“Paper Plane”

by John Kurlak

A humble clone attempt of Glider PRO (1991)

1. How to Play

The idea of the game is that you are a paper airplane. You must fly through a house, one room at a time, avoiding obstacles along the way. You progress left to right through the house. When you get to the last room in the house, you win.

When the game starts, you will see a title screen. You can bypass the title screen by clicking on it or by hitting the <SPACE> bar after the window has focus. You can bypass other title/status screens (i.e., the game over screen, the winner screen) throughout the game in the same way.

You can control your paper airplane with the arrow keys:

Left Arrow	=>	Move left (backwards)
Right Arrow	=>	Move right (forwards)
Down Arrow	=>	Move down

As you navigate the map, gravity will pull your airplane downwards. To stay afloat, you must move your plane so that it is above an upwards-facing vent. By default, the air coming out of a vent is visible. The length of the air stream indicates how far a vent will push you. You cannot move down when you are hovering over an upwards-facing vent. There is a checkbox option at the top of the screen to hide the visibility of the airflow from a vent. Un-ticking the checkbox will significantly increase the difficulty of game play.

You must avoid balloons, basketballs, cabinets, shelves, table tops (you can fly under a table), and the floor. If you hit any of these, you will lose a life. After you lose all of your lives, you will lose the game.

You may collect clocks (give you extra points), folded sheets of paper (give you an extra life), or batteries (give you “battery” power).

Battery power allows you to fly faster for a short period of time. To use your battery power, hold down <ENTER> while flying left or right.

You will win the game when you fly through the window in the last room.

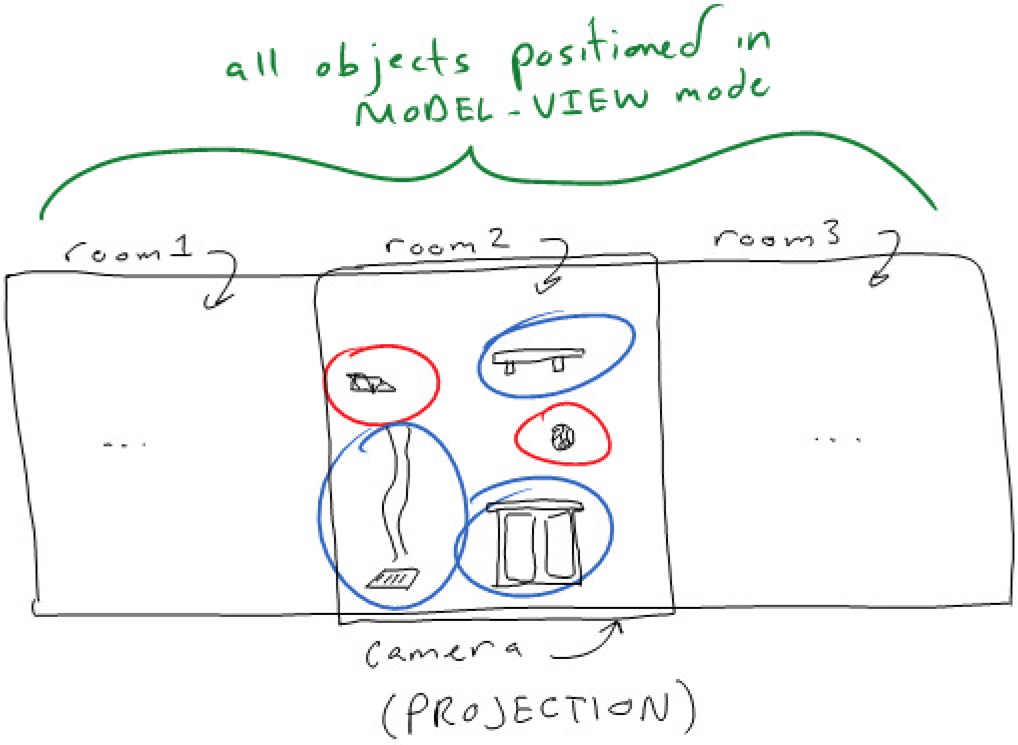
You will quickly realize that I made the rooms very difficult to complete. To me, this makes the game addictive. If you do not feel the same way, you may bypass a level by holding <J> + <K> on the keyboard (by entering my initials, you admit defeat!). However, I have tested each level and can verify that each is do-able.

2. Architecture

Every object that is static (objects in the room that do not move or cannot be picked up) is saved to a display list the first time it is drawn. Then, every successive draw makes the draw call on the display list. Items that can be picked up (clocks, folded sheets of paper, and batteries) also have a display list. When the item is picked up, I queue that item to update its display list. Non-static objects, like balloons, basketballs, and the paper airplane do not have a display list.

I dabbled with the idea of having a giant display list for all of the static objects. At one point in the development process, I implemented that idea. However, it turned out making my game a lot slow, so I removed it. I know that it was a requirement to have this in our project, but I would rather turn in a fast game than one that follows the specification exactly to a tee. If you have to take off points for this, I understand.

The display lists are called from the MODEL_VIEW mode. I simply loop through all of my game sprites and render the corresponding display list for each. I only do this for objects belonging to the current room visible on the screen. This allows me to add any number of rooms with an extremely small hit to performance. I use the PROJECTION mode to move the camera to each successive room as the game progresses. Any time I move the camera via PROJECTION mode, I also have to translate the glider and the heads-up display to ensure that they stay on the screen.



Dynamic: No display list

Static: Each has own display list

Each object has rotation, scale, opacity, bounds, hit bounds, etc...

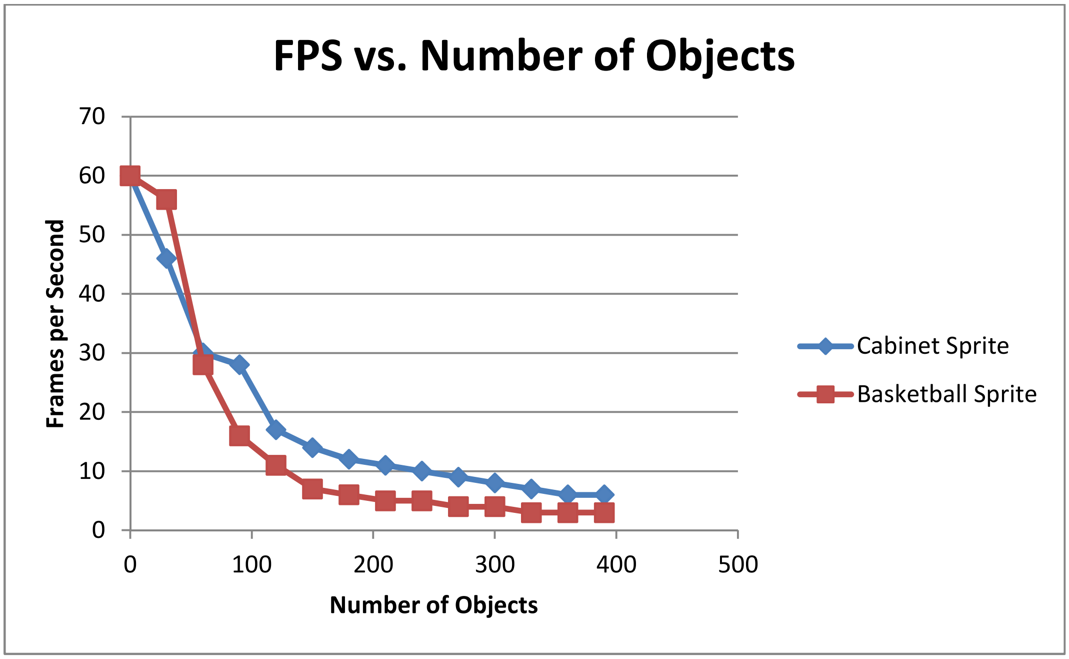
3. Performance Analysis

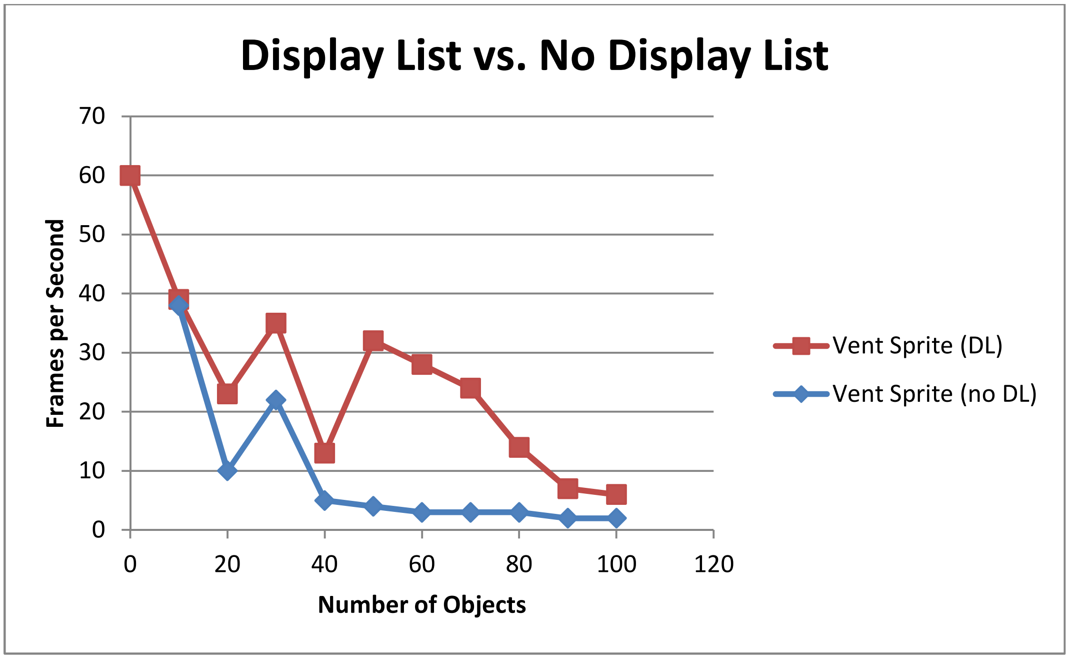
Overall, the game seems pretty scalable. I set it up so that drawing, movement, and other calculations for rooms do not take place unless the room is visible under the camera. This allows me to add a large number of rooms with little to no negative effects. I added up to about 5000 rooms with 7 objects in each with no negative effects on performance. After about 5000 rooms, I was unable to get the game to load (the memory usage was close to 2 GB).

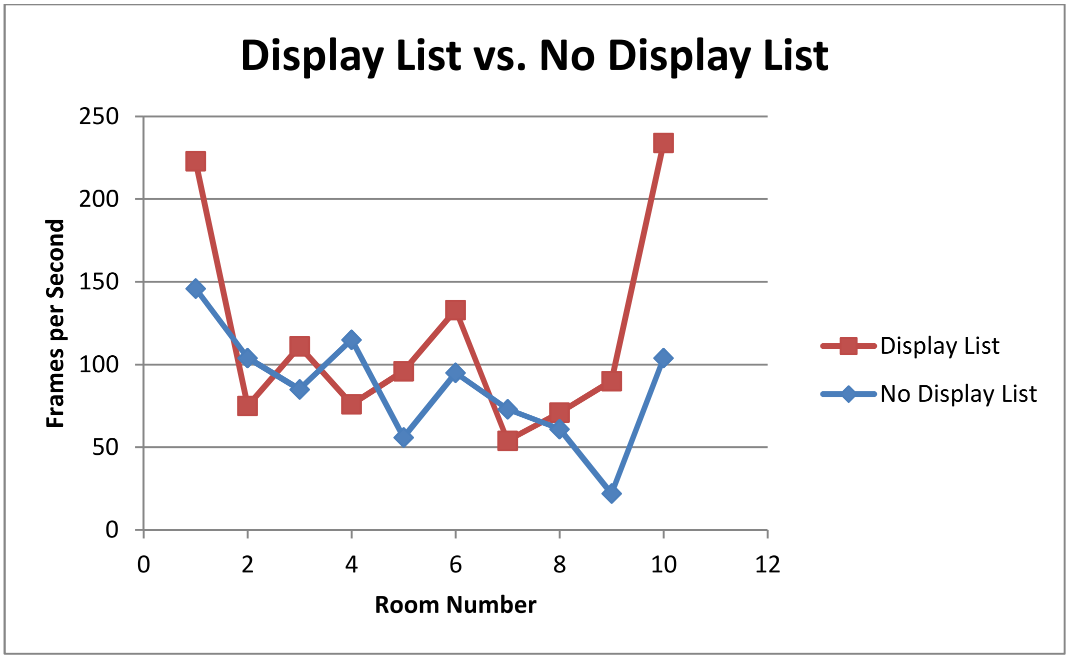
My next test was to see how many objects I could stuff into a room without a negative effect on performance. I knew this would vary with the type of object (some require more computation than others), so I tested with objects on opposite ends of the spectrum: a cabinet (computationally easy) and the basketball (computationally intensive).

I then did frame rate comparisons based on whether display lists were being used or not. I first tested the effect of display lists on computationally intensive objects (vents). Then, I tested the effect of display lists on each room.

The graphs below summarize some of my findings:







There appears to be an inverse square relationship between the number of objects and the frame rate. It also appears as if using display lists proved to be worthwhile.