

Cloudscape: Aerial Strategic Racing

Kenneth Grauel



Figure 1. A few venerable games in this genre.

Executive Summary: High-speed aerial hovercraft racing. Ridiculously fast futuristic vehicles in a frenetic nitro-fueled sprint for the finish line. Narrow, dangerous tracks constantly threaten vehicles with suicidal freefall.

Overview:

Cloudscape is a **single-player** racing game with **AI opponents**. Vehicles are essentially **hovercraft** with extremely fast engines. Generally speaking, the hovercraft never touch the track, but if they do, it's over. Tracks are twisty, usually downhill, one-way, suspended in mid-air, and **riddled with obstacles**. Certain death awaits those who become careless at high speed. Gameplay modes include **practice**, **time trial**, and **championship**, with comprehensive **high scores** and **ghost replay** so players can race against recordings of their best times.

Why Cloudscape?

- Because racing games are **awesome**.
- Because dangerous, mid-air racing games where vehicles can push each other off the track to certain doom are **really, really awesome**.
- Because it offers an excellent environment for **high-end graphics wizardry**.
- Because it's easy to do **impressive physics** with objects that explode on contact with the track.
- Because even if you hate graphics or physics, there's still **plenty left to work on**: sound effects, level design, texturing, user interface, vehicles...

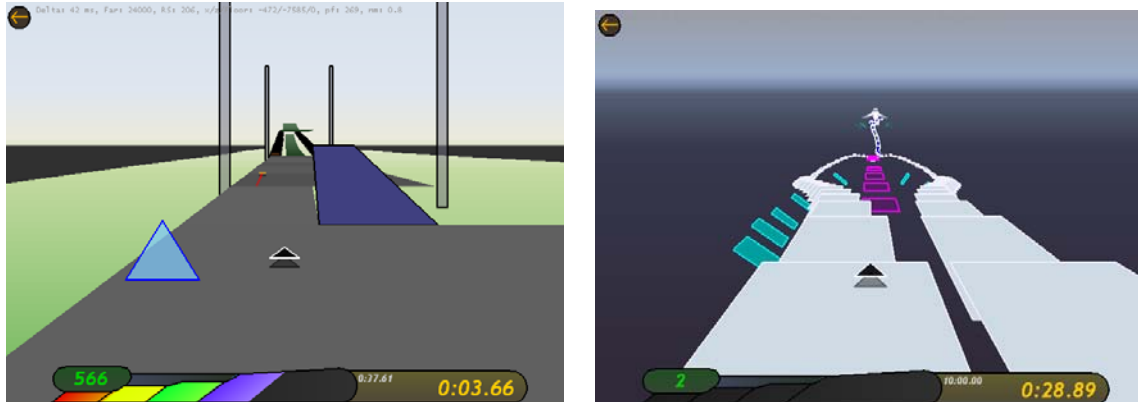


Figure 2. A *very* early predecessor to Cloudscape, coded in Flash.

Gameplay Features:

- Vehicles equipped with **turbo boost**, enough to last several seconds. Vehicles may pick up **turbo refills** around the level in hard-to-reach spots.
- Vehicles can also **jump**, which uses a small amount of turbo.
- Falling vehicles are placed back on the track at the previous **track checkpoint**.
- Various **special zones** on the track such as turbo, slowdown, insta-death, upward fan, and downward fan.
- **Moving hazards** present mobile obstacles for particularly advanced players.
- **Attacker bonus** for players who knock opponents off the track.

Feasibility:

Take a look at the screenshots above in Figure 2. This is a game similar to Cloudscape that I coded about a year ago in Flash, which has no 3D engine or hardware graphics and interprets code several orders of magnitude slower than a comparable C program. Even though the entire 3D engine, including visibility, had to be coded from scratch, the program clocked in at under 2000 lines of code and took about 2 weeks of part time work.

Basically, although Cloudscape is a great deal more complex, 3 months multiplied by several coders should do the trick.

Technical Details:

Below is a rough outline of implementation. However, this whole section is *up for discussion*, except for perhaps using C/C++. I'm sure my fellow team members will be able to improve on these ideas.

Pre-Alpha: Basic graphics display on screen using **OpenGL** or **Ogre**. World geometry and user input are both essentially finished, although lighting and texturing are not expected.

Alpha: Roughly playable. Physics system and graphics infrastructure are basically complete. At least one coder is making progress on UI, sound, and graphics content (which, if we lack a good artist, will be created *procedurally*).

Beta: All game features are in place, including UI and sound. At least one track is essentially complete, allowing for a convincing demo. Work begins on AI opponents, additional level content, and other non-essential features as time permits.

Final: Gameplay is complete.

Some additional notes for particularly masochistic readers:

Graphics Content: The design of the game engine may seem trivial compared to creating attractive game content, but there are several decent shortcuts for even terrible artists. Most of the racetrack proper can be generated procedurally by tracing a changing 2-dimensional cross-section across a moving, rotating line. Skies and backgrounds are easily accomplished with billboards, fog, a horizon line, and some sparse ground objects or a terrain mesh.

Physics: There are several open-source libraries that implement good physics, but it may make more sense to implement a custom solution. I suggest storing a simplified world geometry in an octree with a reasonably large minimum bucket size. We can then view the hovercraft as a sphere of small radius which attempts to maintain a certain distance from surrounding obstacles. Implementation involves casting a reasonable number of rays outward from the vehicle in all directions and attaching springs to any nearby surface.

Engine Sound: Engine sounds can be efficiently synthesized from either pre-recorded looping samples or from scratch for a more futuristic sound.