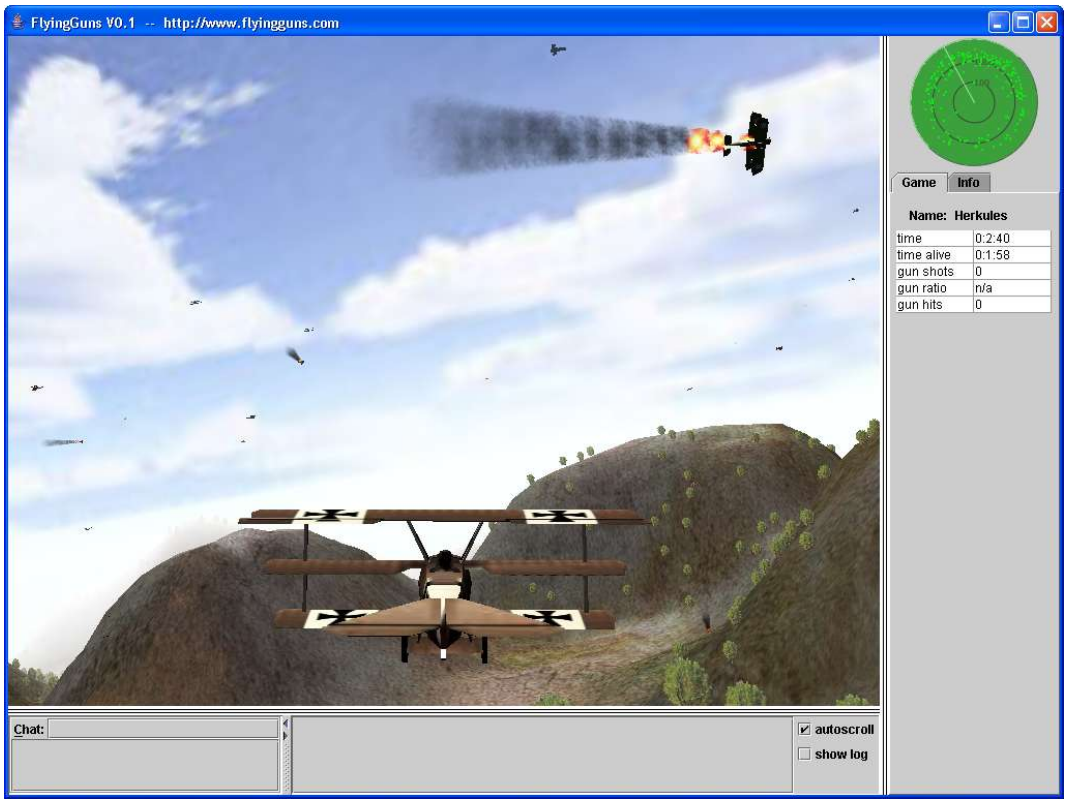
FlyingGuns

2004 Java[tm] Technology Game Contest

www.flyingguns.com



Credits

Joerg 'Herkules' Plewe	idea, design, architecture, network (HeadQuarter), 3D coding
Stefan 'McPfaff' Pfafferott	3D coding, terrain, FX, AI, artwork
Miriam 'Miri' Bette	3D artwork
Marco 'Slick' Giertolla	3D artwork

Special thanks to:

Rainer Foetzki, Fuad 'Sandman' Abdallah, Thomas 'Maverick' Haeuser, Andreas 'Snoopy' Nitsche

About

The FlyingGuns project was started to study and develop certain areas of technology on a cool example. Thus FlyingGuns is a technology driven game. 'Form follows function' ... the game is wrapped around the technical possibilities and the desired, networked environment.
The project does not try to imitate mainstream games just with Java technology, but defines its own goals derived from unique features of the Java environment.
FlyingGuns is developed as part of a sparetime project. This fact had significant impact on the choice of the language, 3D engine and gaming goals.

The game and the underlying technologies are opensource as part of the SourceForge-hosted 'Distributed RealTime Simulation' project DRTS.

Goals

Many people play solitaire in the lunch break at work. We want these people to be able to launch FlyingGuns instead and play against their collegues. Or find someone else to play with on the internet. Without the need to install hundreds of megabytes like mainstream games today.
So the game needs to be a single- and multiplayer game that works on the internet as well as on a LAN. It must be able to operate on a low volume of resources (artwork, sound) in order to be easily downloadable. Gameplay has to be casual, getting fast in and fast out of a game.
To be easily installed, only common technologies can to be used. FlyingGuns relies on a JRE and Java3D to be installed. No 3rd party Java or native libraries are necessary to operate the game.
Low resource volumes, anticipation of poor hardware and the restriction to basic technologies are three topics that fit together very well and form the character of FlyingGuns.

One of the main Java benefits come from the easy deployment. FlyingGuns is meant to be incrementally deployed and kept interesting through new features constantly.

Technology

FlyingGuns is build on top of a framework of technology components.

3D

For rendering, the Java3D extension is used. It offers decent performance but a good set of tools and utilities and can be installed from the same source as the JRE, so it may be encountered as a common base technology.

Network

Networking is a central aspect in FlyingGuns. An own, unique technology named 'HeadQuarter' has been developed that is suitable to serve other games and non-game applications as a data distribution backbone. HeadQuarter provides a set of distributed 'subsystems' that maintain a shared state. Data transfer and messaging is handled by a package named 'ObjectBus' which in turn uses NIOs asynchronous TCP/IP.

Environment

FlyingGuns is meant to be not only a game, but an example of a gaming and distributed simulation framework. As that, it offers a sophisticated project setup, Ant build system, XML based resource management and a lot of tools and utilities.

Gameplay

A 'Red Baron' style, WW1 flightsimulator scenario was chosen as the basic gameplay. This setup has several advantages concerning the goals:

- Flightsimulators can live with moderate resources; the game is not carried by graphics or effects but by the 'feeling' of a planes motion.
- The motion motivated by physics is ideal for network interpolation because it can be very well anticipated algorithmically. Network artifacts are hard to recognize in a true 6DOF, smooth motion. So it still works in well high-latency, low-bandwidth networks.
- Typical weapons are machineguns - easy to handle in a network. Only the firing side can exactly determine wether something is hit or not.

One of the most common gameplay issues in flightsimulator games can be attacked easily: how to find the opponent? Where is the action?

- Slow, tightly-turning planes.
- Logarithmic radar.
- Many, many AI opponents. FlyingGuns can handle hundreds.

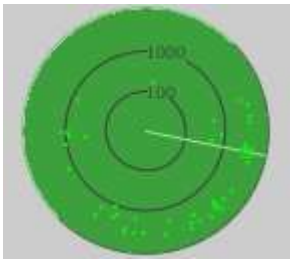


Illustration 1 log. radar

Together with short loading times this gives an instant action gameplay that supports a 'fast in, fast out' of a casual game.

Requirements and installation

FlyingGuns is known to work on Windows, Linux, Mac and Solaris.

	minimal	***recommended***
JRE	1.4.1_01	latest
Java3D	1.3	1.3.1
CPU	700MHz	1.5GHz
GPU	NVidia TNT2 class	GeForce2 class
Memory	128MB	256MB
Devices	keyboard, mouse	joystick (on Windows only)

FlyingGuns requires that a JRE and Java3D are installed upfront. Then there is not much of additional installation. Either FlyingGuns can be started directly off the WWW using WebStart technology or by starting the single-jar build provided for the contest. In the latter case though, no joystick support will be available due to the lack of a native library.

WebStart is the preferred way to start FlyingGuns. The corresponding URL is:

http://www.flyingguns.com/javagamescontest2004/

Starting the game

When them game starts, it first display a setup dialog:

Available planes

Planes are enumerated from an XML file and offered here. Currently the user may select any plane any time.

#enemies

Select number of AI opponents. It is recommended to set this number to 0 when playing a multiplayer game, because the human opponents otherwise can hardly be distinguished from the AI.

User/Password

Enter username. A password is currently not required, for not authenfication is implemented.

as server

The FlyingGuns client is able to act as a server simultaniously. When activated, the client is automatically connected to that server and the servername field displays some contact information (hostname, IP) about the server that the user may hand over to other players.
The server will not be started before 'Start new game' is activated.

Servername

When connecting to another server, this is the servername/IP address.

Port

The IP port number the server is listening.

Start new game

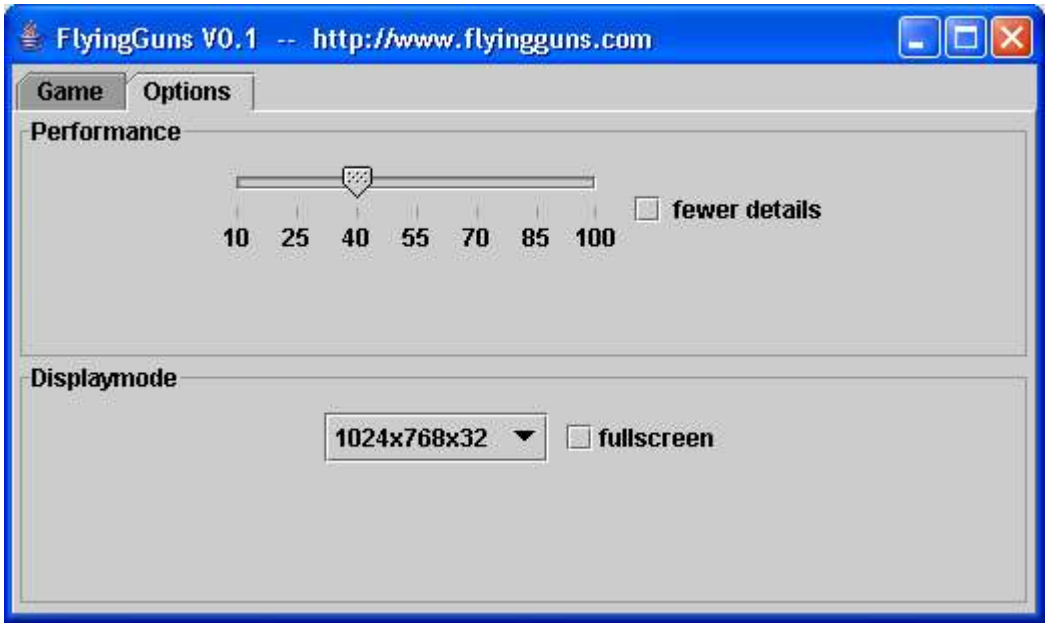
Start a new game either locally or as a server.

Connect

Connect to a running server. Advice: set #enemies to 0 when doing a multiplayer game!

Resume

When getting back to the this dialog from a running game e.g. in order to change the plane, return to the game. When no game has been started or connected to, nothing will happen.



FPS slider

FlyingGuns operates with a framerate limit in order leave time to other threads as well as to other applications coexisting on the desktop. Due to Java timing issues, that given number will never be achieved exactly. Setting the value disables the framerate limit. For this runs the gameloop tightly, other threads (e.g. those created by Java3D) may be throttled resulting in an unsmooth game!

fewer details

For weaker machines, it might be feasable to reduce graphic detail in order to improve performance. The resolution of the terrain will be degraded and fewer trees will get created. This setting is only evaluated when a new game is started.

Displaymode/fullscreen

The user selects a desired resolution and wether he likes to play in a desktop window or fullscreen.

Control

FlyingGuns can be controlled with mouse, keyboard or joystick. As with all flightsimulators, using a joystick is clearly preferred. There is no need to configure which inputdevice to use, because simple oversteering activates a device on a per-axis scale. That means one can use e.g. joystick for ailerons and elevator but still use the keyboard to control the rudder and throttle.

The most basic control is provided by the keyboard. Here is the set of keys to be used to steer the plane:

arrow keys left/right	ailerons
arrow keys up/down	elevator
a/d	rudder
w/s	throttle
c	cycle camera through planes
C	switch players camera
v	toggle nose/follow camera
ESC	break to setup dialog
alt-b	switch bottom GUI
alt-r	switch right GUI
alt-c	activate chat input

Although keyboard control is possible, it turned out that it is difficult esp. for aiming during dogfights. If no joytick is available, mouse control should be preferred.

Flying with the mouse

Mouse control in FlyingGuns is not like it can be found in many action games, esp. 1st-person shooters. It is more like really flying a plane. It is quite indirect and highly sensitive. The mouse doesn't actually control the motion, but the planes aileron and elevator.
Like flying a real plane, it is a good idea to keep controls centered most of the time and to handle them gently. Small movements and re-centering will help to get a feeling for the plane.

Fire the weapons

The machinegune will start to fire by either pressing the <space> key, left mouse button or the trigger on the joystick.
When flying with mouse, the mouse cursor does NOT indicate where the weapon aims to, but just indicates the state of the planes aileron and elevator!

Issues

With latest Java3D 1.3.1, a bug has been introduced that makes some aircrafts vanish erratically from time to time. The logical objects are still in place, but the models are just not rendered correctly. After that happened, we experienced crashes from the native code in some cases. If you can get hold of Java3D 1.3 or one of the following betas - they don't show that error and are definately better for FlyingGuns.

Java3Ds sound support is not the best in the world. Sometime we just couldn't manage that sounds (e.g. the 'firing' sound) is shut down correctly. So you might experience repeating sounds that don't go away any more.