Civil Game Documentation

Release 0.84.0

The Civil game team

Aug 30, 2018

Contents

Civil is a real-time strategy game allowing players to take part in scenarios set during the American Civil war.

CHAPTER 1

Introduction

Welcome the the Civil online help! Here you can get quick information about how to play Civil. To get a list of the currently active keyboard shortcuts you can press F2.

A little button labeled “Menu” can be seen in one of the windows. Clicking it brings up a menu of all available options.

1.1 What’s on the map?

The main in-game display in Civil contains the map and a few extra features. The map is where all units are shown, along with the orders they have. Union units are shown using blue icons, and confederate units with gray/brown icons. A little rectangle under the unit shows the facing of the unit.

The map is made up of various types for terrain that affects movement, combat and visibility if various ways. See “Terrain types” for more info.

A number of windows also can be seen on the map. See “Windows” for more info about these.

Up in the right corner of the map is an area where messages to the player will be shown. These are informational messages from the game, chat messages from you opponent and other info. The messages will slowly scroll up and away from the map.

1.1.1 Toggleable unit features in the map

Several features in the map related to unit info can be toggled on and off, according to how much information you want. If the map feels cluttered you can turn off the features that you do not need at the moment, or are not interested in at all.

To select which features are shown press F3. This brings up a dialog where all features can be individually toggled. See the link below for descriptions of what each of these features are.

1.1.2 Toggleable features window

This dialog allows you to choose what to display on the map. Select the wanted features and click Ok.

“Map labels” toggles wether to show some informative geographical labels in the map, such as village and hill names.

“Objectives” toggles objectives areas on and off.

“Unit symbols” toggles wether to show the rectangular unit symbols. The symbols visualize the unit facing (the triangle) and strength (size). The symbols can be toggled for own and enemy units

“Unit icons” toggles wether to show the icon representation of the units. This shows cavalry as men with horses etc. Icons only show two facings and the strength as a little horizontal bar.

“Unit orders” toggles orders given for own units. Move orders are shown using lines of different colors etc, attack orders as different lines etc.

“Line of sight” shows an outline around the unit which represents the are of the map visible to the unit.

“Superior commanders” shows the command chain from the unit up to its commanding officer and up all the way to the highest level. Use this to make sure that companies are kept close to their CO.

“Weapon ranges” shows the max effective weapon range for the primary weapon of the units. The range is shown as a circle around the unit.

1.2 Giving orders

To give orders to a unit you must first select the unit. Do that by clicking on the unit. It should then be highlighted and you should see some information about the unit in the unit info window.

The highlighted unit has a circle around it. If several units are selected one unit is the primary selection. Secondary units have a magenta circle around them.

When a unit is selected the right mouse button will show a popup menu with all available orders that all the selected units can perform. If one of the selected units can not, say, move fast, then that order is not available.

Various different features the help visualize the unit and its state can be toggled on and off. These include line of sight, organization, facing etc.

1.3 Minimap

The minimap is a little floating window that contains a miniature version of the map. It is normally in the upper left corner. A little rectangle indicates which part of the map is currently shown in the game area.

You can click anywhere in the minimap window to quickly scroll to that part of the map.

1.4 Action phase

This is the action phase of Civil. All the orders you have given your units are executed here and your role is just to observe what happens. You can control the flow of the time using the controls in the action window.

The right mouse button will bring up a context sensitive popup menu.

To end the action phase press Alt-e. Note that you can end the action phase at any time, you do not need to watch all of the action before ending.

1.5 Action window

The action window lets you control the flow of time. This is a little floating window that is normally in the lower right corner of the play area. It is not shown during the orders phase, only during the action phase.

1.6 Units

There are four basic types of units that make up armies in Civil. The units all have different behaviour and uses. Using all units correctly is the key to a victory.

1.6.1 Unit orders

A little window that always shows information about the orders a firendly unit has. Every order you have given to the unit is shown as a separate row in this window. Using this window you can get a quick overview of what the unit is ordered to do. This gets extremely useful if the map is cluttered or the unit has many “wait” and “change mode” orders.

1.6.2 Artillery

Artillery units make up the long range support units in the Civil War. They bombard enemies from a distance and are used to prepare the ground for an infantry assault or to scatter advancing enemies.

Artillery is slow and very vulnerable when attacked from the flank.

1.6.3 Cavalry

Cavalry units are horse mounted fast troops. They are often used for recon and rear are action. Due to their speed they can travel very fast over vast distances.

1.6.4 Headquarters

Commanders for various organizations are placed in headquarter units. This allows a brigade commander to freely move around the battlefield and to visit and rally regiment, battalions and companies. All organizations except companies have separate headquarter units. Companies have the company commander integrated in the company unit.

A headquarter unit can rally nearby routed and disorganized infantry just by its presence. Headquarters are also vulnerable and very likely skirmish targets.

1.6.5 Infantry

Infantry units make up the bulk of the force in a Civil War battle. They are the normal battling units.

When in combat infantry units will use battle formation, which is suited for getting every man in the unit to fire. While moving infantry units are in column mode, which is much faster but also much less suitable for combat.

Infantry is not very vulnerable, but not very effective either, unless the range is really short. Infantry units move slowly in battle formation and quicker when in column mode.

1.6.6 Unit info

A little window always shows information about the current selected unit. The window is by default in the bottom left corner of the map. It is a moveable window, which means that you can move it anywhere you want, and even hide it.

The information in the window includes the unit name, commander info, men and weapon status as well as various other stats.

If several units are selected then the main selected unit is the one whose info is shown. If an enemy unit is clicked only its name and type is shown.

1.7 Audio controls

The audio controls is a little floating window that allows you to select the background music that should be used. You can:

1. listen to the supplied background music by checking the checkbox "Included music".
2. listen to a normal audio CD. Just insert the CD into the CD-ROM and check "CD". You can use the controls at the bottom of the window to skip tracks, pause etc.
3. no music at all. Just check "No music".

1.8 Saving games

The game can be saved at any time during the orders phase, but not in the action phase. To save a game press Alt-s. This brings up a dialog asking for confirmation and a filename. Enter the wanted name and click "Ok" to save. Click "Cancel" to abort the saving. The game continues normally after a save.

1.9 Quitting Civil

To quit Civil just press the keys Alt-q. You will be asked for confirmation first. You may also want to save your game before quitting.

CHAPTER 2

Playing Civil

2.1 Overview

Civil simulates battles from the American Civil War. It focuses on smaller tactical battles, but bigger ones can also be played. The engine is very flexible and allows battles from many eras to be simulated.

Playing Civil is fairly easy, but requires some knowledge about the orders that can be given to units and how they do things. Apart from that, the user interface follows that of most other strategy games: units, targets and movement positions are all selected with the mouse, and keyboard shortcuts can be used to speed up playing.

This document goes through all the needed details for playing Civil, starting with how a game is created, to gaming details and saving games. The first part of the document outlines the needed steps for starting a new or a saved game, and the latter parts concentrate on the actual gameplay mechanics. A special section is dedicated to teaching some winning tactics.

Civil is a strategy game that simulates battles of the American Civil War on a company level, where a company is roughly 100 men (this can be less). Artillery batteries have a few guns each. The action in Civil is played in slow realtime. It emphasises strategic thinking and does not become a clickfest. The action is speeded up a certain amount, for instance five times. That means that one minute of player time means five minutes of simulated time.

Players give orders to units by clicking on the unit in question, then giving the order. More about this in the section. Orders are carried out as soon as possible, as command chains are also simulated.

2.2 Setting up

This chapter describes how to set up a Civil game.

2.2.1 Starting Civil

Civil is always run with the union and confederate as separate applications. It is either played as a human against an computer (AI) player or by two human players. The other player (either human or AI) can be located anywhere on the Internet or an Intranet. One of the human players always works as a server for the game, which in practise means just that the player must start his/her game first, and the other player will connect to the server player. The AI player can never be a server, it must always connect to a human played server.

To start the game issue the command:

% civil

If the game is not in the normal search path then the full path to the client will need to be given, such as:

% /usr/local/bin/civil

The client needs no commandline arguments, although the parameter –fullscreen or -f can be given to start up Civil in fullscreen mode. Fullscreen mode can also be toggled from the startup screen within the game.

If you use a Linux desktop environment that follows the Freedesktop.org recommendations then you have a menu entry for starting Civil in your normal menu hierarchy.

To start the game on Windows or Mac OS X just doubleclick on the icon that was installed along with the game. This will bring up the Civil window.

2.2.2 Basic information dialog

When Civil has been started a window is shown to the player. This is where the player can choose a nickname that is used in the game. This name can be anything, and default to the username on Unix machines. A checkbox also lets the player toggle between fullscreen and windowed mode. Fullscreen mode is not available on all platforms, and may silently do nothing without any harm. Fullscreen mode can also be toggled later from within the game.

The player can view the game credits by pressing the Credits button. This shows a new window with information about all developers and contributors.

Clicking Quit quits Civil.

The button Lounge takes the player to the lounge login. See for more information about the lounge. This currently does nothing as the lounge is not operational.

To proceed with the setup click the Ok button.

2.2.3 Choosing the opponent

This dialog is where the player must decide wether he/she will run as a server or client. If the player wants to run as a server the other player (or AI) will connect to him/her. If the player wants to be the client he/she must connect to another human player that runs as a server. A human player thus always acts as a server.

TODO: screenshot of Choose opponent dialog.

You can choose to play Civil against another human player or against the computer. The latter is the default, as you can see from the checked checkbox. When playing against the computer you do not need a network connection, as the AI client is run on the same machine as the player uses. See for more info on playing against the computer.

To proceed setting up the game click the appropriate checkbox and press Ok. Pressing Back takes you back to the first dialog (see),

If the player chooses to run as a client, then another dialog is shown after Ok is pressed.

TODO: screenshot of Client Setup

Here the player can input the name of the host where the server runs and the port number it uses. Do not change the port unless you're sure that the server actually uses that port. Pressing Ok in this dialog connects to the server and Back returns to the previous dialog. The client will wait after the dialog is accepted for the player running the server to setup the game, and then all data is downloaded from the server. This may take some time, especially over a slow network, as the full scenario is downloaded too. Note that the player running as a client can not do any setup of the game, all is taken care of by the server player. If Civil fails to connect to the server an error will be shown and the player is given the opportunity to correct any faulty parameters.

If the player chooses to run as a server he/she will be presented with a dialog where the port for the server can be set.

TODO: screenshot of Server Setup

No hostname is needed in this case. Pressing Back returns to the previous dialog and Ok makes the server start waiting for a new client. At this point the client player can connect, not sooner. After the client has connected the server player can continue setting up the game. See for information on how to set up a game.

2.2.4 Playing against the computer

Normally Civil requires two human players, one playing the union side and the other playing the confederates. There is also support to play against a simple AI client. To play against the computer the Play against the computer checkbox must be checked in the initial dialog. You are still required to enter a network port that the AI client will use to communicate with Civil. Accept the default port unless you have reasons to change it.

After you click Ok in the Server setup dialog the AI client will be started and you will proceed to setting up the type of game you want.

2.2.5 Starting the Ai client manually

It is also possible to start the AI client manually if you want to. The only difference is how the AI was started, when playing there is no difference at all. To do this choose to play against another human as the server. Click Ok in the Server setup dialog. Civil will now wait for the other player to connect. At this point start the AI client with:

% civil-ai [options]

If the AI client is not in the normal search path then you will have to give the full path to the application, like this:

% /usr/local/bin/civil-ai [options]

The AI client has no user interface of any kind, so all needed parameters must be given on the command line. The recognized options are:

–host=hostname which tells the AI client what server to connect to. This is a required parameter if the server does not run on the same machine as the AI client.

—port=port which tells the AI client what port on the given server to use. Default is 20000. This is a required parameter if the server you are trying to connect to does not use port 20000.

An example setup, where the server is located on server.example.com is shown below. The port is the default, i.e. 20000.

% civil-ai –host=server.example.com

To start the AI client on Windows doubleclick on the installed icon or select the AI player from the Civil menu.

The AI client can be connected as soon as a human player running as a server has been started. If started before this the AI client will print an error and terminate. When the AI client has connected successfully it will require no more human interaction. From the server's point of view the AI client looks just like a human client. In fact there is actually no way of knowing whether a client that connects to the server is a human or an AI, except by the fact that the AI player does not respond in an intelligent way to chat (see).

2.2.6 Game settings

Setting up a game involves basically two things: starting a new game or continuing a saved game. For restoring a saved game see.

TODO: screenshot of main setup dialog

To start a completely new game click the button New game. This will show the scenario dialog:

TODO: screenshot of scenario setup dialog

To select the scenario you want to play click the button Scenario. This will allow you to select a scenario among the installed scenarios. First you have to select the theater. This basically selects the time and place for the battle. You are first presented with a dialog where you select the theater by clicking in the correct checkbox and then click Ok. To cancel and go back to the main dialog press Cancel.

TODO: screenshot of scenario selection dialog

Once you have selected a theater you will be presented with all available scenarios for the selected theatre., as seen below:

TODO: screenshot of scenario selection dialog

Click on a scenario to select it. If you want more information about the currently selected scenario click the button More info. This will show a dialog with more background information about the scenario. When you're satisfied with the scenario selection press Ok to accept the scenario. You are now taken back to the scenario selection dialog. Here you may also need to select which side to play as.

Please note that you must agree with your opponent before starting the game as to which scenario to play and who will play as union/confederate. Civil currently has no facilities for letting players communicate about this matter, except the lounge (see for more information about the lounge). The AI or human client player will just be given the scenario and side that the server player selects.

2.2.7 Loading a saved game

Loading a saved game is done after having connected to a server. On the dialog that is shown after the connection is complete is a button: Load game that is used to load a saved game. When pressing this button a list of all saved games is shown. The saved games are stored by the server player.

TODO: screenshot of available saved games dialog.

Select a saved game from the list by clicking on it and then press Ok to load it. Pressing Cancel cancels the loading and returns to the main dialog. Note that both players must agree to continue the same saved game.

Note that in the actual game both players can choose to save the game at any time. If only one player saves a game then that player must also be the one that loads it, as only that player has the save game file.

2.2.8 Starting the game

When all is set up properly and you have selected a new scenario or loaded a saved game you are ready to start playing Civil. Press Start Game to begin. The server now sends the scenario and all other data to the human/AI client. This can take some time on a slow connection. If something goes badly wrong an error is shown:

TODO: screenshot of error dialog

At this stage the full scenario data is retrieved from the server, so it may take some time depending on the size of the scenario and the network connection speed. The progress bar in the dialog shows the progress of the sending/receiving.

2.3 Lounge

The lounge is currently not operational, it is an experimental feature that is currently without the needed server support!

The lounge is a separate module of the Civil system. It works as a separate optional server where players can do the following things:

2.3.1 chat

2.3.2 download new scenarios

The lounge can be entered by clicking the Lounge button on the first shown dialog. This will present the player with another dialog where the parameters for the lounge connection must be set.

TODO: screenshot of lounge setup dialog

Enter the name of the host where the lounge is running and the port it is using. This information must not be confused with the data for the normal game server. The lounge is a totally separate server application that is not run by default. See <xref linkend="lounge-server"/> for information about starting the lounge server. Any lounge server on the Internet can be used.

2.3.3 Lounge server

The lounge server is not started by default when starting an instance of Civil. It is an extra component that can be run by players that want to run it. To start the lounge server do the following:

% civil-lounge

OS X users should start the terminal application and type the above, ensuring the lounge is in your path.

Windows users; doubleclick the Civil Lounge Server icon or choose the appropriate entry from the Civil menu.

When the server starts it does not show any window or similar, it's a fully console based application. After the server has started clients can join the lounge.

2.4 Playing Civil

When a game has been setup and the scenario data has been downloaded, the main game window will be shown. This chapter describes the basic information needed to play a game, with references to more in-depth information in later chapters.

Turnbased gaming

Civil is a turnbased strategic game. This means that the player gives orders to the units in his/her army. An engine then resolves the orders for the turn and generates action. The game is thus divided into two separate phases:

orders phase where the player gives orders for a given turn to all his/her units.

action phase where the player watches the action for the current turn. The player can not affect the action at this stage, it is merely a movie that represents what actions the orders given resulted in.

This system is really a form of pseudo realtime, as the action is always resolved by the game engine second by second. This makes it very fair, as no player gets the "first go", and all kinds of cheating that can be done in a fully turnbased game are not possible. The player can use as much time as he/she wants for the orders, thus avoiding the stress associated with fully realtime based games.

2.5 Displays

The main display contains several windows above the playfield, which normally only shows only part of the whole game terrain, the actual battlefield is usually several times as big as the part shown by the map.

The battlefield in Civil is viewed from above. Heights can be seen as shadowed formations on the terrain. Units will be rectangles painted in dark blue and grey for union and confederate units respectively. A small triangle inside the unit shows the direction it is currently facing.

2.5.1 Windows

The floating dialogs give information about the current state of the game. One features an overview map of the entire battlefield, and a rectangle within it shows the portion of the battlefield that the map view currently shows. Clicking in the overview (mini-map) display moves the main map display to the co-ordinates selected.

The next window contains the game time and unit information. Each turn in Civil represents a number of minutes, typically 10, so advancing to the next turn increments the clock accordingly. During the action phase it also "ticks" to visualize how the action moves on.

Above the time display, is information that depends on the current state of the game. Usually, it is here where information is shown about the currently selected unit. If an enemy unit is shown the information is very limited, but for an own unit all available information is shown.

To the right is a window that shows the orders for the current friendly unit. For an enemy unit nothing will be shown. Orders to a unit as applied sequentially, i.e. in the order they were issued. In this window the player can see all issued orders for the unit.

At the top right of the screen Civil will show for messages to the player. These messages are usually information that Civil thinks the player should know. Messages that the other player may send (chat) are displayed here also. See for more information on chatting.

The messages will slowly scroll upwards until they will disappear. If a certain number of messages appear at the same time, causing the allocated space to fill up, then older messages will be removed immediately.

TODO: add screenshot of all the windows.

2.5.2 Map

The map display contains all the action within the game. All units that can be seen by your own units are also visible on the map. Your own units are always visible, but enemy units are only visible if at least one of your units can see it. This simulates the normal effect: fog of war. The map is always fully visible, as the commanders of both armies can be considered to have detailed maps of the battlefield, or local guides that know the environment.

Union troops are shown as blue rectangles with a yellow triangle on them, and confederate as grey rectangles with a red triangle. The triangle indicates the facing of the unit.

TODO: small screenshot with troops

The map consists of several types of terrain, all with different charateristics. Roads are easy to move troops on, while providing very little cover. Woods provide cover but restrict movement. See for more information about terrain effects. Learning the effect of the terrain is key to winning games.

On the map there are also drawn different types of lines. These lines tell the player what the units are currently performing; where they are moving and what their chain of command is. The lines are divided into three categories:

Movement orders which show which units are moving and where and how they are moving.

Attack orders which shows the current targets the units have. It is thus easy to see which units are attacking which enemies, and how.

Command control which shows the commanding unit for a particular unit. The player can thus easily keep related units close to their commanders in order to have a good command control. See for more information about command control.

Different colors are used for different types of orders. It is fairly important to learn what the colors mean in order to easily grasp the situation on the battlefield.

2.5.3 Toggling what to show on the map

Most of the features that are shown on the map can be toggled on or off as desired. This allows a lot of information can be presented if wanted, but all unwanted information can be turned off.

The list below contains the keys that can be used to toggle features on and off. All the keys toggle features, so pressing the same key returns the feature to its previous state.

1 toggles viewing of objectives. See for more info about objectives.

2 toggles viewing of labels on the map. Certain locations of interest may have labels attached to them that name the location. The labels are there to add more “feeling” to the map, and can be useful when the scenario is an implementation of an actual battle. The labels can be toggled on and off as needed.

3 toggles viewing of your units. Sometimes it may be of interest to view the map without the your units on it.

4 toggles viewing of enemy units.

5 toggles viewing of unit orders. Normally a line of a certain color indicates what action the unit is performing. If the display becomes too cluttered this display can be toggled using this key.

6 toggles viewing of commander links. Normally a line is drawn from the selected unit to its superior commander, and a line from the superior commander to its superior commander and so on. Click 6 to disable this feature.

2.5.4 Fullscreen mode

Civil can be executed in so called fullscreen mode which means that instead of running in a window as is default, the game runs fullscreen, obsuring your window manager. This can be comfortable when you are not montitoring any other windows or multitasking in any way.

To activate the fullscreen mode press F10. Press it once again to restore the previous setting.

2.6 Giving orders

To give orders to your own units you first need to active them. Activating a unit is as simple as clicking on its icon in the map. To select multiple units press the left mousebutton and drag the mouse (a rectangle will appear) until you have selected the wanted units. At that point release the left mouse button. The units are now selected. Note that one unit is always the primary selected unit, and it has a selection marker that is the same color as the one used when only one unit is selected. This affects a few things, such as moving groups of units. But more about that later on.

Clicking on enemy units will not activate them, but instead give you some short information about the unit. You can not do anything with enemy units by clicking on them. Note that the information presented to you about the enemy unit will most likely be inaccurate, as it is information that is approximated by your commanders who happen to see the enemy. The information is more inaccurate if the distance is large (try for yourself to identify the number of men in a mass of people 1 km away! ;-), but the validity increases as the commanders see the unit for longer and their distance apart shortens.

When you have activated a friendly unit all known information about it will be displayed in the status panel. There you can see for instance the number of men, morale, fatigue, etc the unit currently has. If the unit has movement and/or attack orders these will be shown in the map as lines.

Check up on this later and provide screenshots!

2.6.1 Setting a combat policy

To give the player control over when and how the targets are assigned to units, there is a feature called combat policy. A unit will get assigned targets based on what policy the player has set. There are three possible values for the policy that a unit may have:

> hold fire which means that the unit will never automatically get assigned a skirmish target. It just waits in it's current mode. The exception to this policy is when the unit gets assaulted by an enemy unit and it gets close. At that point a unit with a hold fire policy will override the policy and fire back to avoid being overrun.
>
> defensive fire only which means that the unit will skirmish back at units firing at it, but it will never initiate combat. The unit will target the closest and most dangerous enemy, preferably one in front of it. The same exception applies to this policy as for hold fire.
>
> fire at will which means that the unit is free to pick skirmish targets at will. It will not wait until it's being fired upon, but will start firing immediately that a suitable target comes within range.

To set the policy of a unit press the p key, or choose the menuitem change combat policy from the popup menu. This brings up the dialog below:

TODO: small screenshot of combat policy dialog

The previous policy is highlighted. Choose the wanted policy and click Ok to set the new policy. To set the same policy for many units at the same time select all the wanted units and do the same as above. Note that now there is no previous policy selected. Select the wanted policy and press Ok, thus setting the same policy to all units.

The dialog can be cancelled by pressing Escape at any time. If several units are selected, but no policy is chosen before Ok is clicked nothing will be done.

2.6.2 Help system

At any time you can press F1 to get a small help dialog that describes the currently available actions that can be performed. The actions vary depending on what you are currently performing.

TODO: small screenshot of help dialog

To close the helo dialog click Close or press Escape.

2.6.3 Popup menu

When one of your own units is selected, the right mouse button can be pressed to bring up a small popup menu. The menu lists all the available actions that can be performed with the current unit. Choose the wanted action from the menu.

TODO: small screenshot of popup menu

2.6.4 Ending orders

Once you are satisfied with the orders for the turn the orders phase can be ended. This means that Civil will calculate the action for the turn, and then present what happened to the player. See for more information about the action phase.

To end the orders phase select End orders from the popup menu, or press Alt and e. You will be asked for confirmation before the orders phase is really ended.

2.7 Action phase

This chapter explains how the action is visualized within Civil.

When a player ends the orders phase (see) Civil will calculate the action for the turn. The game will wait for both players to finish their orders before the calculations will be made. This means that if you were the first player to end the orders phase you will have to wait for the other player to finish his/her orders before the action can be calculated.

Once both players have finished the orders Civil will calculate what happened during the orders phase, and then send action data to both clients. The action data can be seen as a movie that shows what happened step by step. Internally Civil uses a high resolution for the time, so a single turn will result in movie with a lot of "frames".

You can control the movie playback and watch the action several times, maybe for instance for viewing some interesting part of the map in detail. To start viewing the action wait for the action to be computed and then received. From that point Civil will enter the action phase. During this phase only the action can be viewed, no orders can be given. To start the action press p or choose Pause from the popup menu. This starts the action, and will show one step of the movie each second.

To pause the action press p or choose Pause from the popup menu. The action phase can be paused at any time.

When you have seen all the action and want to review some part of it you can move to the beginning of the action by pressing a or choose Beginning of action from the popup menu.

To skip some parts of the action phase use the keys f and b to move forward and backward, or use the popup menu entries Forward and Backward.

When you have seen as much of the action as you need and want to start giving orders for the next turn, select End orders from the popup menu, or press Alt and e. You will be asked for confirmation before the action ends. Note that you do not have to view the whole action phase before ending. When the action phase ends new orders can be made in the orders phase (see).

2.8 Chatting

This chapter explains how chat is used within Civil.

The chat feature allows players to send messages between themselves. The AI player will not chat, it simply discards chat messages without replying.

Press the key 0 to bring up a small window above the panel.

TODO: screenshot with chat window active

Everything that is typed on the keyboard is echoed in the chat window. When Enter is pressed the message is sent to the other player and will be shown in the messages window of the panel (see). If you decide not to send the message for some reason, just press Escape to close the small window and resume normal playing. The game will continue to display actions in the background, although the player can not give any orders while writing a chat message.

2.9 Command control

Command control is very important in Civil. CC determines how well a unit is in touch with its commanding unit, i.e. leader. A unit that has a good link to its commanding unit receives orders faster and can thus react faster to the orders given by the player. A unit with a bad command control will receive orders much more slowly, and in extreme cases may not receive them at all.

Command control is determined by the distance to the commanding unit. A short distance means that orders can be sent to the unit faster. It is thus important to make sure that units always stay within a suitable distance from its commanding unit, and not break up (for instance) regiments into companies spread out all over the map.

TODO: screenshot with nice command control "lines"

In future versions of the game CC may be modelled more accurately considering the battlefield status. A unit that would be surrounded by enemies and have all routes cut off to the commanding unit would of course not be in command control, and thus its reactions would be totally dependant on its leader.

2.10 Units

This chapter describes the use of units in Civil. It describes the different stats available to units as well as the different types of units.

2.10.1 Types of units

A few different types of units are available in the game. Each are very different, and all types must be successfully utilized in order to win a game. The various types are listed below. Apart from only "primitive" types there are also the normal military organizations (see) such as regiments, brigades etc.

2.10.2 Infantry

This is the base unit of all armies. Without infantry you wouldn't be able to do much! Infantry units move by foot and are thus quite slow when compared to cavalry. Their speed can be increased by arranging the infantry units in column mode, which will increase marching speed at the cost of decreased combat efficiency. Infantry is often quite vulnerable without good protective artillery-fire when attacking.

2.10.3 Artillery

The artillery unittype works as a support unit and provides artillery-fire at selected locations. Artillery-units may attack from long ranges and still inflict considerable damage upon their target. Successful attacks often require a combination of artillery-fire and an actual attack by infantry. Artillery can be devastating when defending against assaulting close-range units, or it may totally fail. However, being under heavy artillery attack reduces the morale of the targeted unit considerably.

If artillery gets overrun by an attacker the outcome is usually disaster for the artillery. Keep them out of range of enemy assaulting units, or provide some infantry support. Artillery has a suppressing factor for the target as well as lowering the morale if the fire is effective. Attacking artillery from the flank is usually very effective, as it always takes some time (sometimes considerable) for the gun crews to change the facing of the guns in order to be able to fire upon a flanking enemy.

Artillery may only move when limbered. Short distances may be moved by hand, but this is very expensive when considering movement-points. Unlimbered artillery keeps the horses and other supplies behind the main line of the guns. When artillery is limbered the guns are mounted to special horsedrawn carriers, and can then be moved at the pace of infantry. Before the artilery may fire it must again be unlimbered and setup for combat. Limbered artillery is very vulnerable from attacks, as it is basically a very weak infantry-unit without its guns. It is very difficult to do a frontal attack on unlimbered and prepared artillery, and the morale of the attackers will often not suffice. Doing a flank attack on artillery is thus easier and will normally give better results.

2.10.4 Cavalry

Cavalry is the most mobile unit type, as they are horse-mounted. Cavalry is best used when attacking, especially for flanking maneuvres. Due to the use of horses cavalry has the best movement speed, especially when charging. Defending infantry may rout when being charged by cavalry, especially if morale or experience is very low.

2.10.5 Headquarter

This type of unit is the headquarter for all organizations above company level. So each battalion, regiment, brigade etc has a separate unit that acts as the headquarters unit for that organization. Headquarters that are directly superior to companies mostly affect that way the companies perform in combat, while higher headquarters mostly affect command delay. A nearby high headquarter can affect the way units perform in combat by improving rally efficiency and boosting morale. The player is supposed to be the highest commander of the entire army.

Destroying/disordering a headquarter would also affect the troops commanded by that headquarter.

In Civil headquarters are modelled by having some normal units “contain” the headquarter unit. When the unit suffers casualities there is always a chance of the actual commander being wounded or killed. In this case another leader is appointed from the same unit.

2.11 Unit stats

All units have some stats that affect their combat strength. Units might otherwise be strong but failing seriously in an important stat might make the unit quite useless. For instance it’s no good having a strong brigade of infantry which has no fighting morale at all, and who routes for every single battle. This chapter clarifies the various states units can have.

2.11.1 Men

This is simply the number of men that are ready for combat. Originally the unit has all men ready for battle, but combat of various types will wound and kill men. The strength is derived from those men who are ready for combat and who add to the overall combat strength. Resting units may transform some men from wounded to ready, and being out of combat may also do the same.

2.11.2 Morale

The morale of a unit directly reflects its willingness to fight and what it does when it takes casualities. Morale is always reduced when a unit takes casualities, retreats or otherwise has a bad day! It can also be decreased when other units visible to it take heavy casualities. It is increased after successful combat, i.e. when the unit manages to inflict damage on the enemy and it can also be increased when a neighbor does a ‘good fight’. Morale is thus a stat that can be spread out among units like a disease if bad battles are fought. The rallying skills of leaders are important when trying to keep the men in line.

The possible values are:

- very poor

- poor
- medium
- good
- extremely good

2.11.3 Morale when attacking

A unit that attacks another has a higher probability of breaking the attack if it has a low morale. They don't like the casualitites that they get and want to withdraw from the attack. A really bad morale level might send the attacking unit fleeing if the defenders stand firm or losses are taken. Good morale keeps the unit fighting longer, and it can even resist some casualities without halting the attack. A unit with high morale is usually one that has the best chances of doing real damange to defenders. In combination with a skilled and aggressive leader the ingredients for decisive victories are at hand.

2.11.4 Morale when defending

Units with bad morale are more likely to retreat when attacked, and when retreating they are more likely to rout. Units with strong morale will normally keep their lines together when forced to retreat, or withdraw rather than retreat, or retreat rather than rout.

2.11.5 Fatigue

Units gain fatigue when they do anything except rest! Every move they make will give them some fatigue, and battle is tiring. The units will lose fatigue by resting, i.e. doing nothing. It is wise to not drive the units too hard for too long, as constantly being fatigued and in battle will seriously effect morale. Switch the units in the front-line if possible and let units who've taken a beating rest behind the lines for a while.

The possible values are:

- exhausted
- tired
- normal
- rested

2.11.6 Experience

The modifier experience is a general skill modifier for a unit. A unit that has seen a lot of battle is usually quite experienced in the art of war. Rookie troops are not used to the battlefield and have higher chances of doing stupid things, and less experienced in handling the equipment (such as guns).

The experience of a unit is a direct modifier as to how good the unit fares in battle. A experienced unit is more likely to withstand heavy attacks without incurring high casualities, while an experienced attacker has a higher chance of breaking through and achieving decisive victories. For artillery skill is the most important stat, especially when firing from a distance, as it directly determines the chance of hitting the target.

This modifier also affects the melee skill of a unit and directly modifies how good the unit is when it comes to close combat. Melees are the result of assaults where the defender and attacker engage in hand-to-hand combat, and neither has retreated.

The possible values are:

- Green
- Regular
- Veteran
- Elite

2.12 Unit states

Each unit is always in a certain state. The state affects what the unit can currently do an how it reacts on different events. The state of a unit is changed by giving it various orders. The state is not directly for a unit, but is set indirectly by issuing various orders. All types of units can not enter all possible states.

2.12.1 Resting

This state is the resting-state for any unit. It lets the unit regain strength and reduce fatigue. Units will also get medical treatment, thus reducing the number of wounded men, generally increasing morale. The longer a unit rests the better the effect. A very short rest has no effect.

Applies to: all units

2.12.2 Normal

This state means that the unit is in normal combat mode and ready to perform attacks, assaults and defend against enemy actions. A unit in this state may move normally, but movement is more expensive than for the Column state, as the unit moves with all men ready for combat at any time. Consequently, fatigue levels increase faster.

Applies to: infantry

2.12.3 Column

This state is used when a unit is moved a long distance, and when enemy actions are not expected. The men are arranged in a column-mode and can efficiently use roads and paths, which makes movement for infantry cheaper in this state. The unit is however very vulnerable to attacks, as it does not maintain a high combat-readiness. Fatigue is not gained as quickly as when moving in normal state.

Applies to: infantry

2.12.4 Dug in

This state is the ultimate defensive state. The unit will prepare good defensive entrenchments and prepare for enemy attacks. It is very expensive to enter this state, and it generally takes a a lot of time, however, it is quite cheap to leave this state for the ‘normal’ state of the unit. Artillery in this state may fire normally, and is better protected from enemy artillery fire.

Applies to: all units

2.12.5 Limbered

This state is used when moving artillery. The guns are assembled in special carriers and horses are used to pull them. It is quite expensive to limber artillery, but moving in this state is as cheap as moving infantry in column mode. Artillery is very vulnerable to attacks in this state, and if the attack is close-range, i.e. comes from a neighbouring hex and is followed by assault, the artillery basically defends as a weak infantry-unit. It will try to unlimber guns and use them for defense if attacked, but this depends on the skills of the men and the leader, the morale and also fatigue.

Applies to: artillery

2.12.6 Unlimbered

This state is the standard firing state for artillery. The guns are ready to fire and the men are ready for combat. The unit may move while unlimbered, but it is very expensive, as the guns are basically pulled by the men themselves.

Applies to: artillery

2.12.7 Mounted

This state is the normal state for cavalry. This state is used when moving the units, as well as attacking. Cavalry is the only unit that is most efficient in attacking in its standard movement state. Other units may only attack at greatly reduced strength when in some movement state.

Applies to: cavalry

2.12.8 Dismounted

This state is used by cavalry when defending against attackers. Cavalry may move while dismounted, but there is generally no point in doing so, as mounting is cheap for cavalry, and moving when mounted is much faster.

Applies to: cavalry

2.12.9 Disordered

This state is entered when a unit retreats and the skill of the leaders and the unit is not enough to keep the unit retreating in an orderly manner. The morale is higher than for routing units, the men are just a bit disordered, something usually quite simple to sort out. Disordered units have a defensive penalty when attacked and may not perform most missions, such as attacks. Units recover from disordered states by rallying.

Applies to: all units

2.12.10 Retreating

A retreating unit will back away from the enemy in a somewhat ordered fashion. A retreating state may be entered by manually ordering the unit to retreat, or if it is forced to retreat due to enemy attacks. Depending on the rally skill of the leader it may take a while before a retreating unit is restored to normal. A forced retreat often means that the unit has suffered more severe losses than when manually ordered to retreat, thus it will take longer for units on forced retreat to rally.

Applies to: all units

2.12.11 Routed

The ultimate state of disorderedness. The unit is fleeing from the attacker. Requires leader to rally the unit, and this may not always succeed. The unit needs to get its morale up, maybe resting for a while to regain it.

Applies to: all units

2.13 Commanders

Each unit has one immediate leader and any number of leaders above that. The immediate leader is the one that has most influence over what the unit does, and is thus most responsible for outcomes of battles. A leader has a few important stats that very much influence the outcome of battles.

2.13.1 Battle skill

A leader with a low skill level is bound to make bad decisions, while a skilled leader may make extraordinarily good decisions when needed. The skill level is important when doing anything! It's very important when doing assaults, attacks or other actions that may have big changes. Skill directly modifies all results. The battle skill is more or less the skill taught to the leader in military school, i.e. theory. The practical knowledge is experience. Together these directly modify what a leader is capable of doing.

The possible values are:

- Very poor
- Poor
- Medium
- Good
- Extremely good

2.13.2 Rally skill

This skill is used by leaders when they try to keep the 'spirits high' and avoid having the unit retreat under battle. It also reflects the leader's ability to rally the troops after a retreat and again create a unit ready for fighting. When the unit takes casualities this stat is very important.

The possible values are:

- Very poor
- Poor
- Medium
- Good
- Extremely good

2.13.3 Experience

All leaders have more or less battle experience which influences the descicions made in battle. A leader may have low battle skill, but has fought many battles and built up much experience, which means that in order to be a successful leader, experience is often quite sufficient. A leader with good battle skill and experience is often virtually unbeatable.

The possible values are:

- Very poor
- Poor
- Medium
- Good
- Extremely good

A leader's experience is different from the experience of the unit the leader commands. An inexperienced leader is more likely to make bad decisions, such as ill-informed attacks, unnecessary retreats etc, while an experienced leader knows the battlefield better. The experience of the unit only determines how the unit performs in executing the actions dictated by the leader.

2.13.4 Personality

A leader has a special personal style which influences the way the troops are led. An aggressive leader has a much higher chance of doing radical stuff, such as assaults and pursuits when enemies have failed and are retreating/routing. A careful leader extremely rarely pursues when the enemies are just pulling back, but is more likely to when they are retreating or even fleeing. This may cause him to miss some good opportunities but instead save him from hastened descicions that may lead to serious losses, such as a pursuit of a strong enemy that turns into a trap.

The possible values are:

- Careful
- Normal
- Aggressive

An aggressive leader that is very inexperienced can lead to some major disasters for a unit.

2.14 Combat

This chapter explains the possible forms of combat that can take place within Civil and how they affect the parties involved.

2.14.1 Skirmish

A skirmish is the basic form of combat in Civil. Skirmishing means that the unit fires constantly at a target in an organized manner, i.e. when ordered.

A skirmishing unit will fire in a cycle. How fast a unit fires depends on its stats, such as the skill and morale. After a volley of fire the unit will reload and reorganize its lines for the next volley. The unit will continue firing at the enemy as long as the enemy is visible, in range and the unit itself can fire. A unit will stop skirmishing if it gets disorganized or is forced to retreat or rout. It will also stop firing if its ammunition supply is running low.

2.14.2 Attack

An attack means that the unit advances against the defending unit. It will advance in a slow march in combat formation while periodically firing against the enemy unit.

Doing an attack instead of just a skirmish may lead to the enemy unit breaking up and retreating. It is thus a good idea to attack an enemy unit if their position is wanted, or if the enemy needs to be driven away. As the attacking unit advances slowly the attack can be aborted easily with less risk compared to an assault (). If a fast advance or maximum damage is important, an assault is probably a better choice.

Attacks can only be performed by:

- infantry in formation mode
- mounted cavalry

2.14.3 Assault

An assault is the most devastating form of attack, as it means the attacker storms the defender and attempts to overrun it. When a unit is ordered to assault an enemy it will start a fast march against the position. When the unit reaches a certain distance it will stop and fire a few volleys against the enemy, before eventually storming the last distance and engaging in hand-to-hand combat. Either or both units usually end up disorganized after the hand-to-hand combat, and usually one of the units retreats or routs.

The assaulting unit will not run full speed when assaulting, as that would make the troops too tired. Instead they use a fast march/slow jog that keeps the lines intact and makes organized volleys of fire possible.

Many assaults will never have the final storming stage, as the exchange of fire at close range is often devastating, and may lead to either unit breaking up and retreating or routing. If neither unit gets disorganized the assaulting unit will storm the defender after a few volleys of fire. This leads to a battle with large casualities.

Assaults can only be performed by:

- infantry in formation mode
- mounted cavalry

Artillery can not engage in an assault, as it is simply not possible to drag the heavy guns and do something useful. Artillery can support assaulting units by skirmishing with the target (i.e. fire at it).

2.15 Organizations

This chapter describes the organizational layout of the troops in Civil, the normal sizes, types of leaders an so on. The definition “organization” in this context means the normal historical hierarchical military organizations such as companies, regiments, brigades, divisions, corps and armies. These are partially included in Civil to make it more meaningful to simulate command control, leader influence and command delays.

The actual “seen” units on the map are always companies. Some companies can contain the needed organizational leaders for higher organizations too. So there can be a company that has the command for a regiment or brigade, but still show up only as a single company. These units are worthwile targets, as killing or capturing the important commanders disrupts the entire organization. A lost commander in a regiment will be replaced, but usually with a commander of lower rank, and often also of worse quality. Companies with commanders for higher organizations should therefore be protected from heavy casualities, and should be kept near the center of the organization they command, in order to minimize the distance to all other suborganizations in its command.

Seen as a graph the command structure builds up a tree with the root in the highest commander on the battlefield (such as a division commander) and spreading out to lower organizations. The sections below will describe the normal organizations found on the battle field and their characteristics.

2.15.1 Companies

Companies are the basic fighting force of the Civil, and are the only really visible organisations. Infantry, cavalry and artillery are formed into companies, although for artillery and cavalry the real name would be batteries and troops. Infantry companies consist of up to 100 men, but typically much less due to losses. Few companies were historically up to strength, the idea was to instead add reinforcements as new organisations (regiments, brigades etc) instead of bringing older ones up to full strength. Artillery comanies (batteries) consist of a few guns (typically up to 6) along with their crews. Cavalry companies also consist of up to 100 men along with their horses. For more information about the different types of units see

Each company has a commander which affects how the unit behaves.

2.15.2 Battalions

Battalions are organisations consisting of a few companies, typically two to four. Battalions are not always used to make up regiments. A battalion always has a commander.

TODO: figure with typical regiment

2.15.3 Regiments

TODO: figure with typical regiment

2.15.4 Brigades

TODO: figure with typical brigade

2.15.5 Divisions

TODO: figure with typical division

2.16 Objectives

This chapter describes how objectives are used in Civil.

What are objectives

Objectives can be present in a scenario to mark out certain parts of the map that are especially important for some reason. An objective can be an important terrain feature, such as a bridge, a hill, a ford or a road crossing. Objectives are something the player should try to capture, as victory points are awarded to the player who controls objectives at the end of the game.

On the map, objectives are shown with these symbols:

TODO: small screenshot of objective on map

There are two kinds of objectives. The golden objective is a primary objective and is worth more victory points than the silver one See for more information. A scenario may not always use both objectives types, it may not even use objectives at all, if it does not fit into the scenario type.

Objectives are captured by making sure that no enemy units are within a certain range from the objective, and that at least one friendly unit that has not disrupted or routed is near the objective.

2.16.1 Victory points

When the game ends victory points are awarded for each objective the player holds. An objective that is not held by any player is considered to be neutral, and does not award either player victory points. There are two kinds of objectives:

primary objectives are worth 500 points. secondaryobjectives are worth 250 points.

A primary objective is thus worth double the points awarded for secondary objectives. Points are only awarded at the end of the game.

TODO: maybe objectives should award some points for each turn a player holds them? Could maybe avoid the "final turn objective rushes" a little bit?

2.17 Reinforcements

This chapter discusses how reinforcements work in Civil.

Reinforcements are troops that arrive late to a battlefield. Usually most troops are already present at the location for the battle, but some troops may arrive later during the day, or over the course of several days if it's a long battle. Not all scenarios have reinforcements, it's up to the scenario designer to decide if reinforcements are available.

Reinforcements arrive at some location very near the edge of the map, usually in the rear. If several units arrive at the same location (for instance a road) the first ones to arrive may be placed a bit from the edge to give space to the others. As soon as units have arrived on the map they may be controlled in the same way as any other unit.

The time and location where reinforcements arrive is decided by the scenario designer; the player can in no way affect the arrival times.

2.18 Saving a game

This chapter explains how to save games.

Saving games in Civil is very simple. The game can only be saved during the orders phase, not the action phase. To save the game, press the keys Alt and s. This will bring up a dialog where you will be asked wether the game should be saved or not.

TODO: screenshot of question.

Click Ok to save and Cancel to abort the saving. If you chose to save the game another dialog will be shown where you can enter a name.

TODO: screenshot of input dialog.

Press Ok or Enter to accept the name and save the game. If the given name was empty, ie. nothing was written then the saving is aborted. The actual saving process will take some time, so please be patient.

Civil saves games in a directory $HOME./civil/saved_games. The saved games are available for loading from the main dialog (see).

2.19 Scenario management

This chapter explains how scenarios are managed.

This chapter is still very much vapour!

2.19.1 Adding scenarios globally

All scenarios in Civil are managed by the player that acts as the server. The client player only downloads them from the server when a game is started, and does not store them locally in any way. So players can play a game where only one player has a scenario by having that player act as the server. To add a new scenario to a server the server must be made aware of the new scenario.

The first step is to download the scenario from somewhere, and then copy the files (two XML files) into the scenarios directory of the Civil installation. This is typically:

Unix: /usr/share/games/civil/scenarios/ or /usr/share/games/civil/scenarios/. TODO: check these paths.

Windows: c:Program FilesCivilscenarios

After this Civil can be restarted with the player that just installed the new scenarios acting as the server. The new scenario will be selectable from the list of scenarios when a new game is started (see).

Scenarios installed in this way are available to all users that can play Civil on the host machine. To install a scenario for only one player see the next section.

Windows NT, 2000 and XP users should check they have the correct permissions to access the installation directory and run civil. In extreme cases your system administrator can deny users full rights to the installation directory, in which case consult with your local expert on how to correctly setup Civil for multi-user play.

2.19.2 Adding scenarios locally

Scenarios can also be added so that only the local player can use them. Do do this follow the instructions above, but replace the path /usr/share/games/civil/scenarios/ with the path $HOME/.civil/scenarios/. Now the downloaded scenarios are only available to the player who installed them.

2.19.3 Getting scenarios from the lounge

Scenarios can also be downloaded from the lounge. This is done in a fully graphical way, and the scenarios can be downloaded from any lounge on the Internet. Scenarios downloaded in this way will be saved locally for the player downloading, just like in the section above.

2.19.4 Using the tool civil-scenario-admin

Civil can also download scenarios from scenario repositories through a scenario server. To do this use the tool civil-scenario-admin that is not yet written, and thus is very much vaporware. This tool allows the player to conenct to scenario servers around the world, check for new scenarios and download interesting ones. Note that these servers have nothing to do with the normal game server, i.e. civil-server, which is used when playing the game. The scenario server just manage scenarios and let players browse scenarios.

To use the tool civil-scenario-admin start it like this:

% civil-scenario-admin –server=server.example.com –port=8080

After it has started it will show a simple menu which allows you to list scenarios that are available for download from the server and to download one or more of the scenarios. When a scenario is downloaded it is automatically added to the scenarios that the local game server can use. No extra configuration is needed.

2.19.5 Scenario server

You can also run your own scenario server if you want to share your scenarios with people all around the net. To do this you need to use the server civil-scenario-server that performs the sharing. To start the server run the following command:

% civil-scenario-server

This will launch the server and make it listen on port 8080. It will then share out scenarios to players that connect to the server using civil-scenario-admin. Clients can get listings of the scenarios your server shares and optionally download one or more of them.

By default the server shares out all the normal scenarios that it can find, i.e. the same that the normal game server would use.

2.20 Frequently Asked Questions

2.20.1 No keys seem to do anything?

This one is weird. Probably a bug somewhere in our code. The solution seems to be to make sure that Num Lock is not pressed. Having it pressed (lit) seems to lose all keyboard input.

Some Apple keyboards (TiBook, PowerBook etc.) require a qualifer key: FN to be pressed in order to access Functions keys. Don't forget this during the game! If you still have difficulty under OS X, check your Pygame installation. There have been known problems with keyboard input under Fullscreen mode in OS X that have been fixed by reinstallation of Pygame.

2.20.2 Is Python 2.1 hardcoded? Can I change to use Python 2.2 instead? Or any other version?

Sure, check the switches to ./configure. Do this:

% ./configure –with-python=/path/to/my/python

2.20.3 After installing Civil under Windows I get a weird error when I try to launch it: KeyError: HOME

Currently, the Civil installer doesn't configure a HOME environment variable, which is something the game needs in order to save games and store scenarios. The document: README.WINDOWS explains how to configure this for most versions of Windows.

2.21 Licenses for used software

Civil uses a few software packages that are integrated into the source tree. The licenses for these are here.

xmlrpclib

The xmlrpclib library is

CHAPTER 3

History of Civil

3.1 Civil on SourceForge (2000-2004)

The original Civil project was registered on SourceForge on 2000-08-31 with the aim to create a real time strategy game about battles in the American Civil War allowing to play over a network. It was programmed in Python using PyGame as game framework and some Python C extensions for speeding up critical parts. It was published under the open source GPL-2.0 license. The game program code was stored in a CVS version control system.

- Last release: Civil 0.83
- Authors: Jan 'Chakie' Ekholm, Gareth 'The Corruptor' Noyce, Michael 'mikee' Earl, John Eikenberry, Marcus Alanen, Cristian Soviani, Uwe Hermann, Frank Raiser, Mike Szczerban, Jeroen 'slm' Vloothuis, Kalle

3.2 Civil on GitHub (2018-)

On 2017-12-21, Trilarion created the Civil project on Github as a continuation of the Civil project on SourceForge. The sources of the last available release (0.80-0.83) as well as a snapshot of the CVS repository were stored in a git repository on GitHub.

No release has yet been made.

CHAPTER 4

Installation

Civil requires the following libraries and apps

- Python 3
- Pygame
- PyQt

CHAPTER 5

Development

5.1 Line-Of-Sight Calculation Algorithm

This file describes the line-of-sight calculation between units, and is only of interest to developers.

5.1.1 Theory/model

The board is composed of triangles, each of which has a terrain type and an altitude/slope. Each terrain type has a 'visibility stack'; this is essentially a list/function that tells how opaque it is at a given altitude.

I.E., all terrain types are 0% transparent at/below ground level. Most types have some obscuration just above that to some height (due to ground cover, grasses, trees) and are 100% clear at higher altitudes. Note that these values are for looking through an entire hex worth of that material, and thus have to be adjusted down for less distance.

The 'base visibility' between two (3D) points is inversely proportional to the square of the distance between them (the light/apparent size would drop off as the square of distance), less whatever (cumulative) attenuation is caused by terrain obscuration; if I look through two full hexes of 80% visibility, it reduces visibility to 64%.

Why triangles and not hexes? Three reasons: 1) Math is much easier with triangles. 2) Some hexes appear to be composed of multiple terrain types, and we can approximate more closely by subdividing into triangles. 3) If we break into triangles, we can give each a constant slope and have the map be continuous.

5.1.2 Computation

We take our two points, and assume the watcher/target to be 3M tall. Consider the 3D line between them; this is the literal LoS. Taking just the 2D (top-down) view of that line, we figure out all the triangle edges it crosses, in order. The line segments between consecutive pairs of these points each cross a single triangle.

Now, start with the base visibility (constant / distance ^2). We then run through all the triangles we cross, figuring out how much they obscure the view. Well, how much is that?

First, we figure out which triangle we're crossing; take the midpoint of the segment and it's in that triangle. Using the 3D version of the LoS vector again, take the worst (most obscured) altitude for the segment, which will be the lowest, and (because the triangles are flat) will be at one of the two points where we enter/exit the triangle. Compare that to ground level at that point to figure out the height above ground level, then look that up in the terrain visibility 'stack' for the terrain type in this triangle.

Now, we have to adjust the visibility loss for the distance covered by that segment - this is a power thing, rather than linear, because of the way we multiply visibility together. (For instance; if looking through a full hex-width of forest gives you 50% visibility, looking through half that gives you ~70%, because two half-hexes would be 70%*70% = 50%.) Possibly clever things with logarythms might make all this power-fiddling into addition, but that makes my head hurt.

Anyhow, you now multiply the base visibility by all of the segments' visibility, and that's your result. Ta-Da! Being able to see further from hilltops, and not see through hills etc, should fall out automagically if the altitudes and terrain data are set up sensibly.

- mikee

5.2 Layer Refreshing Algorithm

This file describes the layer refreshing system of the game, and is only of interest to developers.

5.2.1 Background

The various panels in the game are called layers, which are created on game startup and never destroyed during game-play. Instead, they are set visible or invisible.

Layers can be made moveable by subclassing FloatingWindowLayer instead of Layer. All Layers can be partially transparent

The refreshing algorithm supports/will support refreshing the internal information, a part of the layer, or the whole layer. The main logic is in playfield.py/paint(), and the redraw itself happens in each layer's paint() method. If a layer wishes to use a border, it can do so by calling self.paintBorder() in its paint() method.

When in doubt, the layer can always update everything.

5.2.2 Internal Refreshing

So what is internal refreshing? Internal refreshing means that the layer is in a _known good **state_** from a graphical point of view. This requires that a) no other layer has partially covered it, and b) the layer itself chooses to use the internal refreshing system.

If either of these are untrue, a full redraw is issued, otherwise the layer can update only the parts that have changed from the previous time. In practice this means knowing what a layer has drawn the previous time, usually using variables.

5.2.3 Refreshing

5.2.4 Usage

Each layer has the variables * self.x * self.y * self.width * self.height * self.need_internal_repaint (readonly)

The four first describe the bounding box of the layer (without the surrounding border), and must be updated accordingly, before asking for a refresh.

A graphical refresh is requested with * playfield.needInternalRepaint (layer) * playfield.needRepaint () * playfield.needRepaint (pygame.Rect)

needInternalRepaint() asks for an internal update, needRepaint () requests a complete repaint of the whole playfield, needRepaint (pygame.Rect) adds the given rectangle to the list of dirty areas that need to be updated, and can be used to add as many areas as one wishes.

The repaint is issued with playfield.paint(), which contains all the logic needed to issue the right orders to all layers.

The variable need_internal_repaint can be read in the layer's paint() method to check if the current context only requires an internal update.

It is always safe to redraw everything in the layer.

There is no way to make a part of the layer transparent in the paint() method, you're too late and should've called playfield.needRepaint (), or even better, playfield.needRepaint (layer.getRect()) to make the layers below refresh.

5.2.5 Current Issues, Specific to Civil

- The terrain can't paint the dirty areas, it repaints

everything. Slooow. Current suggestion is to have a copy of the current map in a surface, and blit straight from the there to the appropriate dirty area(s).

- Some layers *don't know what they are supposed to paint*,

until they get called in the paint routine. This makes unit layers, weapon range layers etc a bit slower than needed, since unneeded refreshing of everything must be done.

- Sometimes a refresh is issued even though there is

no need to, or it could be optimized to only cover a certain area. This goes away slowly

- After a "unsuccessful" select, two refreshes are

issued. (around half a second on my computer…)

5.3 Bugs

OPEN BUGS

Units seem to be able to run through water. The internal terrain data is apparently not ok?

Retreating, routed and moving units will stop and drop all their orders when they run into water. Maybe the retreating/routing units should get some other place to move to?

Ending screen looks horrible. The font is too large.

The weapon description text in the unit status window can overflow.

OPEN OLD BUGS

Some in-game dialogs can be activated twice. For instance if the player presses F1 twice in a row then two HelpBrowser states are activated. Closing the help browser dialog still leaves the game in the second HelpBrowser state, but without any visual clue as to what is going on. Hitting Escape closes the second HelpBrowser state too. States should check for this? UPDATE: At least HelpBrowser can't be activated twice, see state.py how I "fixed" this… –msa

The "wait" cursor used in the setup screens lacks the white color. The cursor files seem to be somehow wrong. Need to check the cursor and fix. It works, but is mostly invisible…

Changing size (to smaller at least) leaves some artifacts on the right border. Seems like parts of hexes? - Yes, correct.

Noticed a weird combination of ending the turn and immediately dragging windows. It mixed up somehow, the action phase play buttons came available but I still had to end the turn again. Will test more later.

“full-screen” gfx pictures must take into account window size => center gfx, black borders around it? (e.g. take big resolution, surrender game)

Exiting Civil from the front dialogs via the window manager can leave the AI client running.

MSA’s NITPICKING SECTION (can be considered as “enhancements, not bugs”, sometimes of questionable value. . .)

G’s Nitpicks:

Running Civil at a lower res means the Game Over screen appears at the lower res. As such, buttons and info is positioned off screen. . .

ON HOLD FOR NOW

Clearer visual signs when we are in action mode, and when in “real” play mode. Clearer visual signs how to end turn, and how to end action phase. - what can we do about this? WONTFIX

FIXED BUGS

Compiling the C LOS extension should be done using distutils. FIXED

The server and AI client will be left running when the human player exits the game. The client should send out some kind of “End game” message? FIXED

Retreating state-machine is wrong. Still refers to “retreatinginfantry” etc. FIXED

engine/ai/rally.py will rally units that are disorganized. But it doesn’t know wether the unit should be, say, limbered or unlimbered. Thus we should maybe have “disorganized_limbered” instead of “disorganized_artillery”. FIXED

Better modes for units. A few of the current ones are a bit silly. Instead of having “meleeing_infantry” we should have “meleeing_formation” and “meleeing_column” so that we know the original mode of the unit, and what mode it should get after the melee is over. Same goes for a lot of modes. FIXED

Clicking “Load game”, then “Cancel”, followed by “New Game” and “Scenario” results in no scenarios being listed. . . FIXED

The input dialog in-game crashes. At least when I tried to change the name of a saved screenshot. Should be a simple fix. FIXED. Requires Pygame 1.5.3+

The in-game dialog “Change combat policy” keeps drawing over itself over and over again, making the text thicker and thicker. It should clear the area before drawing.

Game loading should work. Will try before 0.80 milestone. FIXED. Old saved games are invalid though. Or you need to fix to .

Some dialog layers seem to assume the screen is 1024 pixels wide, as they use the magic constant 512 for calculations. Will break when the screen size is changed. FIXED

The flag scenario.playing is not handled in event_loop.py => Civil will never actually quit when doing an “Quit Civil”. FIXED

LOS doesn’t work at all? Try scenario10, move unit to the west where the enemies should be. Nothing is displayed. - this should work now. There was a missing unit.calcView() that meant that

> units never recalculated what they saw.

Num Lock may not be pressed in-game. Nothing works if it is. Strangely all works during setup even if it's selected. FIXED

During action phase unit info window does not update the info. Needs to connect a signal. FIXED

Changing combat policy dialog has "wrong offset" when clicking the checkboxes. FIXED

Choosing units doesn't work properly when the map has scrolled away from (0,0). Some offset bug… FIXED

The little yellow rectangle in the minimap is too small. It does not take into account the fact that the panel is no more. FIXED

Help browser shouldn't just raise & quit if a help file can't be parsed. (or some section is missing) FIXED

5.4 Scenario format

This appendix describes the format of the scenario files. The files are based on XML files and fairly easy to modify by hand if needed.

The scenario files the Civil uses are normal Zip files that contain two or three files:

scenario.xml which contains the full data for the scenario, along with the map, all unit, objetcives, general info etc.

info.xml which contains the contents of the tag >scenarioinfo< extracted in a separate file. This is for eacy access so that the whole scenario.xml does not have to be parsed when info about the scenario is needed.

los.pickle which is line-of-sight data in pickled form. This file does not need to be here, as it is merely a cache for information that can be recalculated when the scenario is loaded.

The scenarios are compressed with Zip as that makes them smaller and it is also a convenient way to merge several files into one archive. XML as a format is easy to use and makes the scenarios structured, but it also makes them large. Compressing them makes them smaller, thus conserving a lot of disk space and making them a lot faster to load over a network if needed.

The following sections will list all the tags in the XML file and describe what data they contain. All the tags are inside the mandatory toplevel tag <scenario>, with a normal XML header.

This tag is mandatory. It gives some basic info about the scenario, such as textual descriptions och the scenario and the missions, as well a the date and length of the battle. The data in this tag is copied verbatim into the shorter foo-info.xml file. The contents of the <scenarioinfo> is shown to the player when selecting a scenario.

This tag contains the name of the scenario. It is a string that is only shown to the players, it is not used internally in Civil.

This tag gives a one line description of the location of the battle. It is shown to the players only and not used by Civil internally. This should be the geographical location of where the battle took place, such as Gettysburg.

This tag gives the starting date and time for the start of the battle. It has no other data apart from five attributes: year, month, day, hour and minute. Each attribute has a numerical value. A 24 hour clock is used for the hour. This data is used by Civil.

This tag gives the current date and time for the start of the battle. For a new battle this is the same as the starting date, but for a saved game this is the current date at the time of the save It has no other data apart from five attributes: year, month, day, hour and minute. Each attribute has a numerical value. A 24 hour clock is used for the hour. This data is used by Civil. The game is over when the internal clock passes this date.

This tag gives the ending date and time for the start of the battle. It has no other data apart from five attributes: year, month, day, hour and minute. Each attribute has a numerical value. A 24 hour clock is used for the hour. This data is used by Civil. The game is over when the internal clock passes this date.

This tag describes the scenario. It gives a general description of the scenario and can contain information such as historical or fictional info that led to the battle, or other background info. For a historical scenario it could also contain info about that happened in the real battle.

The actual data is any number of <para> tags. The <para> tags are paragraphs of the description, and there is normally least one such tag, unless the scenario has no description.

The data in the <para> tags is only meant for viewing by the players, it is not parsed by Civil in any way.

This tag describes the missions for the rebel and the union player. It consists of two subtags: <rebel> and <union>, and they in turn have any number of <para> tags. The <para> tags are paragraphs of the description, and there is at least one such tag in both <rebel> and <union>, unless the side has no mission description.

The data in the <para> tags is only meant for viewing by the players, it is not parsed by Civil in any way. The mission descriptions should brief the players as to what their mission in the scenario is.

The data in this tag describes all the weapons used by various units in the scenario. This tag contains any number of <weapon> tags, which in turn has a number of attributes that describe the weapon:

- id which is the unique id assigned to the weapon. Units refer to this id when their weapon is described ()
- name which is the name of the weapon, such as Rifle. Not used by Civil.
- type gives the type of weapon. The current types available are rifle and artillery. These affect how the weapon is used.
- range gives the maximum range of the weapon i meters.
- damage gives a value that describes the damage done by the weapon when it hits an enemy unit.
- accuracy which gives the accuracy of the weapon at a given range. TODO: what range?

All weapons need not be used by units, but all weapon id:s that units refer to must be present.

This tag contains all the unit data. The units are grouped inside the two subtags <union> and <rebel>. Full unit organisations are used. The data in the <union> and <rebel> tags is identical, the two tags are only used to give the owner of the organisation. No other tags nor attributes are used.

The units are always organised into brigades. Each brigade is organized into a number of regiments. A regiment has either a number of battalions or companies. A battalion contains companies.

Brigades, regiments and battalions do all have a <headquarter> tag. It defines the data for the commander of the organisation, and it looks the same for all organisations. Each organisation thus must have a headquarter, regardless of how man suborganisations it contains. Companies do not have headquarters.

A <headquarter> tag basically defines the same data as a company (), but with one added tag, namely the

> <commander> tag. All other tags are same for units.

The <commander> tag defines the data for the commander in a headquarter unit. The headquarter unit is merely a wrapper around the commander, and contains some lower officers. The <commander> tag has a name attribute which is the human name of the commander. Additionally it has a few subtags which define properties for the commander.

- rank defines the name of the rank of the commander in a human readable form, such as Captain. It has no attributes.
- experience defines the experience of the commander. A more experienced commander makes less mistakes.
- aggressiveness defines the aggressiveness skill of the commander, ie. the likelihood of the commander doing something radical.
- rally defines the rally skill of the commander.
- motivation defines the motivational skill of the commander.

- All the above tags except rank have an attribute value and no main data. The attribute contains a numerical value for the modifier that it represents.

Brigades are the largest forms of organisations in Civil. Every other organisation is a part of some toplevel brigade. The only visible part of a brigade is the headquarter unit.

A <brigade> tag has the attributes id which defines a unique id for the organisation, and a name which contains a human readable name for the organisation.

Several regiments build up a brigade. The only visible part of a regiment is the headquarter unit.

A <regiment> tag has the attributes id which defines a unique id for the organisation, and a name which contains a human readable name for the organisation.

Several battalions build up a regiment. Battalions are however not a mandatory organisation, as a regiment may not have any battalions at all, instead it can contain companies. If battalions are present in a regiment, the battalions contain all the companies of the regiment.

The only visible part of a regiment is the headquarter unit.

A <battalion> tag has the attributes id which defines a unique id for the organisation, and a name which contains a human readable name for the organisation.

Companies build up the bulk of the visible units in Civil. A company is the lowest level organisation available, in Civil, and it has no separate headquarter unit.

A <company> tag has the attributes id which defines a unique id for the organisation, a type and a name which contains a human readable name for the organisation. The type can have one of the values infantry, cavalry and artillery, and it defines the type of company. The <company> tag also contains these subtags:

- commander which defines the commander of the company. A company has no separate headquarter unit and thus has its commander in the unit itself. See <xref linkend="scenario-tag-units-headquarter"/> for more info about the <commander> tag.
- pos which defines the position of the unit. It has the attributes x and y which give the pixel coordinate, and no main data.
- men which defines the number of men in the unit. The attributes ok and killed give the number of men that are eligible for combat, and the number of killed men. The tag has no main data.
- facing which defines the facing of the unit using an attribute value. It has a value from 0 to 35. The tag has no main data.
- morale which defines the morale of the unit in an attribute value. The value is a numerical value. The tag has no main data.
- fatigue which defines the fatigue of the unit in an attribute value. The value is a numerical value. The tag has no main data.
- experience which defines the experience of the unit in an attribute value. The value is a numerical value. The tag has no main data.
- weapon which defines the main weapon for the unit. It has the attributes id, ok and destroyed. The two latter give the number of weapons that are still useable and the number of weapons that have been destroyed. The id refers to a weapon as given in the <weapons> tag.

Optionally a company may contain a tag <arrives> which indicates which turn the unit will arrive at the battlefield. The tag has no data, only five attributes: year, month, day, hour and minute. These indicate the time when the company is shown on the battlefield. If this tag is not present the unit is present at the battlefield when the battle starts, and if the tag is present then the unit will arrive at the position indicated by <pos> at the specified turn.

This tag contains a list of special named locations in the map. These locations can be known hills, rivers, bridges, villages etc. The locations are given as any number of <location> tags. The <location> tags have two attributes x and

y. These give the position of the location in the map in pixel coordinates. The data of the tag is the text that is to be shown on the map.

Locations have no other function apart from adding to the atmosphere of the scenario, and are not used by Civil in any other way than rendered.

This tag contains the objectives of the scenario. There may be any number of objectives in the scenario, and they are used as special positions on the map that are valuable. An objective can be an important hill, bridge, road crossing or just about anything. The objectives are rendered in the map using special icons to make it easy to spot them.

The objectives are given as a series of <objective> tags. Each <objective> tag contains the following attributes:

- id is used as an unique id for the objective.
- x gives the x position of the objective in pixel coordinates.
- y gives the y position of the objective in pixel coordinates.
- name gives a short name for the objective. This name may be shown in the map.
- points which gives the number of points that the objective is worth.
- owner which gives the current owner of the objective. The owner has captured the objective, and will be awarded points for it unless the opponent captures it. The possible values are union, rebel and unknown.

This tag contains all the map data. It is usually the largest single section in a scenario. Apart from the map data it also contains any extra features that the map may have. The only attributes the <map> tag has are xsize and ysize which define the size of the map. The <map> tag has a subtag <hexes> and an optional subtag <features>.

This tag defines some optional features that may or may not be used in a map. If it is present it contains a numer of <feature> tags, one for each specific feature. A feature can be a house, trench or something similar that isn't raw terrain. The <feature> tag has a few attributes and no main data:

- id which defines a unique id for the feature. It is used from within Civil as an icon id.
- x defines the x pixel position of the feature.
- y defines the y pixel position of the feature.
- type defines the type of feature. TODO: what types are there?

This tag contains all the map data. It is a long list of <hex> subtags. It has no attributes. Each <hex> tag contains the following attributes and no main data:

- x defines the x position of the hex in the map.
- y defines the y position of the hex in the map.
- terrain defines the terrain icon for the hex.

CHAPTER 6

Roadmap

Rough list of things to do before we can release 1.0. The versions are just so that we have some way of defining what we've done so far and what remains. No dates are set for the various versions.

0.50:

- simple combat. Does not need to use all available modifiers nor be exactly realistic, as long as it works.

0.60:

- framework for in-game sound effects
- ending a game (when does a game end, how does it work)
- working end-game screen

0.66:

- fully working turnbased system that works as well as 0.60

0.70:

- working scenario editor that can be used to create scenarios
- loading/saving games
- end-game screen should display statistics and a score
- finish the new floating status, minimap and orders panes.

0.80:

- simple AI client that does not crash, may even do something useful
- better scenario management that makes installing scenarios easier, maybe a separate application (civil-admin) to take care of that? [DONE]
- Updates to the XSLT for Docbook and the Scenario Format [DONE]
- At least one good scenario (TM)

0.8x:

- minor version increments as the tasks for 0.90 get completed.
- simple server-side ai that makes units react to actions, i.e. making units fire back at suitable targets and retreat when overwhelmed.
- Verify: does Line-Of-Sight work?
- visual notification to the player when the action is being calculated or sent over the socket.
- better combat that uses more of the modifiers, terrains, weapon stats etc.
- Include the character level formatting for the FAQ section of the DB processing. [DONE]
- Move all HTML streams to XHTML
- Add table of contents to the Manual XHTML output
- buildings
- Light Forests (scattered trees)
- Nice splash screen for the "Computing Action" dialog

0.85:

- initial realtime engine up and running

0.90:

- initial packages for: tar.gz, deb, rpm, osx and windows [DONE]
- a "credits" screen [Partially complete]
- sound effects for all in-game actions that should have effects
- setup and in-game music
- all required graphics should be completed
- all code should be done. Only tweaks and bugfixes left before it is called 1.0.
- all dialogs should be fixed so that all widgets etc are properly aligned and all looks ok. [DONE]
- Schemas [Partially DONE]
- Fix the colour map look up table generation
- Fix Windows HOME [DONE]
- Fix windows AI starting [DONE]
- Loading splash screen
- Complete the credits screen
- Missing game over screens

- genindex
- search

Built on Aug 30, 2018