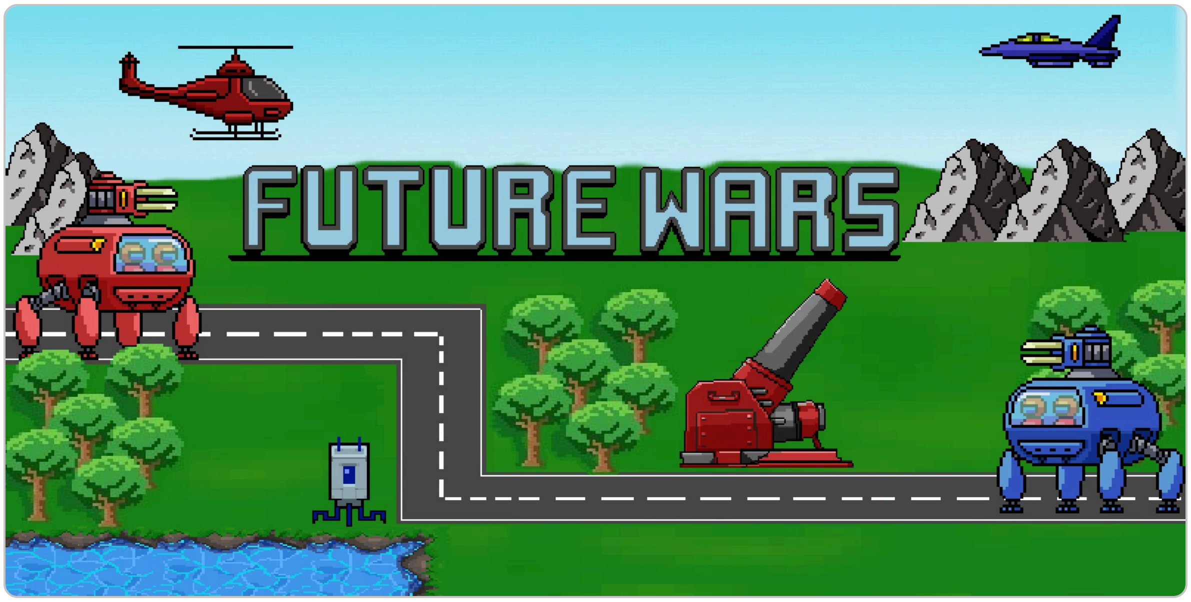
Future Wars

Table of contents

Future Wars



Splash Art

Future Wars is an Advanced Wars clone focusing on moddability and expandability. It was created as a university project by Simon Blum, Elena Schwarzbach, Sonia Sinacci and Max Rodler.

How to play

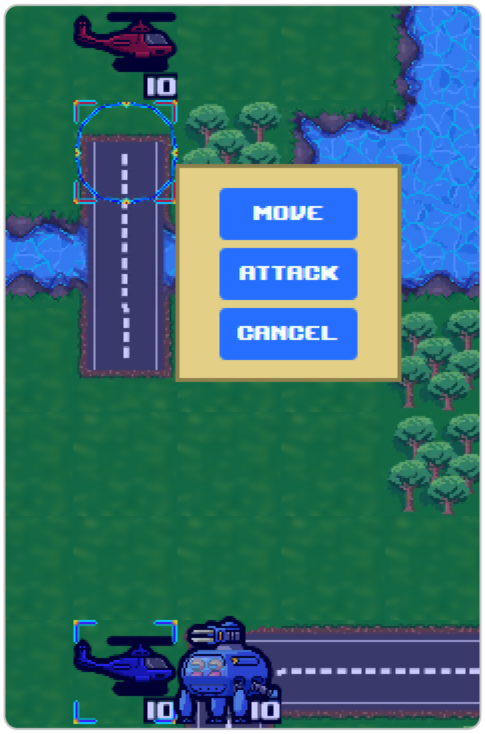
Once loaded into a map, the game is played in turn. For this the first player selects a unit by clicking on it - now he is in action mode and can do a few things:

Move

By clicking on a tile within the highlighted movement range and then selecting the move button, the unit will move to the selected tile.

Attack

When clicking on a tile, which has an enemy unit in range, the attack button will be enabled. By clicking on it, the player will then be able to select a unit to attack.



Ready to attack

Merge

If another unit of the same type is within range, the merge button will be enabled. By clicking on it, the player can merge the two units. This combines the units health, however be aware: Precious hp points can be lost in the process.

FileTypes

The file types that are used by the game are based on csv files. The following file types are used:

Map Files

For more information see Map Files (Map File)

Tile Files

For more information see Tile Files (Tile File)

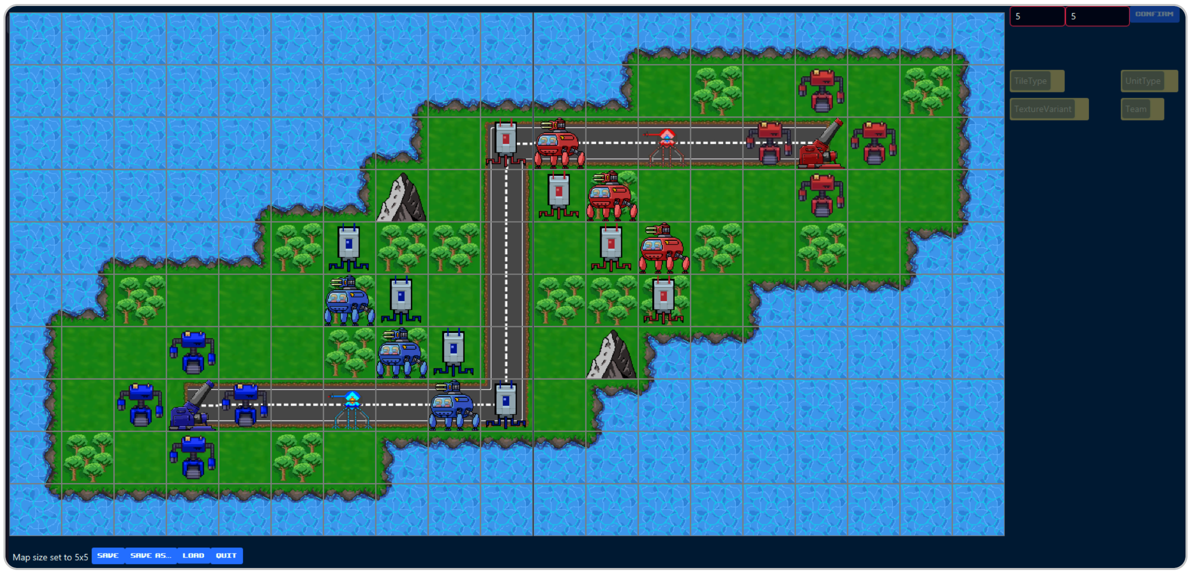
Unit Files

For more information see Unit Files (Unit File)

On top of this ".ini" based property files are used to safe settings and define textures which are used by the game.

Map File

Map files are marked by the ".fwm" suffix. They are used to define the tiles and units that are shown for each map. Unlike tile and unit files, they should not be created manually, instead one should use the map editor to create them.



5

5

Map size set to 5x5

Map Editor

Tile File

Tile files are marked by the ".fwt" suffix. They are structured as follows:

Row	Description	Values
1	Verification - must be:	FUTURE_WARS_TILE_FORMAT
2	*Ignored*	-
3	Tile Type	e.g. BRIDGE_TILE
4	*Ignored*	-
5	Movement Type	Must be one of: "PLAIN","WOODS","SEA","MOUNTAIN"
6	*Ignored*	-
7	Terrain Cover	Integer - for more Information see Damage Calculation (Damage Calculation)
8	*Ignored*	-
9	Texture Variant 1	Filename, including ending of texture variant 1
10	Texture Variant 2	Filename, including ending of texture variant 2
9	Texture Variant 3	Filename, including ending of texture variant 3
10	...	...

Any tile must have at least one texture variant, theoretically there is no limit to the number of texture variants a tile can have.

Unit File

Unit files are marked by the ".fwu" suffix. They are structured as follows:

Row	Description	Values
1	Verification - must be:	FUTURE_WARS_UNIT_FORMAT
2	*Ignored*	-
3	Unit Type	e.g. ANTI_AIR_UNIT
4	*Ignored*	-
5	Attack Range, Movement Range	Both fields must be integers
6	*Ignored*	-
7	Travel Costs	4 Integers for Plain, Wood, Sea And Mountain respectively
8	*Ignored*	-
9	Texture Team 1 (Blue)	Filename, including file ending
10	Texture Team 2 (Red)	Filename, including file ending
11	*Ignored*	-
12	Combat Stats	4 Integers for BaseDamage, Armor, Piercing and Low AirPiercing respectively
13	*Ignored*	-
14	TargetType	Must be one of: "GROUND", "LOW_AIR", "HIGH_AIR"

15	*Ignored*	-
16	Can Attack these Target Types	List of Target Types that can be attacked by this unit

For more information on Combat Stats see Damage Calculation (Damage Calculation)!

How Mod the Game

The game can by modified in 2 distinct ways:

1. Changing the internal files

If you want to change exisiting maps, tiles or units the files located in "src/main/resources/org/itdhbw/futurewars" need to be modified. This however is not recommended.

2. Using user files

A resource directory for user modifications is currently located at the project root. (In practice this should be e.g. at appdata on windows.) This directory can be used to

- Add new maps, by saving from the map editor
- Add new tiles/units by following the instructions
- Change textures by modifying the texture property files
    - These textures can e.g. include the hovered texture or even the splash screen

How to modify exisiting Textures

⚠ **Be Aware**

Specifying wrong paths can lead to undefined behaviour. If errors appear on startup make sure that everything is specified correctly, before trying to play the game.

The texture.properties file

The texture.properties file is located in resources/textures/other. By default, the userfile is empty, as any existing keys will overwrite the default values. If the file does not exist it will be created on startup.

Possible keys

The following keys can be used to modify the textures:

- SELECTABLE
- ATTACKABLE
- FALLBACK
- HOVERED
- HOVERED_OCCUPIED
- HIGHLIGHTED
- SPLASH_ART
- ICON

New textures

Any textures added by the user should lie within the same directory as the texture.properties file.

Example

```
SELECTABLE=selectable_example.png
ATTACKABLE=attackable_example.png
```

This would not work at these files dont exist

How to create your own Map

Any map should be created using the map editor which can be accessed from the main menu.

Saving a map

After creating a map it should be saved under resources/maps. After restarting the game, the map will automatically be loaded and can be played.

Things to note

The Map Size can only be changed **once.** Furthermore, when changing the size of a map, all tiles will be reset.

How to add your own Tiles

Each tile should have it's own directory under resources/tiles.

Tile file

First of all a tile file should be created, as described in the Tile File (Tile File) section.

Textures

After creating the tile file, all textures specified in the file (no path should be specified) are to be added to a 'textures' directory within the tile directory.

Using the Tile

Once the tile is created, it can automatically be used in the map editor.

Example

An example tile can already be found in the users resource directory.

How to add your own Units

Each unit should have it's own directory under resources/units.

Tile file

First of all a tile file should be created, as described in the Unit File (Unit File) section.

Textures

After creating the unit file, both textures should to be added to a 'textures' directory within the unit directory.

Using the Unit

Once the unit is created, it can automatically be used in the map editor.

Example

An example unit can already be found in the users resource directory.

Damage Calculation

Within the original game, the damage ("baseDamage") is read from a matrix that contains the damage for each interaction. Since we want to allow for an easy addition of new units in our version of the game, we decided to dynamically calculate the damage ourselves. However, because the damage in the original game is not linear, we had to introduce some auxiliary values to calculate the damage.

Target Types

To determine which unit can attack which other unit, all units are classified into the following three categories:

- GROUND: All ground units
- HIGH_AIR: Bombers, Fighters
- LOW_AIR: BattleCopter The division into HIGH_AIR and LOW_AIR is necessary because the BattleCopter is significantly different from the other air units. These categories are also known as TargetTypes.

Combat Stats

For the actual damage calculation, the following values were introduced:

- BaseDamage: A base value for the damage, specified by the other values.
- Armor: Indicates how much percentage of incoming damage is absorbed.
- Piercing: Indicates how much percentage of the opponent's armor is ignored.
- LowAirPiercing: Same as Piercing, but only when the opponent is LOW_AIR.

Again, the last value is necessary because the BattleCopter is too different from the other units.

Combat Preview

Comparison the the original game

All these values were adjusted to ensure that the damage values are as close as possible to those of the original game. This was successful for all but two interactions, so there is virtually no noticeable difference in gameplay compared to the original. The calculations in the "UnitAttackController" combine our own calculations (calculateArmor, calculateBaseDamage) with the other calculations listed in the wiki (calculateActualPercentDamage, calculateDamagePoints).

These calculations are a bit more complicated and elaborate, but they allow new units to be implemented into the game easily without the need to create a new matrix.