The Multi-Gen Game

Documentation 0.0.4.9.9-a

[Name redacted]

Release date: Feb. 2021

1 Patch Notes

Updates added in version 0.0.4.9.9 include:

- a web port, via emscripten;
- other features that the Lead Developer will not say until more developers speak to him.

2 Terms & Definitions

1. MGG - Multi-Gen Game. An elaborate, customizable, single-player, 2D grand strategy game, currently written in C++ using the SDL2 game libraries (if you're curious, versions 0.0.1-0.0.3 used Allegro 5). Remark: this is one of many *working titles*. The game will take place in a fictional world over a period of centuries (see IGT), i.e. multiple generations.

2. IGT - In-Game Time. This refers to the passage of time in the *in-game* calendar system. In the game, there are 18 months per year and 10 days per month. (This is subject to change in future versions and can be easily adjusted by changing the `DAYS_PER_MONTH` and `MONTHS_PER_YEAR` values in `src/game-loop.hpp`). Dates are represented (e.g. in the upper-right corner of the WMSL) as `yyy.mm.dd`.

3. WMSL - World-Map Strategy Layer. This is the "outer layer" of the game, the one that is shown any time a game is launched. It is a system in which the user can make strategic decisions about unit placement, respond to Events, and pause and unpause the game (using the space bar). While unpaused, the game "moves forward in time" (see IGT) at a certain rate, modeling the passage of time in a fictional universe.

4. WMSL Proper - if I use this term, it means *just* the window/screen containing the headers ("MAP", "ARMY", etc.) and NOT any temporary window launched separately. This definition is being written viz. a viz. version 0.0.3 (and 0.0.3.9). So if this makes no sense in a future release, then just disregard it.

5. TMB - Tactical-Map Battle. This is a turn-based mini-game that is periodically launched by the WMSL at deterministic moments, or pseudo-randomly depending on context.

6. Unit - a character in the game. A Unit will represent a member of the Army.

7. UC - Unit Class. This refers to the *occupation* or *unit type* for a particular Unit. At the moment, the only examples are Archer and Soldier. This will be greatly expanded later.

8. Stat - one of finitely-many attributes that *all player Units* possess, like Health Points, Strength, Speed, etc. These values serve the primary purpose of affecting a player unit's performance *during a TMB*. As of version 0.0.4, there are roughly 14 different stats, listed in Appendix B.

9. Trait - Whereas Stats are basic things like Health, Strength, etc. (things modified by the StatChange class), Traits are like "personality" traits that can be gained/discovered through Events (which, so far, have not been developed very much). This is similar in concept to Traits in other grand strategy games, in that Events will occur pseudo-randomly at the WMSL when the game is unpaused. However, the idea that is more unique is that they may affect Stat Growths when a Unit levels up.

10. Spawn Pool - This refers to a general region where retired units can be sent to train new units. As currently defined, this is a *player-owned* Province.

11. Lead Developer - I've been using this term, and you may be wondering who it refers to. I often write documentation & emails in the third person (or occasionally the royal We), to sound more official. To be clear, I ([Name redacted]) am the lead developer.

3 The WMSL

When the WMSL is unpaused, IGT will "tick" by, at an adjustable rate. To adjust this rate, try pressing the '+' or '-' keys when the game is *unpaused*. The number displayed in the upper-right corner approximately equals

$$10 * (pace\ in\ ticks/sec).$$

The maximum pace is 10 ticks per second, making 100 the maximum number displayed.[1]

The World Map is composed of **Provinces**, each of which contains **Settlements**. While the game is still active/running, some Provinces are player-owned, others are enemy-owned, and some may be contested (have player units and enemy units both present, both trying to conquer).There are three ways for the game to end:

- The player conquers all provinces (prompting a Victory screen with a Score value).

[1]You may wonder what the minimum pace is. The remarkable fact is: there *is* none - you should be able to set the speed arbitrarily slow, though we're not sure why you'd want to.

- All player units are debilitated (prompting a Defeat screen).
- The player deliberately exits the game ("Cold exit"). A future release will auto-save the game state, but this has not been implemented.

The WMSL also has music enabled by default. To pause or unpause the music, press the 'm' key.

The WMSL, the foundation of the game is, like many other grand-strategy games, composed of different tabs.

3.1 'Map' Tab

When this tab is selected, the world map is displayed. The user may click on a province to get data about that province (name, settlements, etc.). Once you have units in the Field of a Province, you may click the 'S' button to invade a Settlement. (A new window is generated where you can use arrow keys to select the Settlement.) Unpause, give it a few moments, and a TMB should spawn.

3.2 'Army' Tab

All player units are displayed in a scrollable list under the Army tab. This is the heart of the game, where the user can choose to move units into other provinces.

When a new game initiates, the unit list is read from a file, namely 'res/world-map-2.conf'. You may adjust this file by adding or deleting lines that contain unit names.

When the game is unpaused, clicking a unit will simply display data for that unit. However, when it is paused, the user may left-click to see data OR right-click in order to add the selected unit into a temporary "group" along with other selected units. Then, when the group has been set up, the user may click the `P` symbol in the lower-right corner in order to set a destination Province for that group. (When paused, the user can move units to different provinces instantaneously.) Units will land in the "Field" of the Province. After that, you can issue the order to *invade a Settlement*; this order is issued under the Map tab.

3.3 'Research' Tab

(Requirements & design pending)

3.4 'Homefront' Tab[2]

This is where things get interesting. In fact, this is the core of the justification for the name "MultiGen" Game.

TraitModifiers (which are values specific to each Spawn Pool) affect the Traits that are assigned to Units that spawn from that Pool. Then, a given Unit's Traits affect StatGrowths, which affect its Stats. However, here's the new twist: a Unit's Traits, *at the point at which*

[2]not implemented yet. In fact, even the documentation for this subsection is a work in progress.

it retires, will affect the Trait Modifiers of the Spawn Pool which it trains. So Traits can get "recycled" over the course of generations.

As currently written, Retirement happens automatically at a fixed age (60). In a future edition, there may be a more manual "Retire" button.

When a Retired Unit is sent to a Province, the idea is that that Unit's Traits can be used to influence / mathematically raise the Trait Modifiers in that "SpawnPool" (Province).

There are at least two UI candidates, as described below.

3.4.1 Candidate A: The Two-Phase System

Under this system, there are two Phases (temporary game states that apply only to the Homefront Tab): a "Show Retired Units" (SRU) Phase and a "Select Spawn Pool" (SSP) Phase.

During the SRUp, a table of retired units (one row per Unit) is displayed, much like in the Army Tab but for retired units. It shall show Name/Age/Location/Traits. An advantage of this UI candidate is that it would basically be a parallel-looking UI to the UI for the ARMY Tab, except the fields being displayed would be different (and Traits would be displayed instead of Stats - which are very different things). In order to ensure that a Unit's entire list of traits (indefinitely long) fits inside a row of a table, each Trait might have a small icon representation that the user can mouseover. In the SRU phase, the player shall see this list of retired units and be able to decide "ok, THAT one. I want to change THAT Unit's location," while pointing to a single unit[3]. After units are selected, the user shall be permitted to click a button - much like the "Invade Province" button on the ARMY Tab, but probably larger, prettier, and more intuitive - and that button saves the selected unit(s) and toggles the Phase.

Now, the Phase is toggled, but of course, the system shall remember which Units have been selected. The SSP Phase no longer shall show the Retired Unit Roster. Instead (after optionally doing a cool animation), it shall display a world map. This map shall display SpawnPools, and as the User mouses over Spawn Pools, it shall show the Pool's current TraitModifiers and also the "new" Trait Modifiers which would theoretically take effect if the user moves the selected units into that Spawn Pool. Then, the User shall be able to right-click to select that Spawn Pool, at which point the action is done: Selected units are moved into that Spawn Pool and the changes start to take effect.

3.4.2 Candidate B: The Drag and Drop System

This candidate is best summarized as: "Click and Drag" into SpawnPools. It is very similar to Candidate A, but it would try to fit both the Retired Unit List and the World Map on the same window, at the same time. Candidate B removes the need for two different "Phases," which simplifies the design. There may be some resulting tradeoffs, e.g., not all units may fit in the list; but the list will still be scrollable. In order to be able to fit as many traits as possible on the screen at a time, the screen will likely be divided vertically into a top half for the Unit Roster and a bottom half for the (resized) World Map image.

[3]This may become extensible, so they may be able to select multiple units, much like in the ARMY Tab.

3.5 'Espionage' Tab

(Requirements & design pending)

4 The TMB

4.1 Terrain Tile Codes

Each TMB takes place on a map. Map data is read from files, each of which contains a space-delimited table of integers. Each integer (corresponding to a tile in the map) represents the **terrain type** of that tile. A key for these terrain type codes should be inside 'src/oo-tmb.cpp' and is reproduced below:

Terrain Code	Interpretation
0	Ocean[4]
1	Plains
2	Forest
3	Hills
4	Ice
5	Sand
6	Mountains

However, in a future edition, these terrain tile codes will be revamped in order to encode which tiles can be used for Player Unit spawn (other) tiles can be used for Enemy Unit spawn locations. Most likely, these will be implemented by taking the above Terrain Code and doing a bitwise OR with a constant (16 to signal a valid Player Spawn Location and 32 to signal a valid Enemy Spawn Location). So if you open a TMB terrain conf file and see numbers larger than 16 or 32, this should explain why.

4.2 Mechanics

All TMBs are turn-based. Player units have their HP displayed in GREEN font whereas the enemy AI unit HP is in RED[5]. The user may click the icon/sprite for their units, at which moment the right-hand menu updates and gives you options for giving that unit instructions. Go through these options carefully. When you've finished giving orders to each of your units, the turn phase updates (i.e., the enemy takes its turn, issuing orders to its units). Then it is the player turn again and the cycle continues.

4.3 Types of TMBs

There are different variants of TMBs, each with their own victory and defeat conditions (feel free to study them yourself). The most common (and the only one currently used in Alpha Testing) is the RoutTMB, in which the only way to win is to leave all opponent units debilitated.

[4] currently, no Units can directly cross bodies of water. There may be flying units in a future release.

[5] The developer plans to make this an easily-adjustable color scheme in the future.

4.4 Omega Mode

Some open-source games have an Omega Mode that allows the player to have invincibility or special powers. This game offers a slightly different cheat: allowing the AI to play through a TMB *in place of* the user. This can be activated by pressing three keys in sequence: `g`, `o`, `d`. It can be *deactivated* by pressing the `n` or `d` key.

5 Known Bugs

After invading a Settlement or opponent-owned Field, I get a grey/empty/non-responsive screen (all platforms) - believe it or not, this one is not a bug; it's a feature. (`:P`) When an invasion is made, the TMB is added to the list of pending battles. Try clicking back to the original window and un-pausing in order to *jar* the system back into motion and load the battles.

Upon pressing the button to invade a Field, the game crashes (VM platforms) - this is a serious error that seems to happen only on VMs. To minimize the risk of this error, try un-pausing the game for a few seconds *before* you pause and select your army. If this still doesn't work, then try scrolling up and down through the unit list before selecting (right-clicking) unit members. If you want to report this error, let the developer know and provide as much information as possible (OS and version, system specs, call-stack when running through GDB, etc.). In the unfortunate event that you cannot ever avoid this error, you may still compile the source and call `./bin/static/release/oo-tmb 0` to test a basic TMB. (Even if the WMSL doesn't work, the TMB should be much more stable.)

6 Help Wanted

The project could use help in a variety of areas; this list is sorted from most to least important work:

- Artwork (unit animation sprites, navigation bar icons, etc.) and music;
- A new map for the WMSL;
- New features;
- Play testing and bug filing / fixing.

7 Bounties

Bounties (for more serious endeavors) will be listed here. Note the following:

1. If you are the first to complete one of the active tasks, you are eligible to receive the bounty *and* to have your (user)name listed as a Major Contributor to the project.
2. Bounty prices will start low (maybe even at zero!) and rise over time.

3. Bounties will close the moment someone redeems them. If no one redeems them, their status will be renewed or closed on the next patch update.

- B00000: Automatic boundary detection [Open, $0.00 + Major Contributor Status]
  In the current framework, the "adjacent Province" system must be hard-coded for each World Map. If a program were developed to process/analyze the world-map image and find adjacent provinces automatically, this would be most impressive and useful. If you can solve this problem, please contact the Lead Developer.
- B00001: Windows package [Closed]
- B00002: .deb package [Open, $0.00]
  Release a version in deb form, with proper Linux file paths. (Again, if you succeed, show the Lead Developer. The game needs to be at least as stable as the desktop version.)
- B00003: Android Release [Open, $0.00 + Major Contributor Status]
  Release a version of the game for Android, that runs just as smoothly as on desktop. Show the lead Developer exactly how you did it. The Lead Developer must replicate your steps successfully in order for you to redeem this bounty.
- B00004: Solve TMB Responsiveness on Windows [Closed]
- B00005: Teach the Lead Developer DirectFB [Open, $0.00 + Major Contributor Status]
  Somebody teach the Lead Developer how to use the Linux framebuffer. Or, write better documentation on DirectFB. But no one will. He's just ranting in the form of an Issue.
- B00006: Solve Multithreaded TMB Timer Failure [Closed]
- B00007: Efficient Binary Deserializer [Open, $0.00 + Major Contributor Status]
  Implement binary data processing, for serialization of the Unit data (and perhaps Province/Settlement data while you're at it). This is the long-term goal. There are intermediate goals, like (a) reading text data with stream-like objects; (b) using RegEx; (c) using Boost to serialize object data; or (d) some combination of the above. But the long-term goal would be to use an existing data parser, ideally for binary data that the client cannot easily hack / modify.
- B00008: libpng warning: iCCP: CRC Error [Closed]

8 How to Compile (Platform-specific)

8.1 GNU/Linux

In order to *compile* any recent edition of the game, you will need the SDL2 development libraries (`SDL2`, `SDL2_image`, `SDL2_ttf`, and `SDL2_mixer`) installed on your computer. Then you may compile it using the Makefile or using CMake.

However, it should *not* be **necessary** to compile the MGG itself in order to test it. There is a run-script (conveniently called `run-script.sh`) that will try running the game executables in at least four different ways. Note, however, that if the game crashes, this script will simply try to run it again. Thus, *when using this script*, if the game spontaneously reloads and shows a title screen, that means that there was actually a crash - and it will be difficult to diagnose the error after the fact.

8.2 Windows

The binary exectuable was generated using Visual C++ on Windows 10. It *should* run on any 64-bit Windows machine, so compilation should be optional. However, if you insist on doing so[6], try following the steps given here:
https://www.wikihow.com/Set-Up-SDL-with-Visual-Studio.

8.3 Mac

There is no official build for Mac; if you would like to volunteer to create one, contact the Lead Developer. However, other contributors have confirmed that the code can compile using clang on MacOS (either from a terminal, or using XCode). The Lead Developer does not own a Mac machine. Should you have questions / complications compiling the code, ask the lead developer to put you in contact with one of the Mac OS experts.

9 Configuration

NOTE: The configuration for the game is in the process of being rewritten in JSON so that it can be parsed efficiently in a standard, human-readable way. Thus, the following information may or may not hold true for the current version.
The main configuration file is named `res/world-map-3.conf`. It has the following format:

```
<millisecond time delay (int) - may now be obsolete>
(<number of provinces (int)>)
Province 0 (Optional description)
Name: <Province_Name> (Ownership_status) (Ownership_char)
Settlements:
```

Then follows a list of Settlements, each having the form

```
<Settlement Name>
<Ownership Status - 0 for Player, 1 for Enemy>
```

Then, nested inside each Settlement is the Unit data. Each Unit is listed on its own line, having the following format:

[6] and the Lead Developer has no clue why anyone would prefer to do this on Windows rather than Linux, but so be it.

`<Unit_Name> <UC_ID> (Team <TeamID - 0 for Player, 1 for Enemy>) <x> <y>`

Finally at the end of the list of Units, there must be a

`End_Unit`

token. The proceeding line contains name of the next Settlement, or (at the end of the list of Settlements), an

`End`

token. A default configuration file for the *simpler world map* (version 0.0.3) is shown in an appendix for reference.

10 Credits

10.1 Major Contributors

Special thanks to all Major Contributors. These include, in alphabetical order by username:

- Alison (bounty hunting, Aug. 2020),
- Candbot (UI feedback and design, 2019 – present),
- gamez7 (bounty hunting, Aug. 2020),
- MK (SGDT logo, additional artwork, 2018 – 2019),
- MrMcApple (co-development of original Java demo, 2017),
- Slpee (concept, design/vision, 2017 – 2018),
- Tatsu (co-development of C++ code, Nov. – Dec. 2020),
- The Seagull (a new title for the game),
- TrebledJ (co-development of C++ code, 2019 – 2020)
- Zunethia (primary testing, artwork)

10.2 Minor Contributors

Thanks also to all developers not affiliated with the project who have given brief advice along the way, including:

- David Holmes
- Ten

10.3 Music Credits

Main WMSL Theme
Lobo Loco, "Snowmelt Yukkon River"
Creative Commons by Attribution. No changes.

Victory Theme
XTakeRuX, "Destructing Own Kingdom"
Creative Commons by Attribution. No changes.

"One Player Unit Remaining" Theme
Meydän, "Pure Water"
Creative Commons by Attribution. No changes.

"The future is inside us; it's not somewhere else."

Radiohead

A Example Configuration File for Version 0.0.3

```
5
(4)
Province 0 (Left)
Name: Mars (0) (P)
Settlements:
Field
0
End_Unit
Phobos
0
Kathleen 0 (Team 0) 3 4
Graham 0 (Team 0) 4 5
End_Unit
Deimos
0
Linus 0 (Team 0) 3 4
alexge50 0 (Team 0) 4 5
End_Unit
Olympus_Mons
0
Iron_Park 0 (Team 0) 3 4
Nalee 0 (Team 0) 4 5
Chau 0 (Team 0) 4 5
End_Unit
Valles_Marineris
0
Gold_Unit 0 (Team 0) 4 5
End_Unit
End
Province 1 (Top)
Name: Jupiter (2) (C)
Settlements:
Field
0
End_Unit
Io
0
Rockington 0 (Team 0) 1 2
Arin 0 (Team 0) 2 3
Wozar 0 (Team 0) 4 5
Deflated_Pickle 0 (Team 0) 3 4
End_Unit
Ganymede
```

```
1
Caribou 1 (Team 1) 2 3
End_Unit
Europa
1
End_Unit
Callisto
1
End_Unit
End
Province 2 (Bottom)
Name: Pluto (2) (C)
Settlements:
Field
0
End_Unit
Charon
1
End_Unit
Styx
0
Carol 0 (Team 0) 4 5
Laureen 0 (Team 0) 4 5
Rosemary 0 (Team 0) 4 5
Jiarui 0 (Team 0) 4 5
Sramika 0 (Team 0) 4 5
Flo 0 (Team 0) 4 5
Minji 0 (Team 0) 4 5
Alex 0 (Team 0) 4 5
Eric 0 (Team 0) 4 5
Sam 0 (Team 0) 4 5
End_Unit
Hydra
0
Kurt 0 (Team 0) 2 3
Krist 0 (Team 0) 3 4
Dave 0 (Team 0) 4 5
End_Unit
Nix
0
Ginger 0 (Team 0) 2 3
Jack 0 (Team 0) 4 5
End_Unit
End
Province 3 (Right)
```

```
Name: Saturn (1) (E)
Settlements:
Field
1
End_Unit
Titan
1
Robot 1 (Team 1) 3 4
End_Unit
Enceladus
0
End_Unit
Mimas
0
End_Unit
Iapetus
1
Bully 1 (Team 1) 1 2
Block_Dude 1 (Team 1) 3 4
Dinco 1 (Team 1) 4 4
End_Unit
End
```

B Stats

See `src/item.hpp` for an up-to-date listing of stats. Those included as of October 2019 are:

1. Max HP
2. HP
3. Strength
4. Speed
5. Dodge
6. Armor
7. Accuracy
8. Initiative
9. Luck
10. Constitution
11. Magic Strength
12. Hit Speed
13. Magical Resistance
14. Skill

These do not include **Team**, which may or may not be considered a Stat.