VERSION 1.7

NOVEMBER 1, 2023

GETTING STARTED WITH VASL DEVELOPMENT

A STEP-BY-STEP GUIDE

PREPARED BY DOUG RIMMER

DOUGERIMMER@GMAIL.COM

GETTING STARTED WITH VASL DEVELOPMENT

GETTING STARTED WITH VASL DEVELOPMENT

DOCUMENT OVERVIEW

This document provides instructions on obtaining, installing, and using a number of tools necessary to enable VASL development.

This is a multi-step process involving a number of separate tools and products and can appear quite complex. This document tries to provide sufficient detail to allow those interested to get up and running on their own. In the event of problems requiring assistance, contact information is provided.

The document is written to enable those unfamiliar with such tools to create an environment in which they can contribute to further development of VASL. Experienced users will find that they can skip some of the introductory text in each section and proceed directly to the detailed instructions.

Obtaining and installing the tools required for VASL development involves using websites and software that have their own development and update processes. This will often create minor changes in the look-and-feel of these tools. Users should anticipate this and not be stymied if each detailed step is not exactly as described in this document.

For those interested in creating VASL Boards, see separate instructions on the VASL wiki within GitHub (see Section 3 below for information on GitHub).

GETTING STARTED OVERVIEW

In order to work on VASL source code, a number of steps must be completed. The VASL source code is a library, not a complete standalone application. The library is used by VASSAL (http://www.vassalengine.org), which is a standalone application. Development will require the VASL source code along with VASSAL.

The source code is written in Java. To enable access to this, it is necessary to install a version of the Java SE Development Kit (SDK), which includes tools for developing, debugging, and monitoring Java applications.

To obtain the source code for VASL and to manage changes made to the code, it is necessary to use the VASL Repository on GitHub. This requires creating a Git user account and installing software.

To actually develop new code requires an Integrated Development Environment (IDE). Several different IDE's can be used, and this document provides instructions for one of them: IntelliJ. Other IDE's will have help tools that can be used to walk users through the necessary steps.

In order to integrate code changes, it is necessary to 'build' the VASL module. This document provides instructions for doing so using a tool named Maven which has to be obtained and installed.

While many of the above tools can be purchased, all have "community" or free-access versions that can be used for VASL development. No purchases are necessary.

Finally, this document provides more detailed instructions on how to obtain, install and use all of the above-mentioned tools.

1.0 VASL AND VASSAL

1.1 THE VASL MODULE

The VASL source code is a library, not a complete standalone application. The VASL source code is compiled into the VASL module, which is used by VASSAL to provide the ASL game interface via the internet. Usage of the source code is subject to the terms of the Library Gnu Public License (LGPL), described in the LICENSE.txt file and also available from http://www.opensource.org.

1.2 THE VASSAL ENGINE

VASSAL is a standalone application which is used in conjunction with modules such as VASL to provide game interfaces. Neither VASSAL nor any of its modules is a complete game in itself. Both VASSAL and a game module are required in order to play.

1.2.1 VASSAL under the covers: why you need it and what it does

VASL is one of many game modules developed for the VASSAL game engine. The VASSAL module editor is used to add and update VASSAL capabilities to VASL components, examples of which would include counters, map window options (e.g. zoom), etc.

VASL is one of a small number of modules that contain a lot of custom Java code to enhance and augment standard VASSAL features. For example, concealing a stack is performed via custom VASL code.

VASL "development" could involve either of the above.

1.2.2 VASSAL Module documentation

If you wish to know more about VASSAL functionality and how it works, please consult the VASSAL documentation. There are also a series of YouTube videos about VASSAL, some quite detailed.

1.2.3 Using the VASSAL Module Editor

The VASSAL module editor comes with VASSAL program that you install to play VASL. See the VASSAL documentation for more details. The VASSAL module editor is used primarily for counter development and other non-code related changes to VASL. It is not used for adding custom java code.

1.2.4 Use VASSAL code during the development of custom VASL code

In order to access the VASSAL code base functionality while developing custom code for VASL, several references to VASSAL will be created in Section 4.1.3 Step 2.

2.0 GETTING THE UNDERLYING TOOLS: SDK

2.1 SETTING UP THE JAVA SOFTWARE DEVELOPMENT KIT (SDK)

In order to be able to use Java code to create custom code for VASL, a Java Software Development Kit (SDK) must be installed and then referenced. Details on how to reference the SDK in VASL projects are included in Section 4.1.3.

2.1.1 The SDK: why you need it and what it does

The SDK works primarily behind the scenes in VASL development to enable the use of Java code in the VASL module. The SDK includes tools for developing, debugging, and monitoring Java applications which are exposed to the developer via their Integrated Development Environment (see Section 4).

2.1.2 Getting the SDK

The SDK can be found by searching for "Java Software Development Kit" on the internet. Since new versions are issued regularly, no specific version is cited here. It is suggested to use the latest version you can find.

2.1.3 Installing the SDK

From the java site you chose, complete the download and installation of the SDK by following the screen prompts as you would for any software install. It may be helpful to note the directory in which the SDK is installed as you will need to be able to find it when adding Java references in Section 4.

After installation, validate the SDK install, ensure that Java is in your computer's path, and verify that the Environment Variable JAVA_HOME exists on your computer and points to the installation directory. Since methods of doing so can vary by OS, please check the internet for details.

3.0 GETTING THE CODE: GITHUB

The VASL source code is stored in a code repository called GitHub.

3.1 THE GIT CODE REPOSITORY

A Code Repository is essentially a web-based file storage location that includes tools for managing access to files and updates to them and is specially designed to support software development by teams of people.

3.1.1 Code Repository: why you need it and what it does

GitHub is used to store all of the VASL source code. In addition, Git is a version control system (VCS) that records changes to a file or set of files over time so that specific versions can be recalled later. A VCS allows developers to revert files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS generally means that if someone screws things up or loses files, it is possible to easily recover.

3.1.2 How to install Git and access GitHub

3.1.2.1 GitHub

Create a Github account at https://github.com/.

3.1.2.2 Git

Git must be installed on your computer. It can be found at http://www.git-scm.com/ or by googling 'git-scm' if the link does not work. Git-scm is available for most major operating systems. Choose the latest version for your system.

You may see references to the GitHub desktop or GitHub GUI. For the purposes of VASL development, it is NOT recommended that you install this software.

When installing git-scm on Windows use the default options with the exception of allowing git commands from the command prompt.

3.1.3 Cloning the VASL code repository for local development

1. In order to update the VASL code repository, it is necessary to join the vasl-developers team on GitHub. To do so, go to https://github.com/orgs/vasl-developers/people and message one of the owners that you would like to join. (This is only necessary if you wish to have "write" access to the repository so that you can contribute code. If you just want to clone a repository and code and build separately, you don't need to join the team.)

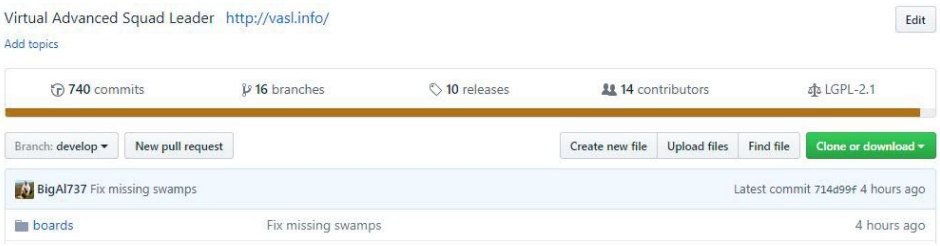
2. Once you have received confirmation that you are a member of the group, sign in to GitHub.

3. Search for "VASL".

4. Select "vasl-developers/vasl" (click on the 'vasl' at the end as it is a separate hyperlink).

5. Click on Branch-develop and type a name for a new branch in the textbox then click on "Create Branch:". If "Create Branch:" does not appear, check that you are a member of the vasl-developers group. For those wishing to develop and contribute code, please start your branch name with the prefix "feature/".

6. Copy the HTTPS Clone URL for your branch to the Clipboard by clicking on the "Cone or download" button on GitHub and copying the URL for the text box that will display.



7. Now go to Section 4.1.3 to complete the cloning process from within the IDE.

4.0 WORKING WITH THE CODE: INTELLIJ

4.1 THE INTELLIJ IDE

Integrated Development Environments (IDE's) provide a means of linking a number of tools together to allow developers to write software code and then manage their work by linking to project management tools and repositories.

4.1.1 IDE's: why you need one and what they do

Above all, the IDE is the place where code gets written. IDE's link to underlying software development tools (such as the SDK) that then provide developers with access to coding languages, structures and libraries. They also integrate tools that allow developers to link their work to larger teams and projects.

A number of IDE's can be used to write VASL source code in Java, for example, Eclipse. The following instructions apply specifically to another one, IntelliJ. They could be used as a guide to working with Eclipse or other IDE's.

4.1.2 Installing an IDE: IntelliJ

To install IntelliJ:

1. Go to https://www.jetbrains.com/idea/download/

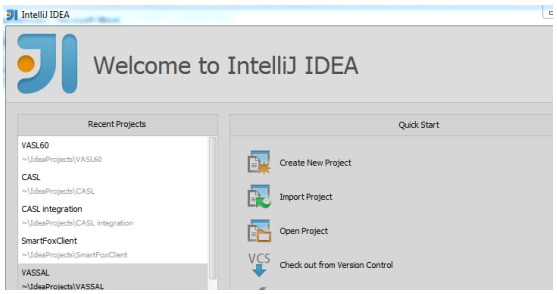
2. Download the Community Edition version for appropriate platform (Windows, Mac, etc.)

3. Run install as normal for the platform (i.e. on Windows, click "Open" or "Run" in the dialog boxes)

4.1.3 Setting up IntelliJ for development

1. Create the VASL Project

Start IntelliJ. You should see an introduction screen with the option to create or import a project from Version Control. Select that option. The screen may have differences from the one shown below due to version changes but should still have an Version Control option. Click it.



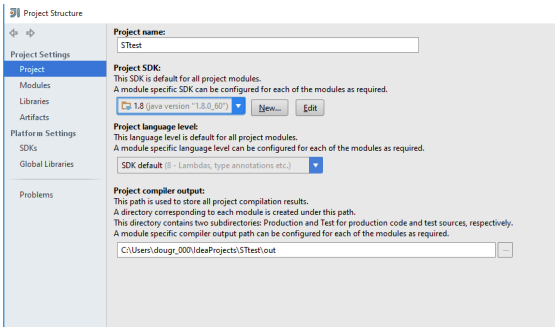
IntelliJ will then present a number of screens to complete the importation process. These screens change often and thus are not shown or described in detail. For the most part it is ok to accept the default choices presented by IntelliJ. A few key points to check are highlighted below.

At some point you should be asked for a URL from which to clone the project using version control. This is the URL that you copied in Section 3.1.3 Step 6. Enter the URL.

If during this process you are given a choice to use Maven, select Yes or Accept.

2. Configure the VASL project

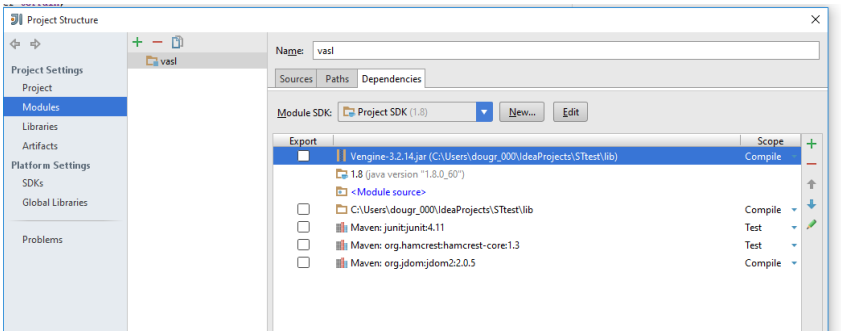
You will need to configure the project in order to enable tools such as the SDK and VASSAL code to be accessed when coding.



Select Menu File →Project Structure. Select Project Settings and check that the Project SDK is the one that you installed (see Section 2.1.3). If not, click the dropdown or "New" button to select the Java SDK.

Click on the Project language level dropdown and select the one that corresponds to your java version.

Click the Modules tab and then the Dependencies tab. The list should include the following items.



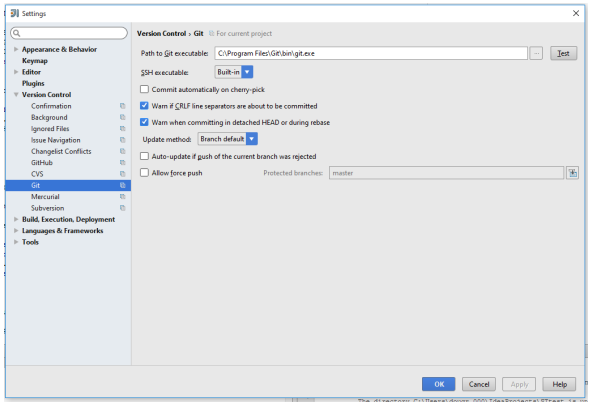
If the Module SDK item is missing, add it.

If the Vengine item is missing, click the green + sign and select “Jars or Directories”. Then navigate to the “.m2” directory under your Username folder. Depending on your OS, its location may vary. From .m2 navigate to repository\org\vassalengine\vassal-app, select the latest VASSAL version folder, expand it and select “Vengine.jar” then click OK. It will be added close to the bottom of the list in the Dependencies panel. Select “Vengine.jar” and use the up arrows to move it to the very top of the list. Click OK to close the Project Structure window.

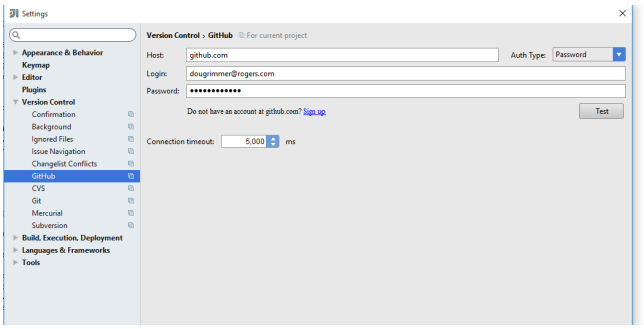
If the .m2 directory is not present on your machine, contact the VASL team on GitHub for support.

3. Work with Git

To ensure that the link to Git is in place, select File -> Settings and on the next screen select Version Control and then Git. Verify that the correct path to your installation of Git (see Section 3.1.2.2) is listed in the Path to Git executable textbox. If not add it by typing the path or navigating to it via the Directory box beside the textbox. Click Test to verify the linkage.



For GitHub, click on the GitHub tab on the same Settings screen and enter your username and password for GitHub. Make sure Host shows GitHub.com and click Test to verify the connection is enabled and then click OK.



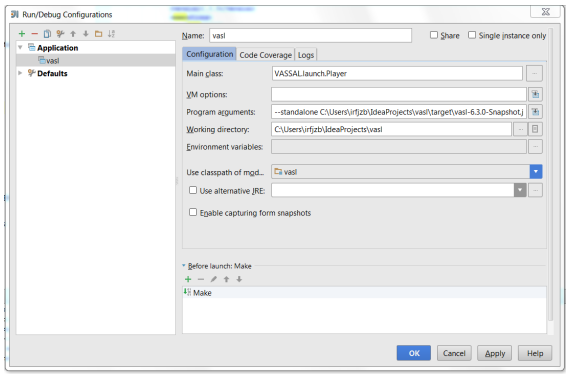
4. Create a Run Configuration

To run the project (to verify that code changes are working), certain Run configurations must be set. Select Menu Run → Edit Configurations. If no item is listed in left-hand listbox, click the green + sign and select “Application”. Enter a name in the Name textbox and “VASSAL.launch.Player” in the Main Class textbox.

Enter --standalone "C:\Users\<you>\IdeaProjects\<your project name>\target\vasl-6.4.1.jar" in the Program Arguments textbox, using the directory where your current module file is located. Note that versions numbers (“6.4.1”) will change over time. This must point to an actual .jar file so when running a module for the first time it is necessary to either build the module first (see Section 6.1) or point to already existing file (such as the one used to play VASL games).

VASL .vmod files can be copied and then renamed .jar for this purpose.

Ensure that the Working Directory textbox refers to the folder where the current project is located. If not, type in the folder name or navigate to it using the Directory button beside the textbox. Select the classpath of module from choices available from the dropdown arrow; do not select “test”. Instead use the option similar to your project name.



Click OK.

Select Menu Run → Run “Your project name” to run the project.

4.1.4 Cloning the VASL code repository for local development from within IntelliJ

1. In order to clone the VASL code repository from within IntelliJ, follow the steps in Section 4.1.3 but start by selecting File -> New -> Project from Version Control. Some of the screens in the remainder of 4.1.3 may contain minor differences.

5.0 MANAGING THE CODE

5.1 USING THE IDE TO CONTROL CODE CHANGES

While writing new code is one thing, ensuring that it integrates properly with the existing code base and new code being developed by other developers on the team is another issue. The IDE and GitHub work together to do so.

5.1.1 Code Management: why we need it and what you must do

Good code management ensures that no work is wasted, and no conflicts enter the code base due to uncoordinated efforts of multiple developers. All developers must access the VASL code and commit changes properly in order to maintain an integrated set of code files that will produce the VASL module for use by ASL players.

5.1.2 Creating a branch on GitHub

Developers should create a branch and use it for specific coding tasks before proceeding to write code. Branches should be created before cloning the source code from GitHub to IntelliJ). See Section 3.1.3 Step 5 for instructions.

5.1.3 Using the GitHub branch in IntelliJ

After cloning the VASL source code into an IntelliJ project, the IntelliJ IDE screen will show the project name and also the name of the GitHub location from which the code has been cloned and to which changes to the code should be "pushed". If this says 'develop' then right-click on 'develop' (or click on a dropdown if available) and see the options available to Checkout the branch created in Section 3.1.3 Step 5.

5.1.4 Committing ("pushing") code changes made in IntelliJ to your branch on GitHub

Once you have cloned the code repository and created a project within the IDE, you can now make changes to the code.

Code changes should be pushed to a "branch" in GitHub (see Sections 5.1.1 and 5.1.2) and not to "develop".

Before creating and pushing a Commit to GitHub, check that your local project is up to date with changes in your GitHub branch by right-clicking on the GitHub branch name (or clicking the dropdown) and selecting Pull if the green arrow is visible and enabled.

To integrate your changes into your branch in the VASL code repository in Git:

1. If you wish to add new files to the repository, first right-click on the file in the project browser and use the "Git →Add" menu option.
2. Select Git -> Commit from the menu system or click the Commit window icon on one of the sidebars. This will display a list of changed files.
3. Select the files that you wish to "push" to GitHub. Be sure to add a Comment to enable others to understand what changes you are committing. Developers should include the issue number (from GitHub) in the commit message.
4. Click "Commit and Push".

6.0 BUILDING THE VASL MODULE: FROM MAVEN

6.1 BUILDING THE VASL MODULE

Building the VASL module is not required to test the impact of your changes on the overall code. You need to build the VASL module if you modified any of the files in the "dist" folder (note there's no code there).

6.1.1 Building VASL: why you need it and what you need to do

You should build the VASL module if you added or changed files in the “dist” folder. This requires the use of software called Maven, which must be installed and then used to create the VASL module. You can also build the VASL module to provide a source module for the Configurations settings in your IDE – see Section 4.1.3 Step 4.

6.1.2 Getting and setting up Maven

Maven can be downloaded from https://maven.apache.org/download.cgi. Once downloaded, click on the Install link on the page and then follow the install instructions. While not difficult, there are several particular steps that must be followed, and it is preferable to use the instructions on the Maven site.

For convenience add the Maven bin folder to your Path. This will allow you to build the VASL module from the command line and the terminal window in IntelliJ. To confirm Maven has been properly installed type “mvn –v” at the command line to display the installed version. Adding items to your computer’s Path varies depending on OS so consult the Internet for details.

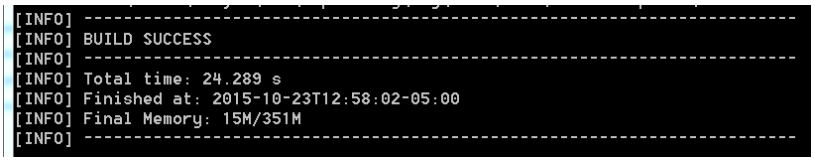
6.1.3 Building the VASL module from Maven

1. After installing Maven, go to the command line. How you do this will depend on your operating system. Check its help system or the web for details. In Windows, for example, press the Winkey + R then type cmd in the dialog box that appears.

2. At the command line, change the directory to your project directory. To do so in Windows, type “chdir DirectoryName” at the command prompt where DirectoryName is the project directory (full path required).

   Example: chdir C:\Users\myusername\IdeaProjects\STtest

3. To create the VASL module, type "mvnw clean packagel" at the command prompt. This process may take several minutes. If successful, the final messages displayed will be similar to this:



4. The build creates a .jar file in the \target directory of the project directory. This .jar file can be used to in the “Program arguments” text box referred to in Section 4.1.3.

Update History

September, 2016	Minor text edits
March, 2017	Minor text edits related to Idea IDE changes
March, 2018	Minor text edits related to Idea IDE changes
June, 2019	Clarifications and updates
April, 2020	Clarifications and updates
November, 2023	Major rewrite