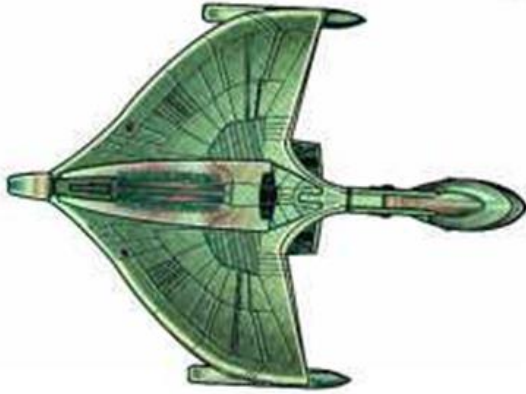


Your Ruber "solar system" and WarBird have been created. The WarBird's camera, missile sites, and intelligens-semite missiles are added to the simulation in this phase. Now it is time for the cadet (cadet) to move the WarBird, shoot, evade smart missiles, and navigate with gravity. Can the cadet pass flight training, or must they resign? The WarBirdSimulation.xls spreadsheet posted with phase 1



update #1 specification also describes the parameters of this assignment. Ship and missile speeds are vector forces to travel along the object's "looking at" direction (-Z). The "/sec" values show how far the ship or missile would travel in a second. "Missile frames" is the number of frames the missile is "alive" and its maximum distance (w/o gravity). The "distance" values describe the total distance traveled and the time in seconds to travel that distance. The "orbit" columns describe the angle of rotation for each target, and the time to make a complete orbit. The "cadet | TQ" columns show the time quantum values for each

cadet type. As the simulation's UPS increases, the play-difficulty of the simulation increases.

Please bring any errors in the simulations parameters to my attention! For any difference between this specification and the posted simulation spreadsheet, the spreadsheet "wins".

**Missile Sites.** Models for missile sites should be created. They should be easily distinguishable (say a box with different color than the planetoid). Missile sites should have a bounding diameter of about 30.

**Cameras.** The WarBird's camera is added to the other cameras from the first phase. When the current view, the WarBird camera shows the ship from behind looking over the ship.

**Ship Movement.** The ship's model is now part of the scene. The ship has thrusters that are on or off. We have no-inertial technology – WarBirds can suddenly halt if no thruster is on. (Live with it, it makes an easier assignment to debug.) The ship has positive and negative translation thrusters for it's "looking at" reference vector. The ship moves the current ship speed pixels **on** each frame when a translation thruster is on. There are three ship speeds: 10 (initial value), 50, 200. Ship speed is selected with the 's' keypress. The ship has rotation thrusters for it's "at", "up", "right" reference vectors. The ship rotates  $\pm 0.02$  radians each frame for a rotation thruster. Ship direction or rotation thrusters are set when an arrow key (or key combination) is pressed. Releasing a thruster key (key combination) turns it off. Holding the key down should "fire" many events. The "right" vector is oriented along positive X. The "up" vector is oriented along positive Y. The "looking at" reference vector is oriented along the negative Z. The ship can also "warp" to one of the two planetoid cameras as a result of the user pressing 'w'. After the ship has warped it can move freely. After warping the ship should be looking at the target, not moving, and see the target move along its orbit out of view. Pressing the 'w' key acts like the 'v' key – the ship warps to the next warp position mod warp positions.

**Missile Movement.** The intelligens-semite missile model should have a radius of 25. The ship has 10 missiles and each missile site has 5 missiles. A ship missile is fired by pressing the 'f' key. The missiles are smart. A ship missile's initial position is the ship's position and its "looking at" vector is the same as the ship's "looking at". For a Missile Site's missile its position is the Missile Site's position and its "looking at" vector is the Missile Sites "up" vector. Missile's intelligence tracking is activated after 50 frames. The missile does not detect for 49 frames (not smart,  $245 = 49 \text{ frames} * 5 \text{ speed}$ ). On each frame the missile is "live" it rotates an "orient-to" radians to face its target and makes a move forward. You can see when the missile becomes live, as this should be the largest rotation it makes. A ship missile's target is the nearest missile site within detection range. A missile site's target is the ship if it is within detection range. Why is it a dumb move to first a ship's missile when its target is behind the

ship? In the ship's camera view you can watch the missile's life by turning the ship towards the missile as it moves. The missile might "jump around" a little but should move towards the target (a nervous, but smart missile). The missile site's missiles are automatically fired when the ship is within a detection distance from the missile. A missile site and the ship can only have one active missile (currentMissile) at a time. When the missile is not targeting it does not test for collisions. Firing a ship's missile should not cause a collision with the ship on the first frame! All missiles detonate (disappear). It's their purpose! They detonate on collision, or when they have travelled all their frames. You do not have to simulate/visualize the explosion. If the WarBird has fired a missile, its missile display count is decremented. The cadet can't fire ('f') another missile until the current missile has detonated.

**TimeQuantum.** Your scene should have a "stepped" frame. The frame rate is based on a time quantum (TQ) pulse that is used to update object positions. The TQ should be set by user input and have at least 4 values: TQ very large (debug), TQ large (trainee), TQ average (pilot), TQ fast (ace). The game should start in ace (TQ fast). The TQ value (cadet type) can be selected by pressing the 't' key. Pressing the 't' key will sequence TQ selections from "ace" to "debug" and then back to "ace". For example, in initial speed (10 ) with a "pilot" TQ the ship would move forward 100 (1,000/100 frames/sec \* 10 / frame) units a second (1000 msec.) with no gravity. As in phase 1, the scene is drawn (displayed) as the `glutIdleFunc()`.

**Shaders.** Use the `simpleVertex.glsl` and `simpleFragment.glsl` shaders from phase 1. Trust me, phase 3 will make the models look much better!! Of course, you will write the shaders to do that...

**Pass or resign.** A cadet passes flight training by destroying both missile sites. Each ship, target, and missile has a bounding sphere for collision testing. If the ship collides with anything the cadet loses the game. If the cadet runs out of missiles without winning, they must resign from the war college. For the ship and missiles, the radius of the bounding sphere is 10 pixels greater than the radius of the object it bounds. All other objects use their radius (half the diameter) as a bounding sphere. If the cadet destroys both missile sites, the window title becomes "Cadet passes flight training". If the cadet resigns the window title becomes "Cadet resigns from War College".

**Changes to specifications.** You can make "compliant changes" to these specifications. A compliant change is one that corrects any error in the specifications, or makes the program more playable. A "non-compliant" change deviates from the intent of the assignment specifications. We can talk about changes you wish to make in lab before or during the review. You should document all changes in your "readme" report included with your final submission.

#### Cadet's actions.

Key	Action	Description
w	warp ship	select next planet warp % nPlanets
f	fire	launches available ship missile
g	gravity	toggle on / off
v	camera view	select next camera % nCameras
t	next TQ value	select next TQ % nTQs
s	next ship speed	select next speed % nSpeeds
↑	ship forward	positive step on "at" vector
↓	ship backward	negative step on "at" vector
→	ship yaws "left"	rotate - 0.02 radians on "up" vector
←	ship yaws "right"	rotate 0.02 radians on "up" vector
control ↑	ship pitches "down"	rotate 0.02 radians on "right" vector
control ↓	ship pitches "up"	rotate - 0.02 radians on "right" vector
control →	ship rolls "left"	rotate 0.02 radians on "at" vector
control ←	ship rolls "right"	rotate - 0.02 radians on "at" vector