Creating a New Game

Creating a new game to play in EveryGame is easy, there are four steps:
* Create images for the board and pieces
* Create a special text file that describes how those pieces are used
* Create a text file that describes how to set up the game
* Create a pdf file that provides game rules

To get started, copy the Template game onto your computer. You might also want to copy some of the other games. Reading though how they work might give you useful ideas for your game, and you may also find it easier to alter an existing image rather than creating an a new one.

Here's how to copy a game off of your iPad.
* Plug your iPad into your computer
* Open iTunes
* Select a your iPad from the column on the left
* Select the Apps tab
* Scroll to the bottom of Apps tab to the File Sharing
* Select EveryGame from the list of applications with shared data
* Select the game you want to copy from the Documents list on the right
* Click "Save To..." and pick a location on your computer

What follows is a step by step tutorial for creating a game. If you prefer, you can also just straight to the bottom, where there are details on the specific xml commands available to create a 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.

Creating Images

You will need to create an image for the board, and an image for each type of piece. The board background should be a .gif or .png image file 1024x768 pixels large, called [gamename]_board.gif. It's probably the easiest to modify the one in the Template, or a similar existing game.

Each different way a piece can look will need an image. For instance, you'd need four images for checkers (red normal, red king, black normal, black king), and 53 images for a deck of cards (one for each card, and one extra for the card backs). The resolution of these images should match the board, and their width and height in pixels should be picked to fit correctly on the board squares.

Finally, some games (like poker) have hands that are hidden off the side of the board. Make images for these the same way. Just remember that the larger the hand, the more of the board it will cover.

Writing the Game XML File

Now that you've made your images, you can create the XML file to tell me how their 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 pawn 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 size 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.

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 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_tag>
</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">
                <sides
paths="Gamename_1.gif,Gamename_2.gif,Gamename_3.gif,Gamename_4.gif,Gamename_5.gif,Gam
ename_6.gif">
                    <piece>DICE_0</piece>
                    <piece>DICE_1</piece>
                </sides>
            </initial_location>
```

```
        </double_tap>
    </single_tap>
</game>
```

You've now got the start of a working game. Looking at the XML file documentation linked will give you a better idea of all of the tags that are available for performing certain settings. Looking at other games can also be a useful guide.

Game Setup File

Some games require special setup beyond setting the initial location for a piece. In particular, games with randomized starting locations often require that the pieces be placed into a deck, shuffled, and then dealt out. In this case, we also create a file called [game_name]_init.xml. This file may be filled with action tags, which let's you encode the moves you might perform by interacting with your iPad.

First off, much like the [game_name]_game.xml file, we will create tags for the top and bottom of the file:
<moves xml_version="1.1">
</moves>

Now we create each move. Lets say we wanted to quadruple tap BOTTOM_RECTANGLE (which if we'd set it up, could shuffle the order of the items in the Location), and then move the top Piece to LEFT_SQUARE, and the second Piece to RIGHT_SQUARE. This would be easy to do in the game -- just tap 4 times, and then drag two Pieces. Let's see how we can represent the same thing in our xml file:

```
<moves xml_version="1.1">
    <action type="quad_tap_name" name="DICE_0"></action>
    <action type="move_piece" board="0" x="100" y="150" to_board="0" to_x="50"
to_y="50" />
    <action type="move_piece" board="0" x="100" y="150" to_board="0" to_x="150"
to_y="50" />
</moves>
```

There are several ways to describe moves (detailed in the xml file documentation). In this case, we tell it to first quadruple tap on a particular piece by name (DICE_0), and then to move pieces, giving the pixel coordinates that the finger started and ended at.

Last Details

Make sure that all files start with the name of your game, and an underscore -- ie [gamename]_game.xml. This is required to upload them properly to the iPad. References to the filenames inside of [gamename]_game.xml should also include the game name.

To make your game extra polished, there are a few final steps you can perform. Create a PDF file called [game_name]_rules.pdf, and store it in your game's directory. If you have a Mac, this is often easy to do from the Print window. The [game_name]_rules.pdf file will automatically be loaded when somebody taps the Rules button in the menu.

Also you could create an [game_name]_info.xml file. Just fill in the fields of the one that appears in the Template game. It's not used right now, but in a future version, it'll help people learn more about the game. Plus there's a field where you can take credit for your work.

All Done

When you're done, load your game onto you iPad, and start playing. Also if you like, email your game to Eggy.EveryGame@gmail.com. Well might include it on our website, or in a future version of EveryGame!

* Plug in your iPad
* Open iTunes

* Click on your iPad in the left hand column
* Click on the Apps tab in the right hand window
* Scroll to the File Sharing Section at the bottom
* Select EveryGame from the Apps menu on the left
* Click the Add... button below the EveryGame Documents window
* The "Choose a file" dialog box will appear
* Select all your files (the shift / command key helps here!)
* Click the "Choose" button on the dialog box
* The files will now be copied to EveryGame
* The next time that you start EveryGame, your game should be ready to play!

Troubleshooting

If the game does not work, it's likely due to a problem in the xml file. Any errors will cause it to stop loading, and only the amount read up to that point will be displayed. Check to make sure that the hierarchy is correctly set (open a close tags strictly nested), that there are no typos in the tag names, and that the filenames match the pieces named. A great way to do this is with Safari or Firefox. If you open the [gamename]_game.xml file, it will display the line number of any problems, and likely also the tag that is mis matched. Sometimes it also works to comment out as much as possible with the <!-- and --> tags, and slowly add entries back in until the error is determined.

One common problem that is seen is an all white board, or pieces that are invisible. Apart from bad tags in the xml file, these can also be caused by missing images, or filenames that differ from the one in the xml file. If an image does not appear, verify that the image has been loaded into your game, that it is prefixed with the game name, and that the name exactly matches in the <sides> tag. Copying the game folder from iTunes back to your hard drive is a good way to determine exactly what files have been loaded.

In the future, we'll be providing better feedback on where the error occurs. However if you can't find the error, you can try sending your file to Eggy.EveryGame@gmail.com. We don't have a full-time team to read the files or anything, but we'll be happy to help when we have some spare time.

New in Version 1.1

EveryGame version 1.1 expands the engine to include some new features. Many of them are designed to aid the creation of "buttons", that perform many actions (possibly on other pieces) when tapped. For instance, this block of code highlights several of them:

```
<draggable value="0">
    <single_tap action="random_side" args="v2,1_2_3_4_5_6,dice1">
        <single_tap action="move_piece" args="stock,dice1">
            <piece>button</piece>
        </single_tap>
    <single_tap>
</draggable>
```

The piece named "button" will appear on the screen, and respond to taps, but will not respond to drags (though it can still be positioned via <initial_location> or "move_piece" actions). When tapped, it will both roll and move "dice1", but do nothing to itself. This opens up a lot of possibilities, like one button to roll multiple dice, etc.

The other new tags added are:
`<capture_location> and <bounce_location>`
these tags are placed onto pieces, and specify what should happen if a piece is moved onto an occupied cell. First, the moving piece's capture_location is checked. If it exists, the piece being moved onto is moved there, freeing the square. If that is not possible, the game will attempt to move the stationary piece to its bounce_location. If neither of these are possible, the move will not occur.

See the XML documentation below for additional specifications on all of these features

Requests

EveryGame uses XML files to specify how the player can interact with pieces and the board. The model allows for a wide array of games, but you might find that some game you would like requires some feature not yet implemented. If this is the case, email Eggy.EveryGame@gmail.com, and let us know. We've got some ideas for new tags we'll be adding soon, and we'll try to add yours in a future version.

XML Documentation

This lists all of the possible tags and arguments that EveryGame accepts.

```
<game>
    xml_version=[float]
        "Version of the program needed to view this xml file"

<single_tap>
<double_tap>
<triple_tap>
<quad_tap>
    action=[string]
        "increment_side"
            args=[comma separated string]
                  "wrap" (default cap)
                  "target" (name of a piece to apply to.  Default self.  Version 1.1)
        "decrement_side"
            args=[comma separated string]
                  "wrap" (default cap)
                  "target" (name of a piece to apply to.  Default self.  Version 1.1)
        "set_side"
            args=[comma separated string]
                  "int" (side to set to)
                  "target" (name of a piece to apply to.  Default self.  Version 1.1)
        "random_side"
            args=[comma separated string]
                  "v2" (first argument of 'v2' specifies new format.  Version 1.1"
                  "underscore separated int list" (ie "2_3_4", random sides to choose
between. default=all.)
                  "target" (name of a piece to apply to.  Default self.  Version 1.1)
        "move_piece"
            args=[comma separated string]
                  "location name to move piece to"
                  "target" (name of a piece to apply to.  Default self.  Version 1.1)
        "shuffle_location"
            args: No args
    args=[string list]
        "Comma separated list.  Any combination from above allowed"

<piece_type>
    type = [string]
            "Identifier that must match location's valid_types"

<sides>
    paths = [string list]
            "A list of file names"

<initial_side>
    value = [int]
```

```
<points>
    value = [int]

<draggable>
    (version 1.1)
    value = [int]
        "0: Not draggable.  1: draggable (default)"

<initial_location>
    name = [string]
            "The name of a location"

<capture_location>
    (version 1.1)
    name = [string]
            "The name of a location or 'swap'.  If moved to an occupied cell, piece
currently in the cell will move here"
            "If name='swap', the two pieces will trade spaces instead.  Takes
precedence over <bounce_location>"

<bounce_location>
    (version 1.1)
    name = [string]
            "The name of a location or 'swap'.  Piece will move here if in a cell,
and another piece moves in"
            "If name='swap', the two pieces will trade spaces instead"

<valid_types>
    types = [string list]
        "List of identifiers that one of which must match piece"

<corner_coord>
    board = [int]
    x = [int]
    y = [int]

<size>
    width = [int]
    height = [int]

<location_type>
    type = [string]
        "stack"
        "queue"
        "cell"

<location_display>
    type = [string]
        "no_display"
        "count"
        "score"

<location_in_action>
<location_out_action>
    See <single_tap>
    action=[string]
    args=[string list]

<piece>...</piece>
    "Create a Piece named ..."
```

```
<location>...</location>
    "Create a Location named ..."
<board>...</board>
    "Create a Board named ..."
    "The first Board is shown as the game background.  Further Boards show up as
tabs"


<action>
    type = [string]
        random_seed
            value = [int]
        single_tap
        double_tap
        triple_tap
        quad_tap
            x = [int]
            y = [int]
            board = [int]
        move_piece
            x = [int]
            y = [int]
            board = [int]
            to_x = [int]
            to_y = [int]
            to_board = [int]
        single_tap_name
        double_tap_name
        triple_tap_name
        quad_tap_name
            name = [string]
                "The name of a piece to tap"
        move_piece_name
            name = [string]
                 "The name of the piece to move"
            from = [string]
                  "The name of the source location"
            to = [string]
                  "The name of the destination location"
```