

Writing the Game XML File

Now that you've made your images, you can create the XML file to tell me how they're used.

Before we get into the precise details, it's helpful if you understand how the underlying board game model works. In EveryGame, there are 3 kinds of objects: Boards, Location, and Pieces.

In EveryGame, a piece is anything that a player can move. Each piece has a number of sides, each with it's own image. So in chess, each pawn is a one-sided piece. Checkers are two sided pieces (normal and kinged). Playing cards are also two-sided pieces (face up / face down). A dice is a six sided piece. Dragging pieces lets you move them around the board, and tapping on them lets you increment, set, or randomize the side. Pieces can also be labeled with a type to allow them only to be placed on certain locations.

Locations are places on the board that hold pieces. There are locations of various types. For instance "cells" hold a single piece, and would be used for the squares on a chess board. "Stacks" and "Queues" hold piles of pieces, and would be used for decks of cards. They could also be used as storage for captured pieces, which would need a location of their own. Locations also can perform actions upon pieces as they're inserted or removed, and can display score values based on the pieces they contain. They may choose what types of Pieces to accept.

Finally, Boards are the large, immovable background images displayed. This includes the main board displayed on the iPad, plus any satellite hands hidden off to the side if needed. The Board's only job is to display the image. Locations exist as coordinates on a particular board.

All of these objects will be written into an XML file. If you've ever used written a .html web page, you'll find it very similar. An XML file is a bit like a set of nested boxes. If you apply a setting to a box, it also gets applied to all of the smaller boxes that it contains.

Start a file called "[gamename]\_game.xml" using either the template or a similar game.

Now we make our first box. Put this line at the top of the file:

<game xml\_version="1.1">

And this line at the very bottom:

</game>

The first line tells EveryGame that we're describing a game that uses the 1.1 xml version. The second line tells it that we're done describing the game. We call the first line an "open" tag, because it starts the setting, and the second line a "close" tag, because it's the end of it. The open tag must ALWAYS be paired with a corresponding close tag. Note that this close tag has the extra slash in front of its name, and never contains any settings. Also, note where the quotation marks go. It's important that the name of the setting not be in quotes, but that the value of the setting (after the equals sign) is.

We will place all of Boards, Locations, and Pieces in between the open and close game tags. Normally we start with Boards. These represent the surfaces the game is played on. The main board, and each players hand will have its background image specified like this:

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
</game>

We use the <sides> tag to specify the name of the background image. That image will be applied to everything between the open and close tag. In this case, we put in a Board named "Main Board." This will serve as the background image for our game. Further boards added would show up as tabs on the side of the screen.

Now we continue with Locations. If we were making chess, we might add these two lines to the file, in between the game tags:

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 </location\_type>  
</game>

This tells EveryGame that every location defined between those two tags will be of type cell (that it holds a single piece). We can add use the size tag to describe how many pixels large the locations should be. Let's make them 100x100.

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 </size>  
 </location\_type>  
</game>

Note that each of these goes completely inside. The following would cause an error when it loads, since the location\_type tag is closed, before the tags it contains has closed. Using tabs can help to keep your levels lined up.

Don't do this!

<location\_type type="cell">  
 <size width="100" height="100">  
</location\_type>  
</size>

Finally, we can place a couple of locations inside of our size tags. For each location, we define its upper-left corner via the corner\_coord tag (with (0,0) the upper left of the screen). This could be determined by looking at your [game\_name]\_board.gif image. Then we use the location tag to actually create the Location. Any text between the location open tag and the location close tag will become the location's name.

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 <corner\_coord board="0" x="0" y="0">  
 <location>LEFT\_SQUARE</location>  
 </corner\_coord>  
 <corner\_coord board="0" x="100" y="0">  
 <location>RIGHT\_SQUARE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
</game>

We now have two locations to hold pieces, name LEFT\_SQUARE and RIGHT\_SQUARE. Their upper left corners are at pixel (0,0) and (0,100) respectively on the main iPad board (board 0). They're both sized 100x100 pixels and of type cell. Lets add a third location. We'll make this one a stack, so that it can hold many pieces. Since it's a new type of Location, it must go after the location\_type=cell tag is closed:

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 <corner\_coord board="0" x="0" y="0">  
 <location>LEFT\_SQUARE</location>  
 </corner\_coord>  
 <corner\_coord board="0" x="100" y="0">  
 <location>RIGHT\_SQUARE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
 <location\_type type="stack">  
 <size width="200" height="100">  
 <corner\_coord board="0" x="0" y="100">  
 <location>BOTTOM\_RECTANGLE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
</game>

This third Location, named BOTTOM\_RECTANGLE is 200x100 pixels, has it's upper left corner at pixel (0,100) on the main iPad board, and can hold many Pieces.

Now, we will create some Pieces, using a process similar to the locations. Let's say that we want a dice that is rolled when tapped, and set back to "1" when double tapped. Let's put in tags that define the behavior for single and double taps. We add nested open and close tags for a single and double tap underneath the locations. Note that the first side is actually number "0".

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 <corner\_coord board="0" x="0" y="0">  
 <location>LEFT\_SQUARE</location>  
 </corner\_coord>  
 <corner\_coord board="0" x="100" y="0">  
 <location>RIGHT\_SQUARE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
 <location\_type type="stack">  
 <size width="200" height="100">  
 <corner\_coord board="0" x="0" y="100">  
 <location>BOTTOM\_RECTANGLE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
  
 <single\_tap action="random\_side" args="v2,0\_1\_2\_3\_4\_5">  
 <double\_tap action="set\_side" args="0">  
  
 </double\_tap>  
 </single\_tap>  
</game>

We should also define the Location the Piece starts in. Let's use BOTTEM\_RECTANGLE. The Location picked must already have been defined, which is why we do the Locations first.

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 <corner\_coord board="0" x="0" y="0">  
 <location>LEFT\_SQUARE</location>  
 </corner\_coord>  
 <corner\_coord board="0" x="100" y="0">  
 <location>RIGHT\_SQUARE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
 <location\_type type="stack">  
 <size width="200" height="100">  
 <corner\_coord board="0" x="0" y="100">  
 <location>BOTTOM\_RECTANGLE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
  
 <single\_tap action="random\_side" args="v2,0\_1\_2\_3\_4\_5">  
 <double\_tap action="set\_side" args="0">  
 <initial\_location name="BOTTOM\_RECTANGLE">  
  
 </initial\_location>  
 </double\_tap>  
 </single\_tag>  
</game>

Finally, we add a tag listing the images to use for each side (and therefore also the number of sides), and a tag to create a piece with a given name, in the same style as the Location. Since BOTTOM\_RECT is a stack, let's add a couple of pieces

<game xml\_version="1.1">  
 <sides paths="Gamename\_board.png">  
 <board>Main board</board>  
 </sides>  
  
 <location\_type type="cell">  
 <size width="100" height="100">  
 <corner\_coord board="0" x="0" y="0">  
 <location>LEFT\_SQUARE</location>  
 </corner\_coord>  
 <corner\_coord board="0" x="100" y="0">  
 <location>RIGHT\_SQUARE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
 <location\_type type="stack">  
 <size width="200" height="100">  
 <corner\_coord board="0" x="0" y="100">  
 <location>BOTTOM\_RECTANGLE</location>  
 </corner\_coord>  
 </size>  
 </location\_type>  
  
 <single\_tap action="random\_side" args="v2,0\_1\_2\_3\_4\_5">  
 <double\_tap action="set\_side" args="0">  
 <initial\_location name="BOTTOM\_RECTANGLE">  
  
 <initial\_location name="BOTTOM\_RECTANGLE">  
 <sides  
paths="Gamename\_1.gif,Gamename\_2.gif,Gamename\_3.gif,Gamename\_4.gif,Gamename\_5.gif,Gamename\_6.gif">  
 <piece>DICE\_0</piece>  
 <piece>DICE\_1</piece>  
 </sides>  
 </initial\_location>  
 </double\_tap>  
 </single\_tap>  
</game>

Important Note

Make sure that all of your files start with the game name, and an underscore (i.e. Checkers\_board.gif, Chess\_game.xml). This is required to organize them once they're loaded back onto the iPad later.