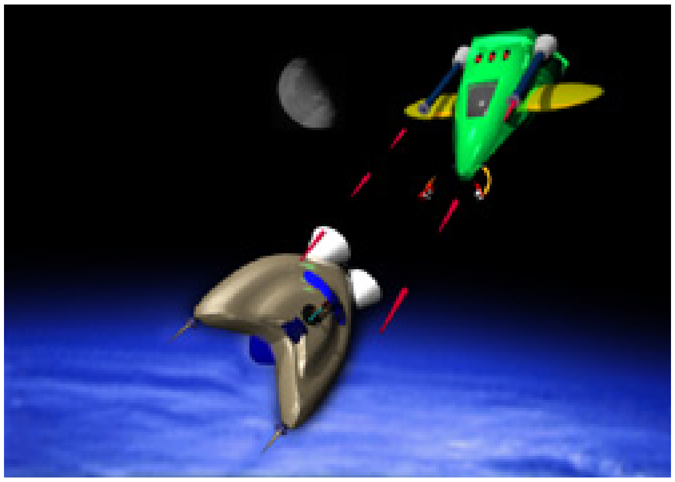
Crossroads
The ACM Student Magazine

The Story of *XPilot*

by Bjørn Stabell and Ken Ronny Schouten

Setting: The northernmost university of the world, the **University of Tromsø** in Norway, early fall 1991. We were undergraduate students in computer science, and we needed something to throw our new-found wisdom at, something that wouldn't be just a throw-away project. We had a vision of a game similar to the ancient Commodore 64 game *Thrust* in spirit, but this was to be for multiple players and run on all UNIX workstations with the X Window System. The result was *XPilot*, which is one of the most popular Internet games existing today.

XPilot in brief



A short explanation of the game play is in order.

You guide a triangle shaped space craft through a two-dimensional cave-like environment. Your vehicle has a very simple steering system: a rear mounted engine provides forward propulsion, while two thrusters (one mounted on each side) are used to rotate. The space you fly through has no resistance, but plenty of gravity. Learning to fly is one of the most challenging aspects of *XPilot*.

But in *XPilot*, as opposed to games like *Thrust* and *Asteriods*, you're not alone. The real fun begins when you fly around with friends. By configuring over one hundred game parameters, you can set up a number of playing modes. For example:

- race through a course between the cave walls,
- have dog-fights with machine guns as the only weapons, so it's all up to your flying and aiming skills,
- play together in teams and try to destroy other teams' treasures or steal their valuables (a ball you have to tow back to your team's home base),
- have a full-out nuclear war with your ``friends'', where strategy and the size of your weapons arsenal means everything.

In a typical *XPilot* game you have to fight to stay alive and not be drained of energy. You have to avoid being hit by weapons fire from the deadly robot players, other human players, and cannons scattered around the world. At the same time, try to set traps for (e.g., plant mines at an enemy home base) and shoot down enemies, search for new weapons, and refuel from fuel stations.

You have extreme configurability in *XPilot*: you can change the game parameters, the cave you fly in, or the shape of your space-craft. All of this makes for varied game play, and that perhaps explains how people can be *XPilot* addicts for years.

Motivation

So, why did we do it? It was probably a combination of us wanting to create something that both we and others would actually use, and hopefully learn something we couldn't learn from assignments. And we could both clearly see the potential in such a game, so we believed in it. No one had ever done something like this with the X Window System, but with UNIX workstations becoming more powerful (the Computer Science Department at the University of Tromsø had a fabulous park of HP9000/720 computers they provided for undergraduate students) we thought the world was ready for an action packed game for UNIX and X.

We were very enthusiastic about the project from the start, perhaps even a bit naïve about the size of our undertaking. What kept us going, though, was the constant positive feedback from our users. It turned out that people were appreciating our creation and so we worked in small steps, implementing feature by feature, so that we always had tangible improvements.

The first steps: learning to fly

We dived into code hacking, aided by local experts. Initially planning to use Motif and Xt, we soon tossed those out for reasons of portability and fear of complexity, and we ended up with a just ANSI C and Xlib. That was an important choice, enabling *XPilot* to become more widely used as well as providing sufficient support for the functionality we needed for the game.

After about a month of intense hacking, *XPilot* consisted of a single process game that drew on multiple X displays, and a small handshake program that users ran to contact the game process and join the game. The most basic elements of the game were there already: players could join and leave the game any time, the movement model (gravity and steering) and collision detection were almost like today's version, and you could see basic visual elements like the triangle ship, weapons fire, and the background walls.

One of the hassles we had to deal with early on was the flickering that occurs when you clear the screen and redraw the game graphics. We needed double buffering. Multi/double-buffering extensions to the X server were not available at that time, but we found an article in **Dr. Dobbs Journal** explaining how to do double buffering by drawing in half of the color planes at a time, and only showing the half that you're finished drawing in. This is very similar to how double buffering is implemented in hardware. The trouble was that with only half of the color planes available, the number of colors available was reduced from 256 to 16 colors, so *XPilot* had to be very economic in its color use. Other double buffering methods were added later, but the color planes method is still the fastest of the software-only methods.

Now *XPilot* was beginning to be playable, and it started to catch on at the undergraduate lab of the University of Tromsø. It soon became so popular that the technical staff began to view it with more critical eyes. They were concerned about the hard play damaging the keyboards, and that the loud noises that inevitably come from *XPilot* players would disturb other students. Besides, the lab suddenly became so popular that it was getting increasingly hard to find a free computer. We had to do something to stop *XPilot* from being banned altogether, and we quickly added some extra tests to ensure that players would behave by preventing them from playing during work hours or when the lab was almost full. The executive of this heuristic was a script aptly named *killjoy*. Much later, auto-repeat on the fire key (less known as "Return") was added to stop the keyboard hammering, but by that time you could tell which Return keys had been used as fire buttons.

Contrary to many other applications, it wasn't difficult to find ``testers'' for *XPilot*. We received many bug reports and suggestions for new features from people in the undergraduate lab. Creativity was everywhere, and *XPilot* soon included features like shields and the memorable sparks. The sparks from explosions and engines are interesting because they are implemented as real objects in *XPilot*, as non-lethal colored bullets. With sparks as real objects, and Newton's third law implemented, the basis was laid for physical phenomena like shock waves from explosions and engine bursts, as well as recoil from the guns. The sparks make a huge difference in the realism of the game dynamics. Besides, they are a really cool effect, especially when they, much later, were upgraded to fade in color based on their age. This feature is somewhat similar to the particle system animations used in, e.g., the Genesis effect in Star Trek III.

The fall 1991 semester was closing and *XPilot* was an epidemic everyone at cs.uit.no was noticing. Strangely, we still had time for the lectures and courses; it seemed like the inspiration we got from *XPilot* had somehow affected our computer science studies as well, in a positive way. Four months had passed since we first thought of *XPilot*.

A game for the Internet

Next semester, Ken Ronny started on his one-year long mandatory military service, while Bjørn continued working on *XPilot*. In May, Bjørn released the first public version of *XPilot*, version 1.0, and was swamped with mail from people all around the world. Over 200 emails arrived the first week, and Bjørn spent the whole wrist-aching week answering them all.

It turned that out a lot of people were using Suns, and we hadn't tested *XPilot* on Suns, so a lot of the problems reported were the same: people on old Suns weren't using ANSI C compilers, and got incomprehensible error messages. The usefulness of a **Frequently Asked Questions (FAQ) for *XPilot*** was suddenly very apparent. People just didn't read, or didn't understand, the installation instructions.

Feedback came from everywhere: Japan, the USA, Europe, from students at universities, and employees of **NASA**, **Intel**, and **NCD**. People sent comments, suggestions, and, most importantly, patches to the game. A memorable ``bug fix'' came from some guys at NASA; it turned off gravity. They couldn't understand why someone would actually want gravity, as their business would be so much simpler without it.

A game by the Internet

One of the great advantages of developing software for the Internet is that you'll have thousands of competent users providing feedback. Often this feedback is in the form of requests to change the source code in a certain way, in order to add another feature or fix a bug. Over the years, tools have been developed to alleviate the problems of communicating these change requests in an efficient and unambiguous way. Two such tools are **diff** and **patch**. The former extracts the differences between two versions of some source code, and the latter takes these differences (called a patch file) and incorporates them into another version of the source code.

This was the form we encouraged people to submit their change requests in, and this was they way they did it. It is the Internet way.

As time went on, the really large patches started pouring in, and it became increasingly difficult to automatically incorporate the patches into our original source code; manual labor had to be used to resolve the cases where two people had changed the code in approximately the same place. We started to use revision control (we still use CVS for this) to manage the different versions in circulation, but we couldn't get away from most of the very monotonous work of resolving such conflicts. When some huge patches dropped in our mailbox, inspiration dropped. Then people started releasing their own versions of *XPilot* (version 1.3something, while the latest version we released was 1.2) and we were increasingly bothered by bug-reports for versions we didn't release. Clearly, some management and guidelines were needed, and those were put in place: people were urged not to release custom-made versions of *XPilot* as this back-fired on us.

In the spring semester of 1993, Ken Ronny was back from the military, and were a two-strong group again. Getting our act together, we released *XPilot* version 2.0 to get around the version chaos of the illegal version 1.3.

Version 2.0 incorporated a lot of features received from people on the Internet, as well as our own development efforts. The code had grown considerably, and we had to be more selective in which patches to accept in order to stop the code from becoming unmanageable. Software entropy in action.

All these new features really made a difference; robot players made single player games possible, bouncing helped newcomers, new weapons and more configuration options encouraged variation in games.

The problem was that we received so many new weapon system patches. With each weapon bound to a key on the keyboard, there soon weren't many keys left to use. Besides, it was becoming increasingly difficult to master all the aspects of the game, and we were afraid newcomers would be turned off by the complexity of the game controls.

It turned out that many other people also felt that all the new features made the game more confusing, and in many cases didn't add that much to the game. The trend was to return to the simplicity of the good old dog-fights, as compared to the full featured games which incorporated, e.g., radar guided or heat seeking fire-and-forget missiles with conventional or nuclear war heads, cluster bombs, cluster mines, electronic counter measures, lasers, stun rays, autopilots, and emergency shields.

We tend to agree with this trend. What makes *XPilot* special is the feeling of realism and control you have while maneuvering the ship. Learning to fly is the real skill. Still, we managed to keep just about everything configurable in the game so that we could promote variation and satisfy different tastes.

The Flying Dutchman

During 1992-1993, a Dutchman named Bert Gÿsbers sent us more and more patches. In spring 1993, he embarked on something we didn't have the motivation to embark on, but it is perhaps the single most important factor for the game's success as an Internet game. He split the game into two processes, a client and a server. Previously, *XPilot* had relied on the network transparency of X to achieve multi-host play. With a real client-server architecture in *XPilot*, the stage was set for a much more efficient protocol, and drastic improvements in playability over longer distances.

With the old architecture (a server drawing on multiple displays), we could barely play from Tromsø to Oslo (a distance of about 1500km), but with the new client-server architecture, games between Tromsø and Amsterdam were quite playable. Cross-Atlantic games were tested too, and though the frame-rate (and packet loss rate) could be quite good, the latency was just too great for this fast packed action game.

Bert's contribution was so significant that we happily included him in the author list. He's an incredibly able coder and person, and he's done an enormous amount of work on *XPilot* ever since. Today, he is the one maintaining *XPilot*, as well as doing further work.

The Meta server

A client-server architecture ploughed the way for more long distance Internet play. Now the problem people had was: ``where do I find a server?'' The meta server solves that question.

A Meta server is a central program that receives information from *XPilot* servers on the Internet. This information can then be accessed interactively on the Internet via telnet to **meta.xpilot.org port 4400**, but recent interfaces in Java and Tcl/Tk are more easy to use. A lot of people use this service to find servers to play on.

In addition to being a passive registry of running *XPilot* servers, the meta server can act as a watchdog, sending messages back to servers that are running old and defunct versions of the game. This was a very effective way of making people aware that they should upgrade. Without the Internet, this would be impossible.

Through the meta server, we also get a rather good indication of the number of servers running in the world; currently the list usually contain around 70-90 servers.

Publicity

XPilot didn't go unnoticed in the world, and we were more or less taken aback when it was embraced and included in the OS distribution of Hewlett-Packard, and later on Linux CD-ROM distributions and a multimedia showcase CD-ROM from Digital.

XPilot soon also made an appearance on USENET, the World-Wide Web, and of course as a mailing list and a FTP archive. Most services related to *XPilot* are now collected under the **xpilot.org** domain.

At long last, it also appeared in the printed media, e.g., in the third issue of **.net**, and in the book **rec.g@ames**, which devoted a whole chapter to the game. Still, we nostalgically remember the first time *XPilot* appeared in ink, in an interview by a fellow student for the local computer science society's newsletter.

In the curtains now are an appearance on one of the major Norwegian TV stations.

Not just bits and bytes

XPilot has also been a bridge between a lot of people from a lot of places, stimulating contact between players and coders. Friendships have been formed, people have gone on vacations together, and some even grew into much more than friendships, all just because they initially shared a common interest in *XPilot*.

As a direct result of such a good community, the *XPilot* players have the **Newbie Manual**, created by **Karen Gould** and **Erwin Zieler**. They received heaps of help from many players around the world, but we can not thank them enough for their enormous effort.

Over the years, we've received numerous comments and stories from players; some are congratulations from new players, some are sad stories of what *XPilot* addiction can lead to, and some are just pure fun. One we remember rather well came from a person that was playing during work hours. When his boss came in the door; because he was an experienced player, he very skillfully managed to land the ship, put the game in pause, and iconify the window before his boss could see his screen. Relieved that his excellent skills had saved him, he talked with his boss for a while; until a new round of play started on the server he was connected to. We had just added a new feature which pops up the *XPilot* client's window each new round, and you can imagine his joy when *XPilot* treacherously filled his screen again, while his boss was looking over his shoulder. This just shows that you have to do usability analysis. Now the client window won't de-iconify when the player is in paused mode because the poor fellow's arguments were just too convincing.

In the fall of 1994, the **European XPilot Team Cup (EXTC)** was held. 18 teams of 4 people each joined to compete for the European Champion title. During the summer a winner was declared, but it wasn't us. Strange how times had changed.

What we learned

Needless to say, we learned a lot from working on *XPilot* as compared to a typical computer science assignment. In particular, several factors were magnitudes larger:

- the time-span of the project

- the number of developers and amount of feedback from them
- the number of customers/users/players
- the size of the software
- the number of platforms supported
- the enthusiasm and motivation (the fun factor)
- the publicity.

It was more useful and fun, while still being at least as challenging as many assignments. We were very fortunate to have had this learning experience, and we wish more people would have it. We feel we've played our way to knowledge, and that's made us more motivated to learn; by creating meaningful assignments, and encouraging students to have and to follow their visions even while at the undergraduate level, we're sure educational institutions could get more people interested in computer science.

In general, we learned:

- You can do it, so have faith in yourself. Live your visions; at least some of them.
- ``Pet projects'', like *XPilot* are an excellent addition to your computer science education that industry appreciates, and a valuable personal learning experience.
- With the Internet, you can really play the role of a vendor or service provider, virtually without any costs except your own time. This is a completely new possibility which we believe is far underestimated by educational institutions.
- By making something that is appreciated by others, you're sure the feedback will keep you motivated to continue. People do care, and you can make a difference.

In addition, of course, we learned many computer science specific lessons.

The law of increasing software entropy is visible in *XPilot*. The code is still fully manageable, and we haven't noticed an increase in the number of bugs, but it's getting increasingly hard to add significantly new things to the game. It's easy enough to add new weapons systems, but not the kind of features we'd like to see, like the following.

- Dead-reckoning is a technique used in, e.g., Distributed Interactive Simulations (DIS), which cuts down latency and network bandwidth at the expense of accuracy in the movement model. In plain English, this means that with dead-reckoning it would be possible to play *XPilot* over much larger distances and lower bandwidth (like a 28.8kbps modem line), but still keep a very high frame-rate.

- Support for more platforms have proven difficult, and up to this point, only UNIX variants with X have been supported. We know of two Amiga ports that were started, but none ever finished. Having support for more platforms (like OpenGL, MS Windows, Macintosh, and AmigaOS) would significantly increase the number of players.
- Cooperating servers could enable players to fly from one map to another.
- Better user interface, and integration of existing *XPilot* tools could make the game more user friendly.

In hindsight, perhaps we should have stopped at one point in time when these issues were becoming apparent, and reconsidered the design, or lack of design. We could have treated the game we had as a prototype, and started all over again. Now, we're stuck with the design decisions that were more or less taken implicitly. We had no formal design, we just coded the game in an exploratory fashion. It's really hard to say a more rigorous and initially bold design would've been beneficial, as it might have impaired the fun of doing it all, and thus eliminated our main motivation.

Some things we definitely would have done differently today would be to throw away C in favor of an object oriented language like C++ or Java. Even in C, we programmed in a somewhat object-oriented fashion; player objects (Player) inherited from movable objects (Object). The inheritance was done in an incredibly hackish way, though - see the source code for the gory details. Using an object-oriented language, we would have been able to use object-oriented techniques more elegantly so that source code complexity was reduced and flexibility increased. The code would have less entropy.

We would also design the game in a more modular fashion. X proved capable of supporting high speed action games, so it wasn't a bad choice. But, we didn't design a clear interface to the window system specific code, so the game was very bound to X.

The game is very useful even in its present state; we just can't make it do **everything**.

Future Plans

So far the only operating systems to run XPilot have been the UNIX flavors, and this has most certainly limited the number of players. As this article is being written, there's a hot debate on the *XPilot* developers' mailing list concerning a Windows NT port of the game. When this is printed, you might have heard of it already, and the guy doing it might have released a Windows NT version. We expect this port to generate a lot of new players.

One problem we discovered when we moved out of the university was that we'd like the *XPilot* services to have persistent addresses, and still be under our control. To facilitate this, we have registered our own Internet domain name, **xpilot.org**, in which people will always find the typical *XPilot* Internet services like the meta server, the FTP archive, the WWW home page, our mailing lists, source code repository (CVS), and perhaps even some game servers.

It is always a fight to keep motivation high. We worked on *XPilot* in our spare time, and as long as it was fun we did a lot of work. But when the project grew in size, we found ourselves involved in more and more administrative affairs, and less in creative development. Also, the code was getting increasingly unmanageable because of the size; it outgrew its sparse design. Consequently, we weren't as motivated to work on it anymore.

We have enough ideas for a lifetime's work in *XPilot* but we just won't and can't spend a lifetime on it, not for free. The game has meant so much to us during these years. It was great fun, a big learning opportunity, a door opener, a thundering headache, and a source of great satisfaction. We don't want to see it die just because we don't have the energy to keep it alive. So hopefully, with the help from the Internet community, it will stay alive as one of the best, and few free games on the Internet.

Acknowledgments

The authors wish to thank all the souls on the Internet that made XPilot what it is today. In particular, of course, we have to mention Bert Gÿsbers, who is Mr. XPilot these days.

Bjorn Stabell is working at SINTEF Applied Mathematics doing scientific

visualization and miscellaneous system and network administrator stuff. Previously, he has worked at Kongsberg Spacetec a.s., and the Computer Centre and the Computer Science Department at the University of Tromsø in **Norway**. He also studied at the University of Tromsø, obtaining the degree of MSc in Informatics.