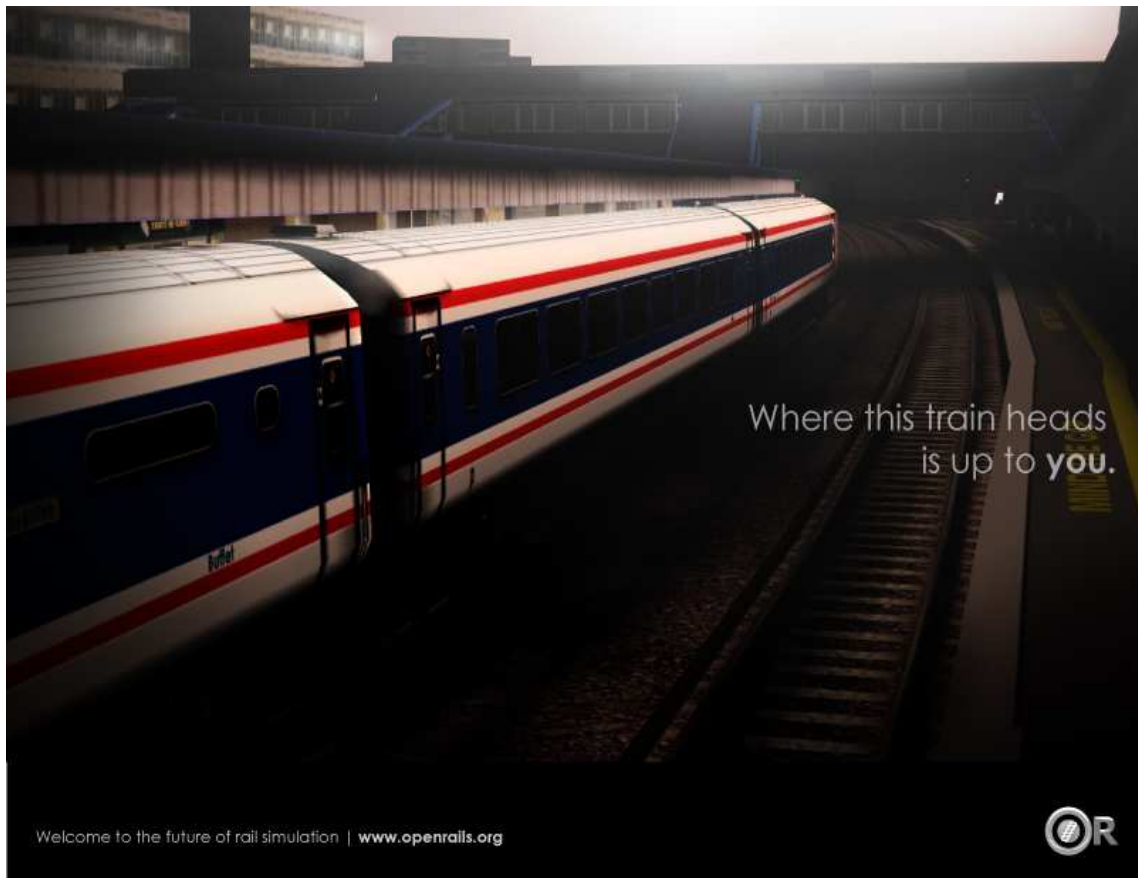




open rails™

The MSTs compatible railroad simulator.



Operations Manual

Version 0.645

Read the included End User License Agreement (EULA)

NO WARRANTIES. openrails.org disclaims any warranty, at all, for its Software. The Open Rails software and any related tools, or documentation is provided “as is” without warranty of any kind, either express or implied, including suitability for use. You, as the user of this software, acknowledge the entire risk from its use.

Trademark Acknowledgment

Open Rails, Open Rails Transport Simulator, ORTS, Open Rails trademark, openrails.org, Open Rails symbol and associated graphical representations of Open Rails are the property of openrails.org. All other third party brands, products, service names, trademarks, or registered service marks are the property of and used to identify the products or services of their respective owners.

Copyright Acknowledgment

©2010 openrails.org All rights reserved.

Table of Contents

Introduction	5
About Open Rails	5
Version	6
Community	6
Open Rails software platform	7
Architecture	7
Open Rails game engine	7
Frames Per Second (FPS) Performance.....	8
Game Clock and Internal Clock	8
Resource Utilization	8
Multi-Threaded Coding	8
Future & Roadmap	9
MSTS File Format Compatibility	12
Open Rails Software Capabilities	13
Open Rails game environment.....	13
Time of Day.....	13
Weather	14
Seasons.....	14
Open Rails HUD	14
Basic HUD display.....	14
Steam Engine HUD – Additional Information.....	15
Extended HUD – F5	16
Monitor Windows.....	18
Compass window.....	18
F1 Information Monitor.....	18
F4 Track Monitor.....	19
F8 Switch Monitor	19
F9 Train Operations Monitor	19
F10 Activity Monitor	20
Open Rails Physics	20
Train Cars (WAG).....	20

Engine – Classes of Motive Power.....	20
Engines – Multiple Units in Same Consist or AI Engines.....	21
Open Rails Braking.....	21
Using the F5 HUD Braking information in the Game.....	22
Dynamic Brakes.....	24
A Glimpse of the Future.....	24
Open Rails Train Engine Lights.....	27
Lighting Functions and Known Issues.....	28
Open Rails Activities.....	29
Player Paths, AI Paths, and How Switches are handled.....	29
Open Rails AI.....	29
Basic AI Functionality.....	29
Open Rails Signaling.....	30
Open Rails Multi-Player.....	30
Open Rails Best Practices.....	30
Polys vs. Draw Calls – What’s Important.....	30
Acknowledgements.....	31
License Agreement.....	33

Introduction

Open Rails software (OPEN RAILS) is a community developed and maintained project from openrails.org. Its objective is to create a new transport simulator platform that is first, compatible with routes, activities, consists, locomotives, and rolling stock created for MSTTS; and second, a platform for future content creation freed of the constraints of MSTTS.

Our goal is to enhance the railroad simulation hobby through a community designed and supported platform built to serve as a lasting foundation for an accurate and immersive simulation experience. By making the source code of the platform accessible to community members, we ensure that OPEN RAILS software will continually evolve to meet the technical, operational, graphical, and content building needs of the community. Open architecture ensures that our considerable investment in building accurate representations of routes and rolling stock will not become obsolete. Access to the source code eliminates the frustration of undocumented behavior and simplifies understanding the internal operation of the simulator without the time consuming trial and error experimentation typically needed today.

Open Rails software is just what the name implies – a railroad simulation platform that's open for inspection, open for continuous improvement, open to third parties and commercial enterprises open to the community, and best of all, an open door to the future.

About Open Rails

To take advantage of almost a decade of content developed by the train simulation community, Open Rails software is a complete new game platform that has backward compatibility with MSTTS content. By leveraging the community's knowledge base on how to develop content for MSTTS, Open Rails software provides a rich environment for community and payware contribution.

The primary objective of the Open Rails project is to create a railroad simulator that will provide 'true to life' operational experience. The Open Rails software is aimed at the serious train simulation hobbyist; someone who cares about loco physics, train handling, signals, AI behavior, dispatching, and most of all running trains in a realistic, prototypical manner. While the project team will strive to deliver an unparalleled graphical experience, 'eye candy' is not the primary objective of Open Rails software.

By developing a completely new railroad simulator, Open Rails software offers the potential to better utilize current and next generation computer resources, like graphics processing units (GPUs), multi-core CPUs, advanced APIs such as PhysX, and widescreen monitors, among many others. The software is distributed in restricted source code form so the user community can understand how the software functions to facilitate feedback and to improve the capabilities of Open Rails software.

Version

This document is based on version 0.645

Community

At the present time, Open Rails software is offered without technical support. Therefore, users are encouraged to use their favorite train simulation forums to get support from the community.

- Train-Sim.Com <http://forums.flightsim.com/vbts/>
- UK Train Sim <http://forums.uktrainsim.com/index.php>
- Elvas Tower <http://www.elvastower.com/forums/index.php?/index>

The Open Rails team is planning on launching a wiki that will offer the community a central resource for technical assistance on installation, creating content, known issues, tutorials, and a place for the community to provide feedback to the Open Rails team.

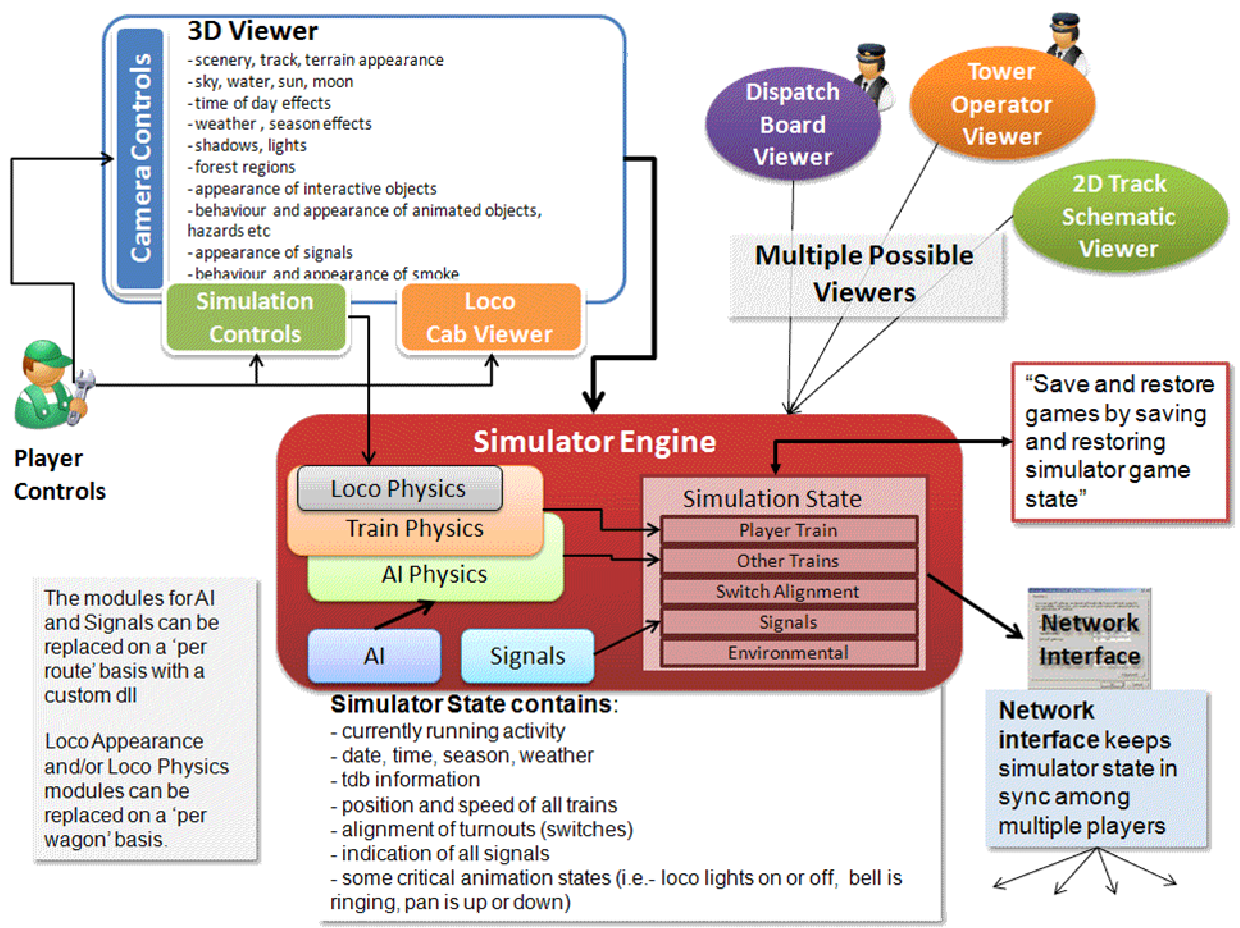
The Open Rails team is NOT planning on hosting a forum on the Open Rails website. We believe that the best solution is for the current train simulation forum sites to remain the destination for users who want to discuss topics relating to Open Rails software. The Open Rails team monitors and actively participates at these forums.

Open Rails software platform

Architecture

To better understand how the Open Rails game operates, performs, and functions, an architecture diagram lays out how the software code is organized. The architecture of the Open Rails software allows for modular extension and development, while providing standardized methods to customize the simulation experience.

i Please note that this diagram includes many capabilities and functions that are yet to be implemented.



Open Rails game engine

The Open Rails software is built on Microsoft's XNA game platform. Source code is developed in Microsoft Visual C# programming language.

The XNA Framework is based on the native implementation of .NET Compact Framework 2.0 for Xbox 360 development and .NET Framework 2.0 on Windows. It includes an extensive set of class libraries, specific to game development, to promote maximum code reuse across target platforms. The framework runs on a version of the Common Language Runtime that is optimized for gaming to provide a managed execution environment. The runtime is available for Windows XP, Windows Vista, Windows 7, and Xbox 360. Since XNA games are written for the runtime, they can run on any platform that supports the XNA Framework with minimal or no modification of the Game engine.¹



A license fee is payable to Microsoft to use XNA Game Studio for Xbox 360 game. At this time, the Open Rails team has not investigated whether the Open Rails software is suitable for Xbox.

Frames Per Second (FPS) Performance

For the current release, Open Rails development team has untethered the FPS performance from the synch rate of the monitor. This allows the development team to more easily document performance improvements. The Open Rails team at a later date may decide to limit FPS to the synch rate of the monitor.

Game Clock and Internal Clock

Like other simulation software, Open Rails software uses two internal “clocks”; a game clock and an internal clock. The game clock is required to synchronize the movement of trains, signal status, and present the correct game environment. The internal clock is used synchronize the software process for optimal efficiency and correct display of the game environment.

The Open Rails team is dedicated to ensuring the game clock properly manages time in the simulation, so that a train will cover the proper distance in the correct time. The development team considers this vital aspect for an accurate simulation by ensuring activities run consistently across community member’s computer systems.

Resource Utilization

Because Open Rails software is designed for Microsoft’s XNA game framework, it natively exploits today’s graphics cards ability to offload much of the display rendering workload from the computer’s CPU.

Multi-Threaded Coding

The Open Rails software is designed from the ground up to support up to 4 CPUs, either as virtual or physical units. Instead of a single thread looping and updating all the elements of the simulation, the software uses four threads for the main functions of the software.

Thread 1 - Main Render Loop (RenderProcess)

Thread 2 - Physics and Animation (UpdaterProcess)

¹ http://en.wikipedia.org/wiki/Microsoft_XNA

Thread 3 - Shape and Texture Loading/Unloading (LoaderProcess)

Thread 4 – Sound

The RenderProcess runs in the main game thread. During its initialization, it starts two subsidiary threads, one of which runs the UpdaterProcess and the other the LoaderProcess. It is important that the UpdaterProcess stay a frame ahead of RenderProcess, preparing any updates to camera, sky, terrain, trains, etc. required before the scene can be properly rendered. If there are not sufficient compute resources for the UpdaterProcess to prepare the next frame for the RenderProcess, the software reduces the frame rate until it can “catch up”.

Initial testing indicates that “stutters” are significantly reduced because the process (LoaderProcess) associated with loading shapes and textures when crossing tile boundaries do not compete with the main rendering loop (RenderProcess) for the same CPU cycles. Thread safety issues are handled primarily through data partitioning rather than locks or semaphores.

Ongoing testing by the Open Rails team and the community will determine what and where the practical limits of the software lie. As the development team receives feedback from the community, improvements and better optimization of the software will contribute to better overall performance – potentially allowing high polygon models with densely populated routes at acceptable frame rates.

Future & Roadmap

The goal of the Open Rails development team is to make as much of the existing MSTs content run on the game software. The development team initial focus is on providing a fairly complete visual replacement for MSTs that effectively builds on that content. As the Open Rails team has grown and matured along with the software, we believed that providing the community our vision was important.

From the perspective of the Open Rails team, the prior public download (v128) was the **Demonstration Release**. This release focused on demonstrating what was possible with an open, community-based development effort. Almost 15,000 downloads of the software illustrated the interest and excitement of the community, but recognition of the work yet to be done.

The last release (v360) was the **Foundation Release**. It combines a wealth of detailed information about physics, dispatcher, software performance and train forces in the F5 HUD. New and exciting improvements across the 3D environment, Dynamic Shadows, Engine Lighting, Slack Action and Coupler Physics, BIN-compatible Sound Management, Advanced Dispatcher, AI, and 2D Cab view (without animated controls) have been added. In addition, the team has implemented the initial F9 Train Operations Window, F4 Track Monitor, F8 Switch Monitor, and F10 Activity Monitor. Each is no means complete. It is a milestone release for the team. It established the foundation for our signaling, AI and dispatcher capabilities, advanced sound and the visual representation of the train sim world.

Looking ahead to 2011, we still have lots of work to do.

Our basic goals are unchanged - moving from Foundation to Realism and eventually to **Independence**. Following are just some of the highlights of what the community can expect in the upcoming months from the Open Rails team.

- Replacement of irrKlang sound engine with a new OpenAL implementation.

This provides Open Rails software with better ability to play, control, and extend sounds within the game engine. The major improvements you will see with the OpenAL approach are improved buffering/queuing of sounds via a dll written in C++ to open the wav files. This provides the foundation for much higher quality sounds than MSTools supports. The main goals of moving to an OpenAL sound engine are:

- The Garbage Collection activity is much lower; the queuing and playing functionality now allocates and frees CLR objects.
- It is now capable to use extensible format wave files
- The buffering solved the issue of the small audible gap between certain sounds
- The sound sources are optimized further by distance: they completely stopped and their resources are released when out of hearing distance and a reduced queuing functionality ensures the proper (re)activation
- This way the initial triggers are working as expected, they won't be completely restarted after coming into sight and run immediately when a sound source is activated

- Totally new signaling, interlocking and AI train control modules

This design allows for simulation of dark territory to today's high speed European systems and allows for signal and train control that spans operating eras from the Train Order & Timetable to the most sophisticated CTC systems. As part of this effort, Open Rails will offer dynamic AI where each train solves their pathing and routing more the 10 times per second. This means that each time an Activity is run, AI trains will react, in real time, to the timing of meets and other variables. The main goals achieved are:

- Minimize deadlocks between player and non-player trains
- Allow non-player trains to have complete pathing, movement orders (i.e. pickups, drop offs) and timetable management
- Simple ability to customize signals to accommodate regional, geographic, or operational differences
- Allow for dynamic pathing/signaling solutions within activities based on real time adjustments to player and non-player train location, timing and work orders

- Ability to use mixed signal environments - dark territory to full automatic in-cab train control within different or the same route
- Ability to pop-up a local dispatcher view of signaling and route to manually make adjustments and changes to signal states and routes through signals
- Ability to prioritize trains as per prototype operations
- Elimination of AI Dispatcher and its inherent logic problems to be replaced by self-aware AI trains

The net impact will be the implementation of a next generation Activity Editor for setting up paths (routes through signaling and interlocking) for both Player and AI trains.

- Next generation physics model for adhesion, traction, engine components and their interaction with the 3D world

The physics underlying adhesion, traction, engine components and their performance will be based on a world-class simulation model that takes into account all of the major components of diesel, electric and steam engines. This includes elements never before simulated in a consumer train simulation game like DC vs AC traction motors, number of axles in the truck design, wheel creep and hop, electric shunting, tilting, momentum, thermodynamics of the steam cycle including compensation for fuel types, plus modeling of different electric traction systems. Open Rails will approach the level of physics realism only available in professional simulators.

Some of the major goals for our new physics are:

- Freedom from the constraints and limitations of the MSTTS physics
- Ability to dynamically simulate component performance based on player input (traction motor degrading due to overheating); firing of steam locomotive including all major elements such as cylinders, blowers, grate, fuel types; component failure (turbo, electrical system, etc); or dynamic rail conditions on adhesion/traction physics
- Steam cycle implementation delivering an accurate simulation of the smallest to the largest compound engines as a SINGLE engine including different fuel types, firing strategies and steam usage (heating, injectors, etc.)
- Interaction of different diesel electric components to the overall physics covering first, second and third generation designs
- Next generation electric traction model capable of simulating different electrical traction systems: AC, DC and different generations of electric traction engines
- New sound triggers to give audio complement to the new physics model

For those models that do not have the upgraded Open Rails capabilities, existing MSTS content will continue to utilize the current Open Rails physics that matches the current MSTS physics model.

- New formats for model building to free content creators from the constraints of MSTS limitations

As we've announced, the Open Rails team is actively working on delivery a foundation for creating higher quality content than what's available via the MSTS infrastructure. Up first is support for DDS textures that provides better graphics and potentially provides even better performance as video cards & XNA can natively handle DDS format. At this time there has only been very preliminary discussion of beyond the current shape (s) file format.

As we move further into the year, the Open Rails team will be announcing more details about our road map. So stay tuned as it's going to be a fun ride in 2011.

MSTS File Format Compatibility

Open Rails software supports the MSTS file formats detailed below. The software uses a file parser to read the MSTS file information for use by the Open Rails software. Testing of the parser software indicates that it will locate many errors or malformation in these files that are not highlighted by the MSTS train sim software or by other utilities. In most cases, the Open Rails software will ignore the error in the file and run properly. Open Rails software logs these errors in a log file on the user's desktop. This log file may be used to correct problems identified by the Open Rails software.

Trainset:

The software currently supports Shape (.s); Shape Definition (.sd); Sound (.sms); and texture Ace (.ace) files; including displaying the correct LOD, alpha and transparency attributes. The software does NOT fully support the following file types: Engine (ENG); Wagon (WAG); it substitutes MSTS-style physics to enable the user to operate trains.

Consists:

Open Rails software reads and displays Consist files (.con) used for Player Train, AI Train, and Loose Consists in activities. The development team is aware that some consists may be flipped compared to how MSTS displays them.

Services:

Open Rails software supports MSTS Service files (.svc) for the creation of both Player and AI services.

Paths:

Open Rails software supports MSTS Path files (.pat) for determining the path of both Player and AI Trains.

Routes:

Open Rails software supports the following MSTS Route files with the limitations noted.

- Route Database file (.rdb) - CarSpawner is supported.
- Reference File (.ref) – Open Rails does not provide a Route Editor in the current release.
- Track Database file (.tdb) –
- Route File (.trk) – Level Crossings are supported. Overhead wire is not supported.
- Sigcfg (.dat) file – About 80% of signal & scripting capabilities are supported.
- Sigscr (.dat) file – About 80% of signal & scripting capabilities are supported.
- Speedpost (.dat) file – Not supported at the present time
- Spotter (.dat) file – Not supported at the present time
- Ssource (.dat) file – Not supported at the present time
- Telepole (.dat) file – Yes, supported
- Tsection (.dat) file – Yes, supported
- Ttype (.dat) file – Yes, supported

Environment:

Open Rails software does not support advanced water and dynamic weather effects at this time.

Activities:

Open Rails software supports limited activities. At this time, only Free Roam (“Explore Route”) activities are fully supported. In addition, Open Rails AI capabilities are limited in the current release which may result in unexpected behavior such as stand offs and other problems completing activities compared to MSTS since Open Rails has not implemented a complete signaling system. The Dispatcher uses “non-signaled” territory logic to control spawning and passing of Player and AI trains.

Satellite:

Open Rails software uses its own sky, cloud, sun, moon and precipitation effects developed exclusively for it. The starting parameters for time of day and weather are read from the activity file to determine the starting display in Open Rails software.

Open Rails Software Capabilities

Open Rails game environment

Time of Day

Open Rails software “reads” the **StartTime** from the MSTS .act file to determine what the game time is for the activity. In combination with the longitude and latitude of the route and the season, Open Rails computes the actual sun position in the sky. This provides an extremely

realistic representation of the time of day selected for the activity. For example, 12 noon in the winter will have a lower sun position in the northern hemisphere than 12 noon in the summer. Open Rails game environment will accurately represent these differences.

Once the activity is started, Open Rails software allows the player to advance the environment “time of day” independently of the movement of trains. Thus, the player train may sit stationary while the time of day is moved ahead or backward.

In addition, Open Rails offers similar functionality to the *time acceleration* switch for MSTs. Pressing the **Page Up** or **Page Down** keys doubles the game clock. Pressing the **Page Up** key again doubles the game clock speed. To return to ‘normal’ game clock speed, you must press the **Page Down** key the same number of times as the **Page Up** key was pressed to accelerate time.

Weather

Open Rails software determines the type of weather to display from the *Weather* parameter in the MSTs Activity file. The User may also adjust the level of cloud cover while the Activity is running by pressing the **Ctrl** key + “**plus**” or “**minus**” sign. This demonstrates Open Rails software’s foundation for dynamic weather effects in the game.

Seasons

Open Rails software determines the season, and its related alternative textures, to display from the *Season* parameter in the MSTs Activity file. Open Rails program supports all seasons available in MSTs Activity file – summer, spring, fall and winter.

Open Rails HUD

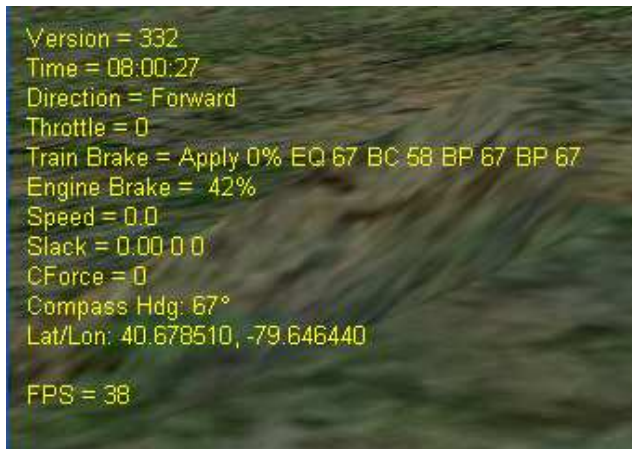
Basic HUD display

Selecting the F5 key displays the HUD information in the current version of Open Rails software. The User cannot modify the HUD at the present time. The following information is displayed:

- Version = The version of the Open Rails software you are running
- Time = Game time of the Activity
- Direction = Position of the Reverser - Diesel and Steam. Note: there is no Neutral position at the present time.
- Throttle = Displays the current position of throttle expressed as a percentage of full throttle. Throttle correctly uses Notches and configured % of power for Diesel engines or % of throttle for steam engines.
- Train Brake = Shows the current position of the train brake system and the PSI of the train brakes. Braking correctly reflects the braking system used; hold/release, self-lapping or graduated release. The Train brake HUD line has two Brake Reservoir pressure numbers: the first is Equalization Reservoir (EQ) and Brake Cylinder (BC) pressure numbers. The BP numbers reflect the brake pressure in the lead engine and the second is at the last car of the train.
- Engine Brake = percentage of independent engine brake. Not fully releasing will impact train brake pressures.

- Speed = the speed in Miles per Hour. At the present time, display of metric units is not supported.
- Slack =
- Coupler Force = Force exerted on couplers in Newtons. (1 Lbs of force = 4.4482216 Newtons)
- Camera Heading – the compass direction in degrees.
- Latitude/Longitude – in decimal degrees. Negative longitudes indicate western hemisphere; negative latitudes indicate southern hemisphere.
- FPS = Number of Frames rendered per second

An example of the basic HUD for Diesel locomotives follows:



Steam Engine HUD – Additional Information

When using a steam engine the following additional information is displayed in the HUD:

- Boiler PSI – stays constant at full boiler pressure defined in the ENG file parameter “MaxBoilerPressure”
- Steam Generation in lbs – based on key physics data in the ENG file. Steam generation is assumed to be ‘perfect’ from an operational perspective – fire temp, water level, etc. and is limited by the “MaxBoilerOutput” parameter in the ENG file
- Steam Usage in lbs – based on entirely new physics code developed by the Open Rails team and is calculated based on parsing the eng file for the following parameters: number of cylinders; cylinder stroke; cylinder diameter; boiler volume; maximum boiler pressure; maximum boiler output; exhaust limit; and basic steam usage. An example of the basic HUD for Steam locomotives follows:

is reserved for a train and '=' otherwise. The digit or letter corresponds to the number of the train the track is reserved for ('A' for train number 10). A '/' is added at a junction node that is a potential passing location. If the path includes a passing section the start of the passing section will be marked with a '\' and the passing track vector nodes will be represented by '_' characters. The main track part of the passing section is marked by other characters up to the next '/'. A '?' is added to mark a reverse point.

The dispatcher works by reserving track vector nodes for each train. An AI train will be allowed to move (or start) only if all of the nodes up to the next potential passing location are not reserved for another train. If this condition cannot be met, the AI train will not spawn.

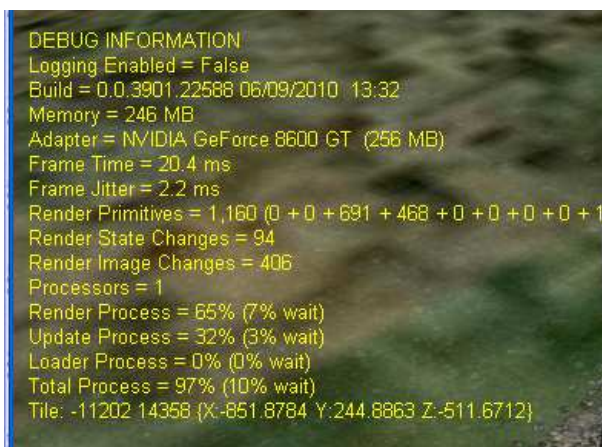
There are other reasons that an AI train might not appear. The current dispatcher assumes that all routes are unswitched. The dispatcher issues a track authority (which is similar to a track warrant) to all trains. For an AI train to start the tracks it needs must not already be reserved for another train. The dispatcher compares the paths of the trains to identify possible passing points and then reserved tracks for a train up until a passing point. When a train gets near the next passing point the reservation is extended to the next one. The end result is that an AI train won't be allowed to start if it needs to occupy the same track as the player train and that track has already been reserved for the player.

At the moment reservations are for both directions. The Open Rails development team plans on changing this in the future to allow trains to follow each other.

Basic F5 HUD + Debug Information

The third extended F5 HUD display replaces the Dispatcher information with Debug information. Pressing the F12 key enables frame by frame logging of the performance of Open Rails software. This is helpful in debugging performance related problems or stuttering. Logging file is automatically saved in the **Open Rails/Program** folder with the file name "dump.csv".

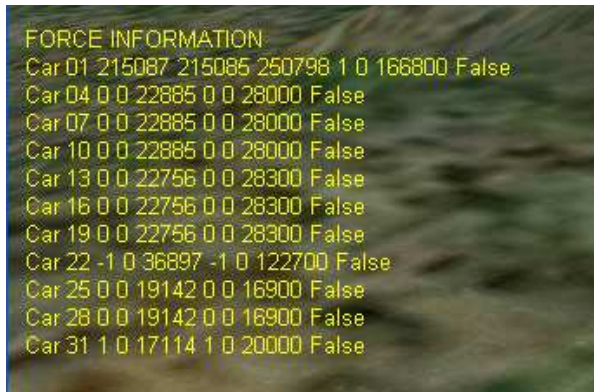
Wide varieties of parameters are shown from frame wait and render speeds in milliseconds, to number of primitives, Process Thread resource utilization and number of Logical CPUs from the system's bios.



```
DEBUG INFORMATION
Logging Enabled = False
Build = 0.0.3901.22588 06/09/2010 13:32
Memory = 246 MB
Adapter = NVIDIA GeForce 8600 GT (256 MB)
Frame Time = 20.4 ms
Frame Jitter = 2.2 ms
Render Primitives = 1,160 (0 + 0 + 691 + 468 + 0 + 0 + 0 + 0 + 1)
Render State Changes = 94
Render Image Changes = 406
Processors = 1
Render Process = 65% (7% wait)
Update Process = 32% (3% wait)
Loader Process = 0% (0% wait)
Total Process = 97% (10% wait)
Tile: -11202 14368 (X:-851.8784 Y:244.8863 Z:-511.6712)
```

Basic F5 HUD + Train Force Information

The fourth extended HUD display replaces the Debug information with Force information. The first number is the total force acting on the car. This is the sum of the other forces after the signs are properly adjusted. The second number is the motive force which should only be non-zero for locomotives. The third number is the friction force calculated from the Davis equation. The fourth value is the force due to gravity. The fifth value is the coupler force between this car and the next (negative is pull and positive is push). The last number is mass in kg. All the force values are in Newtons. Many of these values are relative to the orientation of the car, but some are relative to the train. The last field is "True" if the car is flipped with respect to the train and otherwise it's "False".



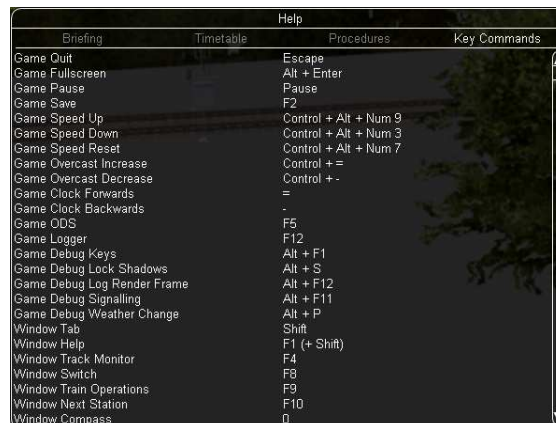
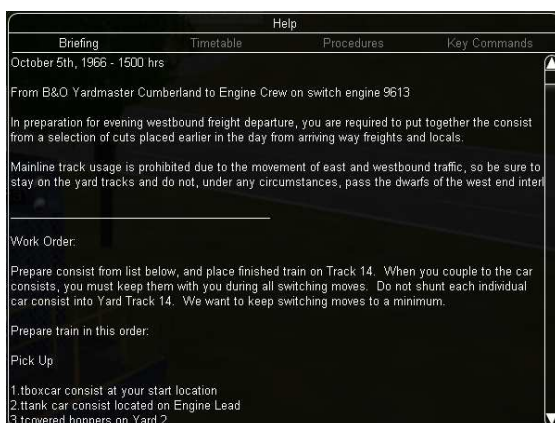
Monitor Windows

Compass window

Open Rails software displays a compass that provides a heading based on the direction of the camera. To activate the compass window press the **0** (zero) key. To deactivate the compass window, press the **0** (zero) key a second time.

F1 Information Monitor

Open Rails software now offers Briefing, Timetable, Procedures and Keyboard commands in a tabbed format using the F1 key.



F4 Track Monitor

Track Monitor show current speed in Miles per Hour (or Kilometers per Hour depending on the Route definition) and the Projected Speed based on the current throttle setting. Signal status has not been implemented at this time. In addition, the distance to the next Reverse Point is shown or to the end of the currently authorized path (message = End of Authorization) in the case of a passing siding.



This displays the distance in meters to the next location where the Player must either reverse direction (message = Reverser), make a station stop or if a switch needs to be thrown. The "station" item changes depending on what's "next" according to the virtual dispatcher. If a meet with an AI train is ahead, the distance to the end of the siding will be displayed (message = End of Authorization).

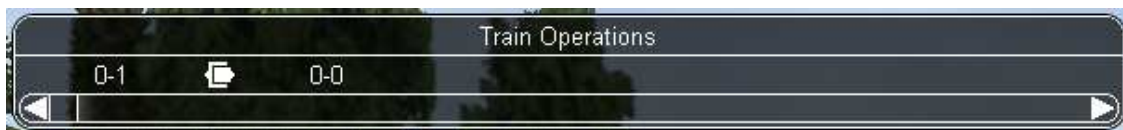
F8 Switch Monitor

Switch Monitor show the status of the turnout directly in front and behind the train. Clicking on the leg of the turnout will throw the turnout in that direction. Alternatively, clicking on the switch in the Main Window will also throw the turnout, as will using the **G** key and **Shift+G** key



F9 Train Operations Monitor

Open Rails Train Operations window is similar in function to the F9 window in MSTs. This is the development team's initial implementation of the feature. Clicking on the coupler icon between any two cars uncouples consist at that point. Cars are numbered according to their UiD in the Consist file (.con) or UiD in the Activity file (.act). Only the last two numbers are shown in the window, so duplicate numbers can and will appear. Scrolling is accomplished by clicking on the arrows at the left and right bottom corner of the window.



F10 Activity Monitor

The early implementation Activity Monitor in Open Rails software is similar in function to MSTs. It records the Arrival time of your train versus the Actual time as well as the Departure Time. At the present time a text message alerts the engineer as to the proper departure time, the departure sound is not implemented at the present time.



Next Station				
				15:03:07
Station	Arrive	Actual	Depart	Actual

Open Rails Physics

Train Cars (WAG)

Open Rails physics is in the early stages of development. At the present time, Open Rails physics calculates resistance based on partial real world physics: gravity, mass and rolling resistance. This is calculated individually for each car in the train. The cars are summed up as a single unit for faster calculation by the program. Physics calculation are redone whenever train cars are uncoupled or coupled to the consist.

The program calculates rolling resistance, or friction, based on parsing the Friction parameters in the WAG file. The Open Rails software identifies whether the WAG file used FCalc or some other friction data. If FCalc was used to determine the Friction variables within the .waf file, the Open Rails software compares that data to the Open Rails Davis equations to identify the closest match with the Open Rails Davis equation. If no-FCalc Friction parameters are used in the WAG file, the Open Rails software ignores those values, substituting its actual Davis equation values for the train car.

At the present time, there is no “slack” action for couplers. The Open Rails team is reviewing options to implement the physics of couplers.

Engine – Classes of Motive Power

Open Rails software provides for different classes of engines: diesel, electric, steam and default. If needed, additional classes can be created with unique performance characteristics.

At the present time, diesel and electric locomotive physics use the default engine physics. Default engine physics simply uses the MaxPower and MaxForce parameters to determine the pulling power of the engine modified by the Reverser and Throttle position.

Steam engines employ a more developed, but still not complete physics model. Open Rails steam physics parses the eng file for the following parameters: NumCylinders; CylinderStroke; CylinderVolume; BoilerVolume; MaxBoilerPressure; MaxBoilerOutput; ExhaustLimit; and

BasicSteamUsage. These parameters are used as input values for Open Rails independently developed steam locomotive physics equations.

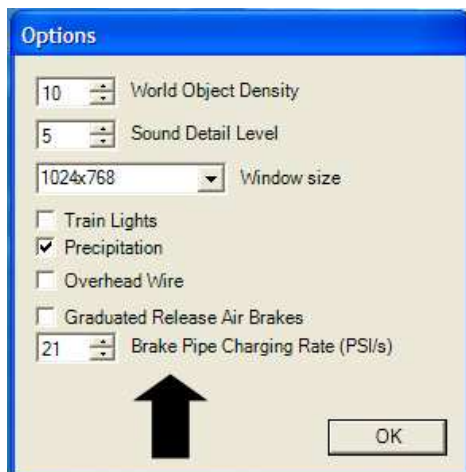
Engines – Multiple Units in Same Consist or AI Engines

Locomotives can either be controlled by a player, or controlled by the train's MU signals for braking and throttle position, etc. The player controlled loco generates the MU signals which pass along to every unit in the train. For AI trains, the AI software directly generates the MU signals - there is no player controlled train. In this way, all engines use the same physics code for power and friction.

i *This software model will ensure that non-player controlled engines will behave exactly the same way as player controlled ones.*

Open Rails Braking

Open Rails software has implemented its own braking physics in the current release. It is based on the Westinghouse 26C and 26F air brake and controller system. Open Rails braking will “read” the type of braking from the ENG file to determine if the braking physics uses passenger or freight standards, self-lapping or not. This is controlled within the Option menu as shown below.



Selecting **Graduated Release Air Brakes** in the Options menu allows a partial release of the brakes. Some 26C brake valves have a cut-off valve that has three positions: passenger, freight and cut-out. Checked is equivalent to passenger standard and unchecked is equivalent to freight standard.

The **Graduated Release Air Brakes** option controls two different features. If the train brake controller has a self-lapping notch and the **Graduated Release Air Brakes** box is checked, then the amount of brake pressure can be adjusted up or down by changing the control in this notch. If the **Graduated Release Air Brakes** option is not checked, then the brakes can only be increased in this notch and one of the release positions is required to release the brakes.

Another capability controlled by the **Graduated Release Air Brakes** checkbox is the behavior of the brakes on each car in train. If the **Graduated Release Air Brakes** box is checked, then the brake cylinder pressure is regulated to keep it proportional to the difference between the emergency reservoir pressure and the brake pipe pressure. If the **Graduated Release Air Brakes** box is not checked and the brake pipe pressure rises above the auxiliary reservoir pressure, then the brake cylinder pressure is released completely at a rate determined by the retainer setting.

At the moment only single pipe air brakes are modeled. So the auxiliary reservoir needs to be charged by the brake pipe and depending on the WAG file parameters setting this can delay the brake release. When the **Graduated Release Air Brakes** box is not checked, the auxiliary reservoir is also charged by the emergency reservoir (until both are equal and then both are charged from the pipe). When the **Graduated Release Air Brakes** box is checked, the auxiliary reservoir is only charged from the brake pipe. The Open Rails software implements it this way because the emergency reservoir is used as the reference pressure for regulating the brake cylinder pressure.

The end result is that you will get a slower release when the **Graduated Release Air Brakes** box is checked. This shouldn't be an issue with two pipe air brakes because the second pipe can be the source of air for charging the auxiliary reservoirs.

Open Rails software modeled most of this graduated release car brake behavior based on the 26F control valve, but this valve is designed for use on locomotives. That valve uses a control reservoir to maintain the reference pressure and Open Rails software simply replaced the control reservoir with the emergency reservoir.

Increasing the **Brake Pipe Charging Rate (PSI/Second)** value controls the charging rate. Increasing the value will reduce the time required to recharge the train; while decreasing the value will slow the charging rate. However, this might be limited by the train brake controller parameter settings in the ENG file. The brake pipe pressure can't go up faster than the equalization reservoir.

The default value, 21, should cause the recharge time from a full set to be about 1 minute for every 12 cars. If **Brake Pipe Charging Rate (PSI/Second)** value is set to 1000, the pipe pressure gradient features will be disabled and will disable some of the other new brake features, but not all of them.

Brake system charging time depends on the train length as it should, but at the moment there is no modeling of main reservoirs and compressors.

Using the F5 HUD Braking information in the Game

This helps users of Open Rails understand the status of braking within the game. Open Rails braking physics is more realistic than MSTTS, as it replicates the connection, charging and exhaust of brake lines. When coupling to static consists, please notice that the brake line for the newly added car doesn't have any values. This is because the train brake line/hose has not been connected.

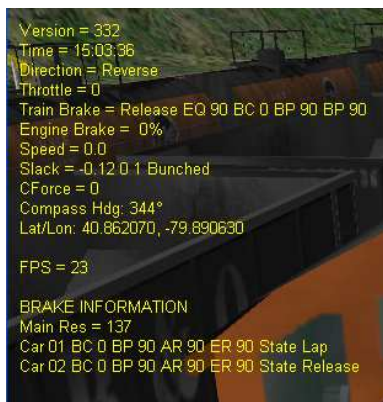


Pressing the **Backslash** key (\) connects the brake hoses for all cars that have been coupled to the engine. Upon connection charging of the brake train line commences. Open Rails uses a default charging rate of about 1 minute per

Notice that upon connection all the newly added cars have their handbrakes set to 100%.



Pressing the **Colon** key (:) will release all handbrakes on the consist as shown below.



Pressing the **Shift +?** (Question mark) will immediately fully charge the train brakes line if you don't want to wait for the train brake line to charge.



Dynamic Brakes

Open Rails software supports dynamic braking for Engines. To increase the Dynamic brakes press **Period** (.) and **Comma** (,) to decrease them. Initially dynamic brakes are off, the throttle works and there is no dynamic brake line in the HUD. To turn on dynamic brakes set the throttle to zero and then press **Period**, this will add a dynamic brake line to the HUD. Pressing **Period** successive times increases the Dynamic braking forces. To turn off the Dynamic brakes press **Comma** after setting the brakes to 0%. The throttle will not work when the Dynamic brakes are on.

The Dynamic brake force as a function of control setting and speed can be defined in a *DynamicBrakeForceCurves* table that works like the *MaxTractiveForceCurves* table. If there is no *DynamicBrakeForceCurves* defined in the ENG file, than one is created based on the MSTs parameter values.

Dynamic brake sounds have not been implemented at this time.

A Glimpse of the Future

Proprietary Open Rails Braking Parameters – File Inclusions for ENG files

Open Rails has implemented additional specific braking parameters to deliver realism in braking performance in the sim.



The Open Rails team intends to provide separate documentation detailing the usage, syntax, and methodology of how to use these new braking and performance parameters. Check the Open Rails web site periodically for current status.

Following are a list of specific OR parameters and their default values. The default values are used in place of MSTs braking parameters; however, two MSTs parameters are used for the release state: *MaxAuxiliaryChargingRate* and *EmergencyResChargingRate*

wagon(brakepipevolume - Volume of car's brake pipe in cubic feet (default .5). This is dependent on the train length calculated from the ENG to the last car in the train. This aggregate factor is used to approximate the effects of train length on other factors.

i *Strictly speaking this value should depend on the car length, but the Open Rails Development team doesn't believe it's worth the extra complication or CPU time that would be needed to calculate it in real time. We will let the community customize this effect by adjusting the brake **servicetimefactor** instead, but the Open Rails Development team doesn't believe this is worth the effort by the user for the added realism.*

engine(mainreschargingrate - Rate of main reservoir pressure change in PSI per second when the compressor is on (default .4).

engine(enginebrakerelaserate - Rate of engine brake pressure decrease in PSI per second (default 12.5).

engine(enginebrakeapplicationrate - Rate of engine brake pressure increase in PSI per second (default 12.5).

engine(brakepipechargingrate - Rate of lead engine brake pipe pressure increase in PSI per second (default 21).

engine(brakeservicetimefactor - Time in seconds for lead engine brake pipe pressure to drop to about 1/3 for service application (default 1.009).

engine(brakeemergencytimefactor - Time in seconds for lead engine brake pipe pressure to drop to about 1/3 in emergency (default .1).

engine(brakepipetimefactor - Time in seconds for a difference in pipe pressure between adjacent cars to equalize to about 1/3 (default .003).

An example of the expanded Open Rails file syntax is as follows:

```
include ( ..\bc13ge70tonner.eng )

Wagon (
    MaxReleaseRate ( 2.17 )
    MaxApplicationRate ( 3.37 )
    MaxAuxiliaryChargingRate ( .4 )
    EmergencyResChargingRate ( .4 )
    BrakePipeVolume ( .4 )
)

Engine (
    AirBrakeMainresvolume ( 16 )
    MainResChargingRate ( .5 )
    BrakePipeChargingRate ( 21 )
    EngineBrakeReleaseRate ( 12.5 )
    EngineBrakeApplicationRate ( 12.5 )
    BrakePipeTimeFactor ( .00446 )
    BrakeServiceTimeFactor ( 1.46 )
    BrakeEmergencyTimeFactor ( .15 )
    MaxTractiveForceCurves (
        0 ( 0 0 50 0 )
    )
)
```

```

.125 (
    0 23125
    .3 23125
    1 6984
    2 3492
    5 1397
    10 698
    20 349
    50 140
)
.25 (
    0 46250
    .61 46250
    1 27940
    2 13969
    5 5588
    10 2794
    20 1397
    50 559
)
.375 (
    0 69375
    .91 69375
    2 31430
    5 12572
    10 6287
    20 3143
    50 1257
)
.5 (
    0 92500
    1.21 92500
    5 22350
    10 11175
    20 5588
    50 2235
)
.625 (
    0 115625
    1.51 115625
    5 34922
    10 17461
    20 8730
    50 3492
)
.75 (
    0 138750
    1.82 138750
    5 50288
    10 25144
    20 12572
    50 5029
)
.875 (
    0 161875
    2.12 161875
    5 68447
    10 34223
    20 17112
    50 6845
)
1 (
    0 185000

```

```
2.42 185000
5 89400
10 44700
20 22350
50 8940
```

Dynamic Tractive Force

The Open Rails development team has been experimenting with max/continuous tractive force, where it can be dynamically altered during game play using the *MaxTractiveForceCurves* parameter as shown earlier. The parameters were based on the Handbook of Railway Vehicle Dynamics. This says the increased traction motor heat increase resistance which decreases current and tractive force. I used a moving average of the actual tractive force to approximate the heat in the motors. Tractive force is allowed to be at the maximum per the ENG file, if the average heat calculation is near zero. If the average is near the continuous rating than the tractive force is de-rated to the continuous rating. There is a parameter called *ContinuousForceTimeFactor* that roughly controls the time over which the tractive force is averaged. The default is 1800 seconds.

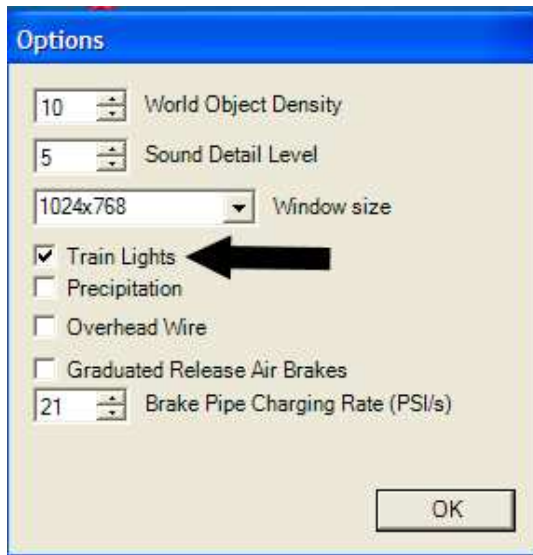
Brake Retainers

The retainers are controlled using the left and right **Brace** (**{** and **}**) keys. The left Brace (**{**) key will reset the retainer on all cars to exhaust (the default position). The retainer setting is increased each time the right Brace (**}**) key is pressed. First the number of cars is increased (25%, 50% and then 100%) and then the retainer setting is changed (low pressure, high pressure and then slow direct). For 25% the retainer is set on every fourth car starting at the rear of the train. 50% is every other car and 100% is every car. These changes can only be made when stopped. When the retainer is set to exhaust, the ENG file release rate value is used, otherwise the release rates are hard coded based on some AB brake documentation used by the Open Rails development team.

Open Rails Train Engine Lights

Enabling Open Rails Lighting

You must select the Engine Lighting checkbox for lighting to work. At the present time, the Engine Lighting checkbox is DISABLED by default.



Lighting Functions and Known Issues

Phase one of the Open Rails Lighting adds basic train light functions as detailed below:

- The **H** key controls these light states: OFF-DIM-ON. As in MSTs, tap once for dim, then again for bright. Press **CTRL+H** in succession to reverse the sequence.
- Dim/bright lights work as long as the lights are set up that way in the ENG file.
- Fade-in/fade-out is supported.
- Scenery illumination is enabled but only when the headlight is bright.

Lighting states are now fully implemented:

- Penalty, Control, Service, TimeOfDay, Weather, and Coupling modes.
- Multiple States

Known Issues:

- Where the Lights block in an ENG file contains in-line comments (format: #comment), the comments are ignored but each one generates an error message in the console window. Not to worry: The message is more an advisory than an actual error.
- Glow-light objects are built as quads whose "radius" is specified in the Lights block. The position of each quad is also specified there. If two adjacent lights are positioned too closely or if their radius is too large, they will overlap and Z-fighting will occur. This occasionally can be seen in both MSTs and Open Rails software.

Fixing Open Rails Lighting Problems

By organizing the ENG lighting section to the following specification, usually the Rails Lighting errors will be eliminated. The Lighting section must follow the specified sequence of lighting definitions:

1. Sphere of lights, both headlights which also includes dim and bright,

2. ditch lights,
3. the rest of the lights.

If errors still occur, removing the comments from the lighting section almost always fixes the error conditions. The reason eliminating “comments” fixes Open Rails lighting errors is that there are many ways the community have used this feature in MSTs. The Open Rails team doesn’t believe it’s a wise investment of their time to accommodate all the potential ways the community has implemented “comments”. In the future, the Open Rails team will provide a standardized methodology for handling comments within the Lighting section of the ENG file that is backwards compatible with MSTs.

Open Rails Activities

Player Paths, AI Paths, and How Switches are handled

If the player path requires a switch to be aligned both ways, the alignment that is the last on the path is used. If an AI train crosses the player path before the player train gets there, the AI train will leave the switches aligned for the main route (the default setting for most switches)

If you throw a switch to move into a siding, the switch at the far end of the siding is aligned to let you leave when your train first occupies the siding. But after that it is not changed back to its original setting. If the switch gets throw the other way, you can leave the siding with the switch aligned incorrectly. If you uncouple and re-couple to the train while it occupies the misaligned switch, the rear end of the train will switch tracks.

Open Rails AI

Basic AI Functionality

- The software has a basic AI system for now. In time, the AI system will become more advanced with new features.
- AI trains can pass if both paths have passing sections defined at the same place. Waiting points and reverse points should work, although the trains might not stop in the same location as MSTs.
- AI trains will stop near any switch that is not lined properly, throw the switch and then proceed. (This function will probably get improved and enhanced with community feedback)
- AI trains can uncouple and leave cars.
- At the moment uncoupling only works from the front of the train and coupling doesn’t work yet. To uncouple put a waiting point in the path with a waiting time greater than 40000 seconds.
- A value of 4NNSS where NN and SS are two digit numbers will cause the first NN cars to be uncoupled and the train will move again after waiting SS seconds.
- Priorities: AI trains should start as scheduled as long as there is no other AI train already on a conflict path.

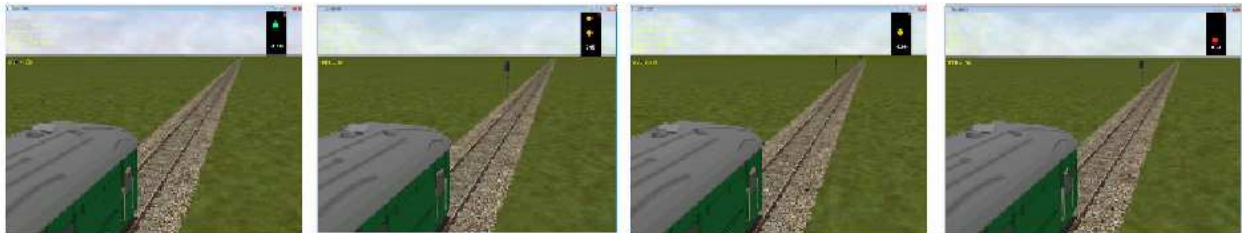
Open Rails Signaling

At the present time, Open Rails is in the initial proof of concept stage regarding signals and signal block detection for player and AI trains. *This is NOT included in the public demo download version.*

A basic Signal Monitor (F4 Key) has been developed for testing the Open Rails signaling efforts. *Please note the Signal Monitor graphic does not necessarily represent the final Signal Monitor.*

What the Proof of Concept accomplished:

- Linking the signals together
- Update the signals using a default procedure
- Determine the distance to next signal and aspect from the driver train



The Signaling Proof of Concept will be moving to the next phase of the project, covering:

- Integration with the SIGCFG file for different signal types
- Recognition of different Signal Types, only NORMAL signals work at the moment
- Account for signals with multiple heads.
- Display the aspect on the actual signal shape.
- AI trains to recognize signals and behave according to the signal indication.
- Signal Scripting.

Open Rails Multi-Player



Multipayer capabilities are being investigated and preliminary design work is in progress for Client Server multi-player capabilities.

Open Rails Best Practices

Polys vs. Draw Calls – What's Important

Poly counts are still important in Open Rails software, but with newer video cards they're much less important than in the early days of MSTs. What does remain important to both environments are Draw Calls.

A Draw Call occurs when the CPU sends a block of data to the Video Card. Each model in view, plus terrain, will evoke one or more Draw Calls *per frame* (i.e., a frame rate of 60/second means all of the draw calls needed to display a scene are repeated 60 times a second). Given the large number of models displayed in any scene and a reasonable frame rate, the total number of Draw Calls per second creates a very large demand on the CPU. Open Rails software will adjust the frame rate according to the number of required Draw Calls. For example, if your CPU can handle 60,000 Draw Calls per second and the scene in view requires 1000 Draw Calls, your frame rate per second will be 60. For the same CPU, if the scene requires 2000 Draw Calls, your frame rate per second will be 30. Newer design / faster CPU's can do more Draw Calls per second than older design / slower CPU's.

Generally speaking, each Draw Call sends one or more polygon meshes for each occurrence of a texture file for a model (and usually more when there are multiple material types). What this means in practice is if you have a model that uses two texture files and there are three instances of that model in view there will be six draw calls – once for each of the models (3 in view) times once for each texture file (2 files used), results in six Draw Calls. As an aid to performance Open Rails will examine a scene and will issue Draw Calls for only the models that are visible. As you rotate the camera, other models will come into view and some that were in view will leave the scene, resulting in a variable number of Draw Calls, all of which will affect the frame rate.

Model builders are advised that the best performance will result by not mixing different material types in a texture file as well as using the fewest number of texture files as is practical.

Players are advised to adjust camera positions to cull models from being in view whenever frame rates fall to unacceptable levels and to adjust the camera again to include more models when frame rates are high.

Acknowledgements

The Open Rails team would like to acknowledge the following for their work, without which Open Rails would not be possible.

- John Sandford, Jim Jendro, Wes Card. For deciphering the MSTS tile files and providing computer algorithms that go a long way toward helping us do the same.
- Riemer Grootjans. For his informative web tutorials and detailed code, and for his book, "XNA 3.0 Game Programming Recipes."
- Jan Vytlačil. For showing us how to make it rain and snow.
- Paul Bourke. For the high-resolution star maps of the northern and southern hemispheres.

And Wayne Campbell. For inspiring this improbable journey.

License Agreement

Open Rails transport simulator

(“Open Rails”)

End User License Agreement

IMPORTANT-READ CAREFULLY: THIS END-USER LICENSE AGREEMENT (“EULA”) IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR A SINGLE ENTITY HEREBY REFERRED TO AS “YOU”) AND OPEN RAILS.ORG FOR THE ABOVE REFERENCED SOFTWARE PRODUCT(S), WHICH INCLUDES COMPUTER SOFTWARE AND MAY INCLUDE DOWNLOADED BINARY OR SOURCE CODE FILES, AND “ONLINE” OR ELECTRONIC DOCUMENTATION. THE SOFTWARE PRODUCT INCLUDES ANY UPDATES, SUPPLEMENTS, PATCHES, OR MAINTENANCE PACKS PROVIDED BY OPENRAILS.ORG. BY INSTALLING, COPYING OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA. IF YOU DO NOT AGREE TO THE TERMS OF THIS EULA, DO NOT INSTALL OR USE THE SOFTWARE. YOU SHALL INFORM ALL USERS OF THE SOFTWARE OF THE TERMS AND CONDITIONS OF THIS EULA.

This EULA, grants You, the user, a non-exclusive license to use the Software under the terms and conditions stated herein. You agree that all updates, enhancements, maintenance releases, patches, bug-fixes or other modifications to the Software provided to You, on a when and if available basis, shall be governed by the terms and conditions, including the limited warranty, exclusive remedies and limitations of liability provisions, contained in this EULA, or the then-current version of this EULA.

You may: (i) use the Software on any number of computers You own; (ii) make modifications to the original source code of the software for your own personal use; (iii) distribute the compiled version of the software; (iv) distribute software plug-ins, add-on files, and any other secondary content created for the software; (v) make copies of the Software, documentation or other user information, tools, or content accompanying the Software for back-up purposes; (vi) make a copy of or print documentation provided in electronic form for Your internal use only; and, (vii) use Open Rails trademarks solely for these purposes, but You must incorporate all patent, copyright, trademark and other notices included on the materials on any copies that You make.

You may not: (i) sell, sublicense, rent, or lease the Software to another party; (ii) disseminate any modification, revision, correction, or change, in any manner, to the Software source code except and only to the extent that such activity is expressly permitted by this EULA or by written permission of openrails.org; (iii) transfer or assign Your rights to use the Software; (iv) use the Software in violation of applicable local, federal or International laws or regulations; (v) use the Software for any purpose other than as permitted in this EULA; or, (vi) remove, destroy, erase, alter or otherwise modify Open Rails trademarks.

NO WARRANTIES. Openrails.org disclaims any warranty, at all, for its Software. The OPEN RAILS software and any related tools, or documentation is provided “as is” without warranty of any kind, either express or implied, including suitability for use. You, as the user of this software, acknowledge the entire risk from its use.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall openrails.org or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if openrails.org has been advised of the possibility of such damages.

COPYRIGHT. Any intellectual property or content which may be accessed through the use of Open Rails software program is the property of the respective property or content owner and may be protected or prohibited by its applicable copyright. You, as the user of the software, must determine the applicability of any third party intellectual property or content. This EULA grants you no rights to use such content. The Open Rails software program(s) itself, title and copyrights, any accompanying documentation, and copies of the software program are the property of openrails.org.

Trademark Acknowledgment

Open Rails, OPEN RAILS Transport Simulator, OPEN RAILS, OPEN RAILS trademark, openrails.org, OPEN RAILS symbol and associated graphical representations of OPEN RAILS are the property of OPEN RAILS.org. All other third party brands, products, service names, trademarks, or registered service marks are the property of and used to identify the products or services of their respective owners.

Copyright Acknowledgment

©2010 openrails.org All rights reserved.