

# Table of Contents

Preparation .....	1
-------------------	---

## Preparation

The permutations and combinations possible in board games are nearly infinite: there are board games with a single board, and those with multiple boards; games with many simple pieces and games where the behavior of the pieces is very complex.

As a result, it's up to you to select the proper tools needed for the game you want to make.

### Supported Game Types

The world of board games is vast, and VASSAL can accommodate a huge number of games. VASSAL supports any of the following game types:

- Traditional board games such as chess, checkers, *Monopoly*, or *Risk*.
- Hex-and-counter, block, and card-driven war games.
- Hobby games, such as Eurogames.
- Card games (traditional, collectible, or limited).
- Role-playing games that use a tactical map.

This list is not exhaustive; most any type of board-based game could be played on VASSAL.

VASSAL is also an excellent platform on which to playtest new game designs. A game designer has instant access to a worldwide audience of playtesters. No physical game sets need to be printed or distributed. It's easy to add or modify features of the game during the process of game development, and feedback can be obtained in real time.

**Live and PBEM Games:** One advantage VASSAL has over many Internet board game applications is its support of live play. You can log into the VASSAL server and play opponents in real time. In addition, VASSAL can also be used for Play by Email (PBEM) games. You can even switch between the two for the same game. There are no differences in design between modules played by email and modules played live, although some features may improve game play for one style or the other.

**Platforms Supported:** VASSAL games can be played on Windows, Mac OS X and Linux platforms. Further, because of VASSAL's Java architecture, players on different platforms can play against one another without regard to platform. If you have a Mac, and a friend runs Windows, you can play any module against one another.

### Scoping a Game

One step that will make building your module much easier is proper preparation. Before you begin the design process, it pays to take time to scope your chosen game. The complete design of a module can take anywhere from a few hours to a few weeks or more. A little planning beforehand can make your module easier to build, easier to create, and easier to maintain later.

At a minimum, every board game has a board and pieces. Everything else is negotiable.

Before even opening VASSAL, some questions to ask include:

- **Rules:** What are the rules of the game?
- **Flow:** What's the basic flow of play in the game? What's the goal of the game? What are the sides?
- **Gameplay Requirements:** Are random results needed for the game? How are these results generated? If dice need to be rolled, what kind of dice are rolled, and how many? Are turns tracked in the game? (For example, in many war games, turns are numbered; but in Monopoly, turns need not be tracked.) Are there limited pieces, or unlimited pieces, or some mix of the two? Is this a tactical game? If so, will players quickly need to determine the range between counters? Will players need a private area for personal possessions, such as cards, tokens, or, units?
- **Graphic Requirements:** What graphics will you use? How many maps does the game need? How will the counter images be generated? Will you need to prepare charts or other play aids?
- **Other Requirements:** Is there any special functionality or rules in the game? Will VASSAL be able to handle them?

#### *\*Examples of Game Scope\**

Shown here are some simple examples of game scope. Evaluating the elements of your game in detail will make it easier to determine the required components in your module later.

#### *\*Preparation: Graphics Files in Your Module\**

### **Chess**

Chess is played on a single board and has two players. There are 6 kinds of pieces, in two colors, in limited quantities. There are no random results and turns are not tracked. The pieces have no special abilities, but are deleted from the game after being captured.

### **Small Wargame**

- A. typical hex-and-counter wargame depicting a single Napoleonic battle may have the following scope:
- Two players.
  - A single map of the battlefield.
  - Turns are tracked.
  - Six-sided dice are used to resolve battles.
  - Limited pieces—units are placed at game start in fixed locations directly on the game board.
  - On Turn 5, the French player receives limited reinforcements; a place will be needed to keep these reinforcements until they are ready to enter.

### **Monster Wargame**

An ambitious game depicting the entirety of World War II in the Pacific may have this scope:

- There are multiple large maps, depicting several theaters of operation.
- There are multiple sides.
- Players may deploy unlimited pieces, in several unit types and nationalities.
- There are dozens of different pieces available to each side.
- Game money is spent to construct and improve units. The money is in the form of paper certificates that players exchange with the "bank." Players will need a place to keep their unspent money and units before they deploy them.
- Unit counters can be improved through training, or be depleted by damage in combat.
- Leader counters will work differently from unit counters. Instead of being depleted, leaders are killed (removed from the game).
- Turns are tracked, and phases and segments are tracked in each turn.
- The game uses six-sided and ten-sided dice to resolve game results.

### **Card Game**

A card game might have this scope:

- No map image is needed, but a common space is needed to place cards (a "table").
- There are multiple decks, each accessible only to certain players.
- Players will need a place to store their private hands and keep them secret from other players.
- Card decks will be needed: two draw decks and a discard pile. Cards will sometimes need to be facedown or face-up.
- Turns are not recorded, but at the end of each turn, played Cards will be moved to the discard pile. It would be nice to do this automatically.

The possibilities for a game's scope are infinite. As a result, the burden is on you to determine how best to assemble your module, using the tools at hand.

### **Graphics Files in Your Module**

A simple module may have just a few graphic images. A more typical module would require dozens or even hundreds of distinct images.

You need to create, scan, or otherwise acquire the graphics files to be included in your module. Graphic file requirements for a module can include:

- Game boards (for one or more boards)

*\*Preparation: Graphics Files in Your Module\**

- Game pieces (for counters, cards, markers, and other game tokens)
- Charts (for tables and game aids)
- Button icons

VASSAL has a limited set of graphics files available for use in building modules. These include a small set of default icons, which you can use for buttons. In addition, you can create a limited set of pieces, using NATO military symbols. See page 69 for more details.

### *\*Graphic File Support\**

VASSAL supports graphic files in SVG, PNG, GIF, and JPG formats. These are listed in order of preference, with SVG and PNG files being recommended over the other types. SVG and PNG files are the most scalable and reliable, GIF files less so, while using JPG files can cause graphics issues with the display of your module.

### *\*Graphic Filenames\**

When working with graphic files, consider these points:

**Unique Names:** Even if graphic files come from different locations on your hard drive, once added to your module, they are stored in a common folder. A graphic file added to a module that has the same filename as an existing file will overwrite any existing file. Accordingly, you should make sure all of your graphic files are named uniquely, in order to avoid overwriting existing files.

*In some cases, such as when updating a module, overwriting existing files may be desired. See page 110.*

**Naming Convention:** You should establish a standard pattern for graphic filenames. This will help when finding, replacing, or updating your graphics files later on, particularly in modules with many individual files.

For example, in a World War II game, with pieces divided by nationality, division, and unit type, and potentially hundreds of graphic images, you might use this system to help organize the image files:

(3 letter national abbreviation)(Division #)(Unit Type)(Identifier).png.

Examples of resulting filenames from this system could be:

- GerDiv1Inf3.png: A PNG image for German Division 1, Infantry Type 3.
- AmeDiv2Arm4.png: A PNG image for American Division 2, Armor Type 4. Of course, you can decide on any naming convention that fits your module best.

### *\*Graphic Dimensions\**

The dimensions of your graphics are an important factor in determining the performance impact of your module. A module with many sizeable graphics can cause significant performance delays on player systems. In addition, large graphic images can be awkward to manipulate on many computer screens.

While there is no upper size limit to the dimensions of graphics you can use in your module, but for best results, it's suggested you adhere to the following guidelines.

- **Main Boards:** A typical main board is usually 2000-3000 pixels in its longest dimension, and generally under 5000 pixels maximum. If a board graphic must be larger, consider breaking up

the board into two or more smaller boards and re-allocating screen real estate. A very large map can be awkward to view on a screen, and will have a major impact on system performance. For example, if the physical game includes a game map, a space for cards, and game tables printed on the map, you could consider moving the card space to a Map Window and the game tables to Chart windows. (In addition, a module can include tools to enable players to re-scale their view of the map on screen, which can mitigate the limitations of a small map.)

- **Other Boards:** Depending on their purpose, other Boards are usually smaller than the main Board. For example, a Private Window intended to hold a player's private pieces could be much smaller than the main Board, perhaps 500 pixels across.
- **Pieces:** Pieces, obviously, must be scaled to fit your maps. In particular, if you use a Grid on the map, the pieces must be appropriately scaled for the Grid cells. Most pieces like tokens and counters are between 50-100 pixels across. (Some pieces, like cards or money tokens, are usually larger than ordinary pieces, as they are in physical games. Cards are usually between 200-500 pixels across.)
- **Charts:** Chart graphics are typically from 500-1000 pixels across. (There is no Zoom function for most charts, so for best use, they need to fit easily on most computer screens at full size.)

#### *\*Preparation: Help and Text Files\**

- **Icons:** Button icons can be any size. There is no upper or lower limit on dimensions, but 10-50 pixels is probably the most useful size. Test the visual quality of your icons so you can decide on a common, compatible size for your buttons. Icon buttons need not be all the same size, but they should be sized to be easily visible and accessible by your players.

#### *\*Non-Rectangular Graphics\**

Most graphics used in games (for example, map, counter, and card images) are rectangular (or square). However, your graphics need not be rectangular if you make use of transparency in creating the files. Both PNG and GIF files support transparency.

For example, to make a circular image for a coin counter, create the coin image as an ordinary, rectangular PNG file in your image editor. Any portion of the image outside of the circular coin portion would need to be marked as transparent using the image editor. When the image is added to a game piece, the counter will look like a circular coin, with no empty space around it.

*Game Pieces can include an optional Trait, Non-Rectangular, which can make using non-rectangular graphics easier for players. See page 55 for more information.*

#### *\*Performance Impact\**

A module is not limited by size on a disk; it is limited by memory space available in RAM only. In general, the graphics used in a module are the biggest driver of memory usage, requiring 4 bytes per pixel for each image that is currently being displayed. Gauge the performance impact of your module accordingly.

For example: A map measuring 2000x3000 pixels is displayed with 20 counters on the map that measure 50x50 pixels each.

The total RAM required equals  $(2000 \times 3000 \times 4) + (20 \times (50 \times 50 \times 4)) = 24,200,000$  bytes, or approximately 254 MB of RAM.

## Help and Text Files

Modules can include any number of help and text files. These files can be used for various purposes, such as:

- To supply help on how to use the module.
- To give credits and acknowledgements for the design of the module.
- To provide rules, rules summaries, or important charts.

Such files will need to be created in the HTML or text file editor of your choice. It's a good idea to create the necessary files before designing the module. This will make the module design process go more smoothly.

For more information on help files, see page 99.

## Additional Tools

The following tools may be useful to have on hand when designing a module:

- **An image editor application:** An image editor will be helpful, to create graphics or manipulate and crop scanned artwork.
- **A text or HTML editor:** A text or HTML editor will be needed for the creation of text or HTML help files.
- **A scanner:** A flatbed scanner is useful for scanning game art, such as maps, counters and cards.
- **A Java compiler (for module developers):** In most cases, and for most games, custom coding will *not* be necessary for the creation of a module, and *no* Java programming skills will be needed. VASSAL is flexible and powerful enough to handle the vast majority of available games without any coding skills. However, a highly automated module may require custom Java coding. In this case, a Java compiler may be necessary for the creation of custom classes. A discussion of such custom coding is beyond the scope of this guide.