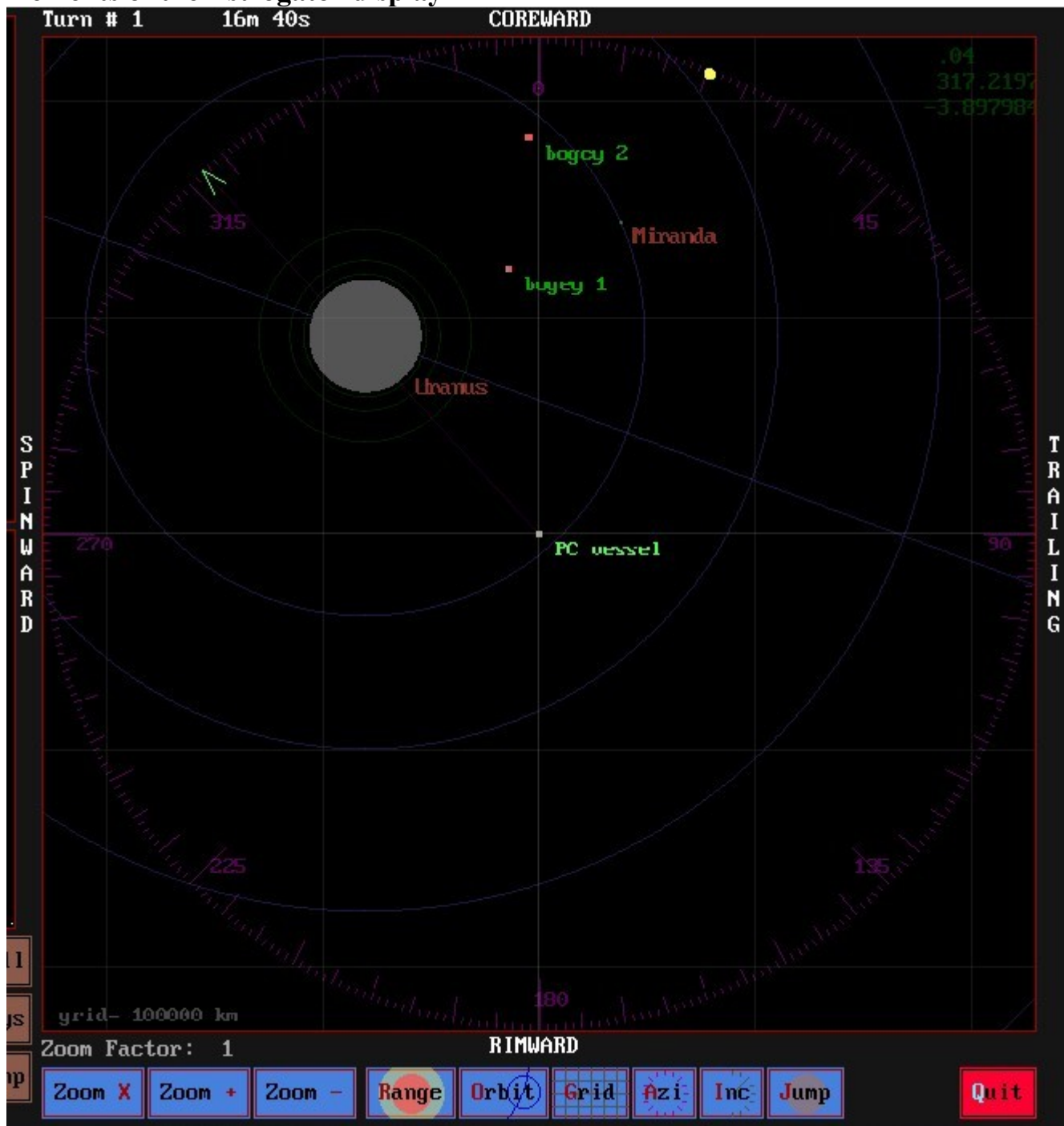
Azi: The azimuth display gives 360° “longitude” directions relative to the ecliptic plane. Toggled by hotkey “a”, it is oriented from 0° at Coreward, {*a Traveller canon direction indicating the galactic core*}, and ascends clockwise through Trailing (90°), Rimward (180°), Spinward (270°) and back to Coreward. When present, it also displays a small yellow circle which indicates the direction to the system's primary star. In addition, it displays the direction of travel for the active unit by a faint line extending from the active unit in the center to a green arrow situated upon the azimuth scale. This display is on by default.

Inc: The inclinometer display, toggled by “i”, gives a 90° “latitude” display divergent from the ecliptic plane. Its primary use being in showing the inclination from the ecliptic of the active unit. This display is off by default to reduce display clutter, and would only be used if 3D operation is desired.

Jump: The jump shader is toggled by “j”, and displays shaded spheres around gravitationally massive bodies at 10 and 100 diameters. This gives a visual reference for when a ship is at a sufficient distance from the body to safely enter jumpspace. When enabled, it displays an additional toggle adjacent to it, which allows the GM to choose jump zones based upon planetary diameters or planetary densities, which can alter the size of the zones for planets and stars of different density than the standard “earth” density of 1. Use of this would be dependent upon house rules to that effect. This display is off by default.

Quit: This ends program execution, saving a final autosave before exit.

Elements of the Astrogator display



Here we see the main astrogator display in its default toggle state. On the upper left it is indicated that this is the result of executing turn #1 and that 16 minutes, 40 seconds have elapsed. "PC Vessel" is presently the active unit, in bright green, located in the center of the screen and the faint line and arrow through Uranus (no snickering please) out to the azimuth scale reveals its heading to be approximately 317°. Other units, the "bogeys", are displayed in darker green. Additional information on all units is displayed in the data fields on the left side of the program display.

We see that the grid scale is presently 100,000kms per square at zoom factor 1. We know that the primary star is at approximately 20° as indicated by the yellow circle on the azimuth scale. We can also see the primary gravitational influence vector in the faint green numbers at the upper right, which display Gs, azimuth and inclination, respectively from top to bottom. This reveals the gravitational influence that Uranus is exerting upon the active unit, and explains the units present heading.

Ship Data Fields

Meanwhile, over on the upper left of the screen we have the ship data display. The active unit in this display is labeled so after the name, as well as being highlighted as white text on a grey background. All other units are shown as blue text on black.

All units display a unit number, name, absolute positions in X,Y & Z, speed, heading and inclination angle. Non-active units, unless occluded, also display a bearing, inclination and distance from the active unit.

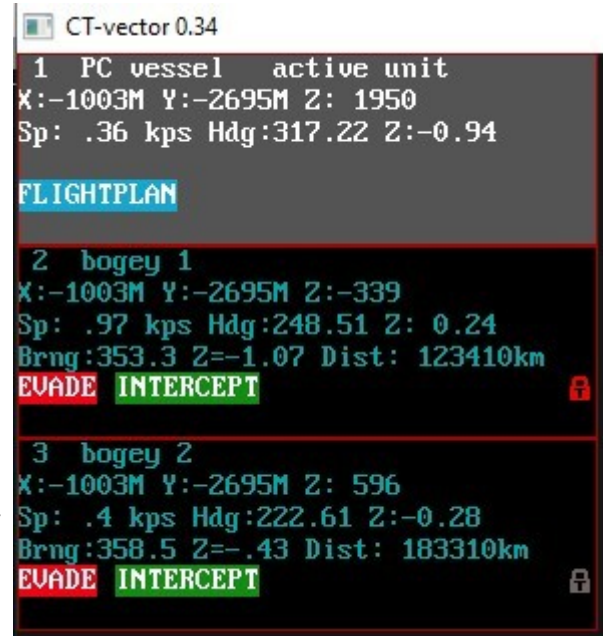
Target Lock: The small padlock “T” icon displays in either red or grey, red indicates that a unit is within target lock range, grey indicates that the range is too far to establish a target lock. The target lock range is dependent upon whether the ship is equipped with civilian ($\frac{1}{2}$ light second) or military (2 light seconds) grade sensors.

The GM may elect to click on the target lock if player(s) roll for, and establish a lock, whereupon the subject of the lock will display “target locked” in it’s data field and the unit point in the astrogator show in a red X box. We see here that “Bogey 1” is within target lock range, “Bogey 2” is too distant yet.

Once a lock is established, there are two ways it may be broken. One is for the distance between the vessels to exceed 3 light seconds (900,000kms), or for the sensors to be occluded by a planetary body.

Flightplan/Evade/Intercept: As of this writing, these routines have not been fully developed. I hope to have properly functioning algorithms in future releases. The idea of these controls is to introduce automated maneuvers to the active unit vessel. [FLIGHTPLAN] will make it possible to select a planetary destination, and all subsequent navigation commands will be automatically determined to move the ship toward that destination. [INTERCEPT], which is partially working, will set the active unit to seek to close range with the unit for which the “intercept” button was clicked. Similarly, [EVADE] will set the active unit to seek to avoid closing range with the unit for which the “evade” button was clicked

Choosing the active unit: There are several methods of choosing the active unit. The up and down arrow hotkeys will scroll through the data display, moving the active unit as it goes. Alternately, the user can left click on the astrogator near the desired unit, the non-active unit closest to the click point will be chosen, or the desired unit data field can be clicked anywhere except on the target lock icon or the “evade”, “intercept” or “cancel” buttons.



Operations in 3 dimensions

Even as space itself is an immense void with width, height and depth, Traveller was always envisioned as a 2 dimensional environment in order to simplify game mechanics. Both interstellar and interplanetary settings are represented as 2 dimensional phenomena within the game. In that spirit, CT Vector defaults to a view that is set perpendicular to the ecliptic ($z=0$) plane upon which all planets move, as though one were looking down on a sheet of paper on the gaming table. It displays the azimuth scale by default, but not the inclinometer scale. Nav command input requires heading and thrust, but inclination is easily bypassed, whereupon, all units will continue to move on the ecliptic plane. Gamemasters and players can easily ignore the presence of the 3D model.

However, the ease with which a third axis can be defined in a computer makes the capability a no brainer. The trick then becomes to represent a 3D environment on a flat screen. While there are numerous ways to do this, some of them quite complex, I have opted for a simple, single axis rotation scheme, which makes it possible visualize relative “Z” positions and spherical radii phenomena. It is here that we introduce the Z Panner shown on the right.

Similar to any scroll bar the Z Panner permits mouse click/drag selection of alternate views rotated around the X axis, as defined by the presently active unit. Other vessels and celestial bodies are rotated about that axis. The panner has a range of 180° , with 0° being parallel to the ecliptic plane, facing coreward; to the default 90° perpendicular to the ecliptic plane; to an inverted 180° , parallel to the ecliptic and facing rimward. Once an alternate view has been set in the Z Panner bar, a degree value is displayed in the bar and the bottom button will display “3D”, indicating that a 3 dimensional perspective is now active. No data is shown on the bar when the display is in 2D mode, as shown at right, and the button will display “2D”.

This bottom button will also toggle between 2D and 3D mode via a left mouseclick or the user may toggle modes with hotkey “3”. Toggling in this manner will alternate between the default overhead view and the last angle from the ecliptic plane that was set on the Z Panner.

Three gray blocks are shown in the bar, clicking on the middle defaults to overhead 2D display mode, and the upper and lower blocks set the view to 180° and 0° , respectively. Each of those viewing angles being parallel to the ecliptic plane. The 91° - 180° angle display is visually indistinguishable from 0° - 89° angles in many circumstances, so a rather unusual trope is used to indicate that the view is essentially “standing on one’s head”, which is to display the planetary names inverted. Weird, but I’m otherwise at a loss...

Ship placement in 3D mode: It should be noted that when a 3D viewpoint is selected any right mouseclick placement of the active unit will occur relative to the newly defined view plane and not the ecliptic. This means that placing a ship in 2D mode will not effect its Z coordinate, but in 3D mode the vessel will be relocated with both its Y and Z coordinates set according to the definition of the viewing plane and not the ecliptic plane. If a ship is placed while viewing at $-180^\circ/180^\circ$, then the ship will be placed with its Y coordinate unaffected. If playing solely in 2D and this effect is not desired, the user should insure that the display is in 2D mode before placing the vessel.

Active unit coordinates may also be adjusted via the Edit feature (hotkey “e”) if necessary to fine tune a position.



More to come....

Star System Worksheet for _____ Sector: _____ Subsector: _____

Rank	Name	Parent	-----Radius (kms)-----		-----Period-----		Density	Azimuth	Class	Notes/
			Planetary	Orbital	Orbital	Rotation	1=Earth	zero year	(stellar)	Zero year azimuth
				from	(years)	(days)		degrees	(Gas giant)	
				parent					(etc.)	

[illegible]

For use in arranging data for entry with SysInputII.exe