**Moonscraper Chart Editor:**

***User Interface Art and Design Specifications***

Introduction-

*Moonscraper Chart Editor* is a modding tool used to create custom songs for *Guitar Hero*-based rhythm games that can be used with any programs that process chart or midi files. Examples of these games include *Guitar Hero 3* (via GHTCP), *Phase Shift*, *Frets on Fire* and *Clone Hero*. It is made with the Unity game engine.

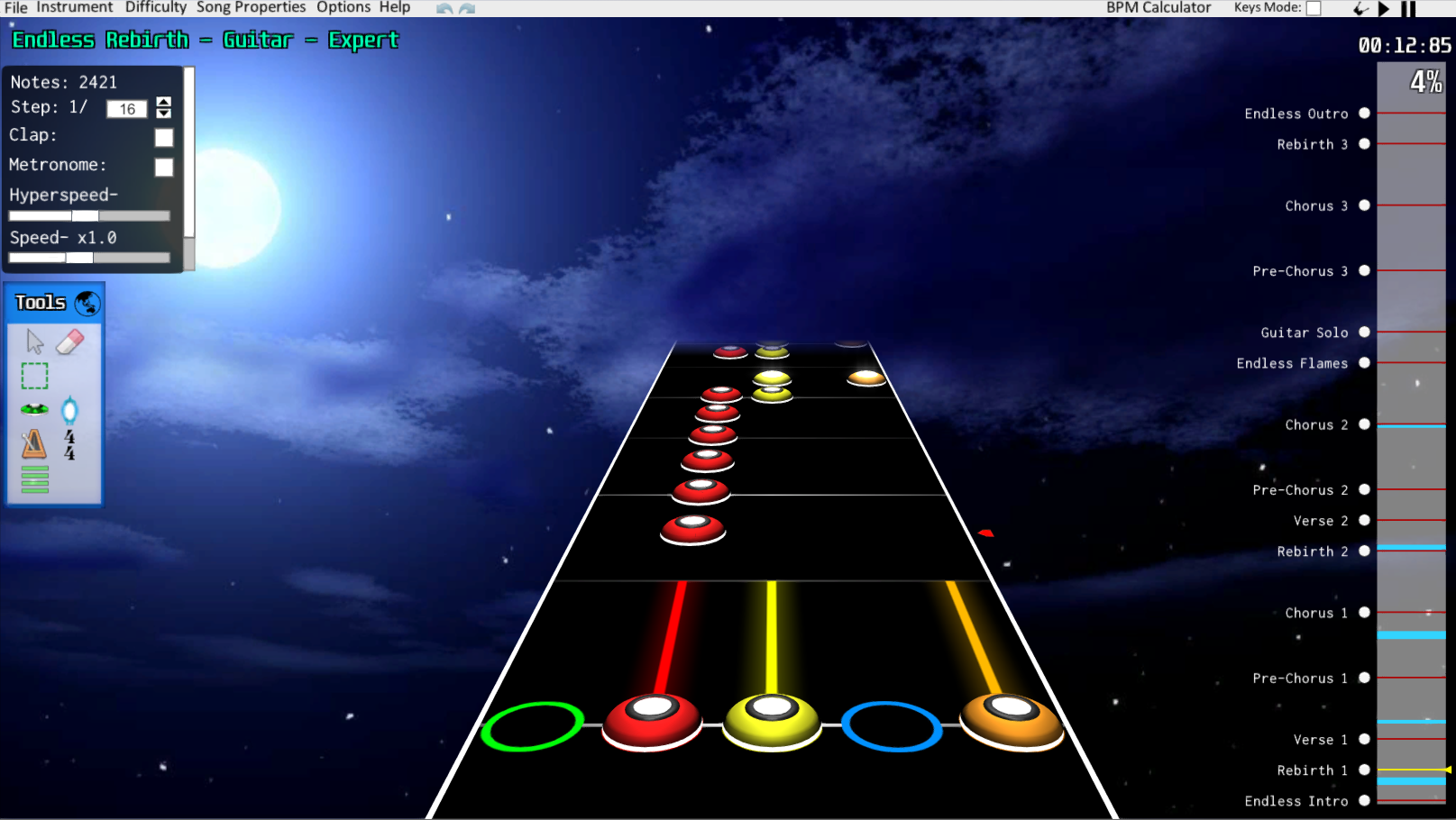


Fig. 1 with default assets

The name “Moonscraper” and the icon was originally based on the winged rabbit, or Wolpertinger, featured in the pause menu in *Guitar Hero 3* (see fig. 2) in conjunction with the rabbit in the moon folklore to produce figure 3 as the icon for this program.



Fig. 2

Fig. 3

Contents-

* Important note
* Before Starting: 9-slicing
* Main assets needed
  + Background panels
    - Properties
    - Tools
    - Inspectors
    - Menus
    - Gameplay stats
  + Standard GUI
    - Buttons
    - Sliders
    - Toggles
    - Dropdowns
  + Timeline
* Optional

**Important note**

Moonscraper features customisability in some of its assets. These can be replaced by users by dropping files into the “Custom Resources” folder provided along with the rest of the program files. Most notably for this context would be the background image, as it can be swapped out for any jpg or png file the user desires, and may affect the readability of text and colour clashing with some of the UI. The default background that loads if the user does not specify a picture is the background featured in figure 1, so use this as a based if you so have to.

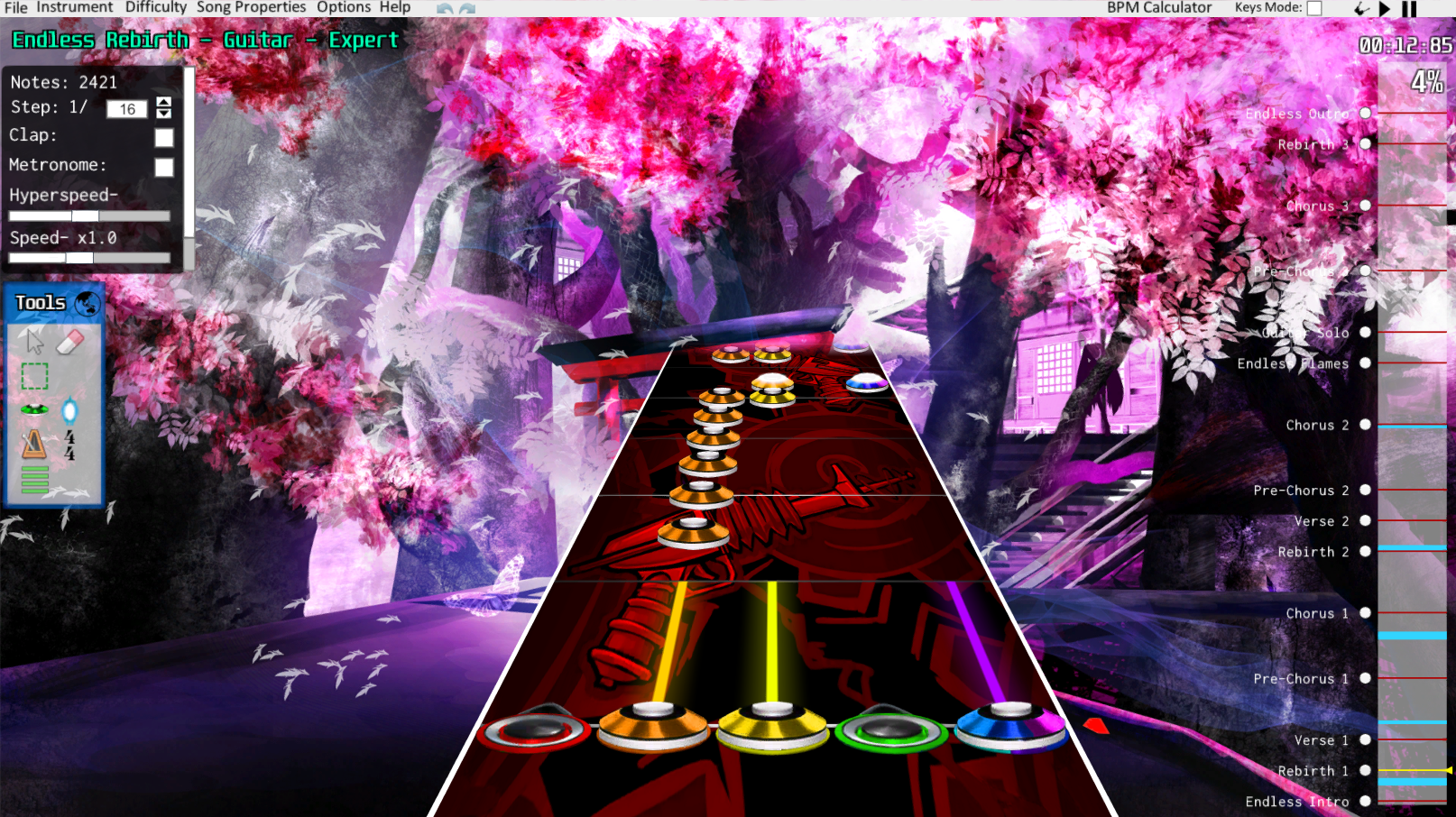


Fig. 4, Moonscraper screenshot using custom notes, background and fretboard

**Before starting: 9-slicing**

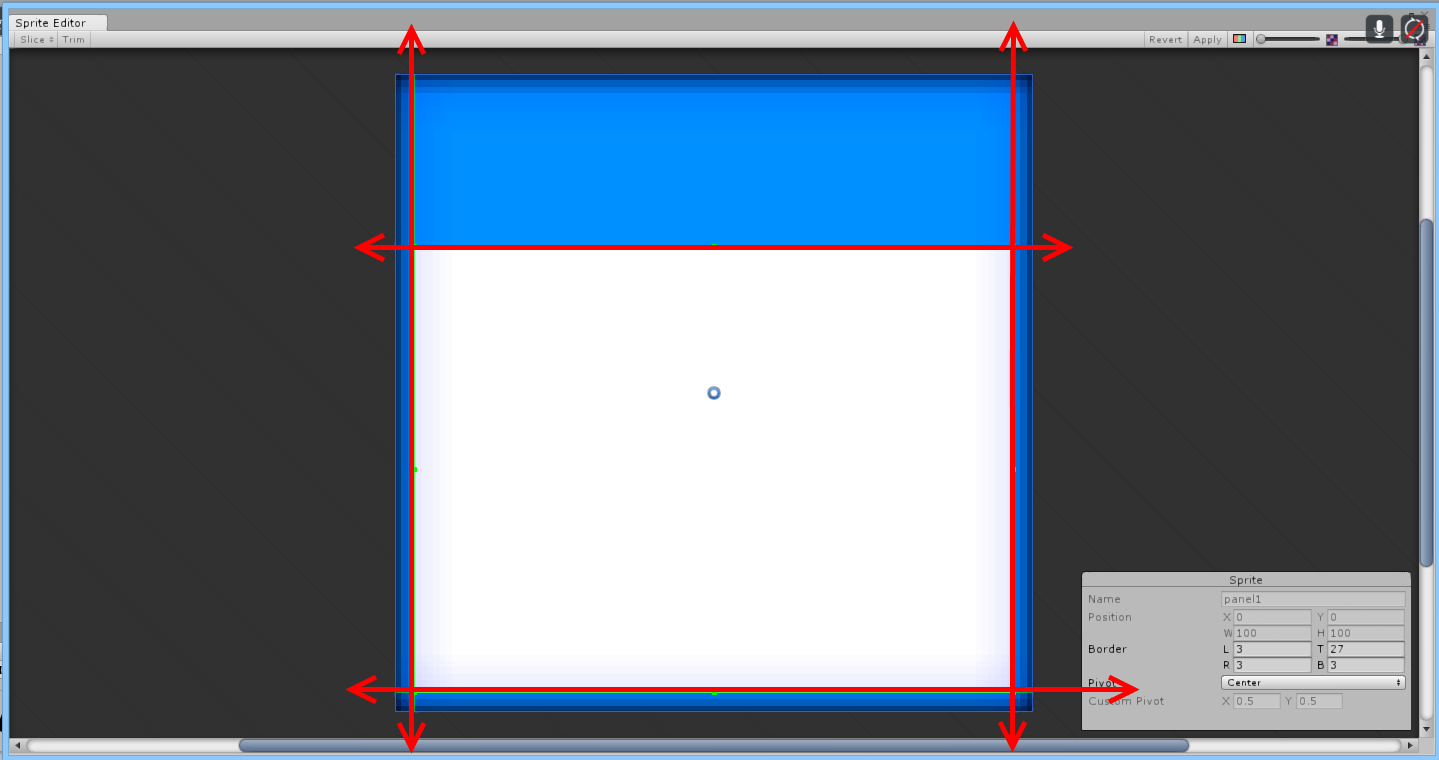
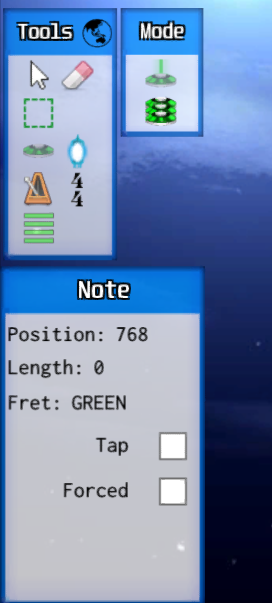


Fig. 5

If you play around with the program, you’ll notice that some of the background panels are the same, but are scaled at different sizes (see figure 6). This re-use of the same image, scaled to different sizes without distortion, is achieved through what is known as 9-slicing. It essentially allows the corners of an image to remain unscaled, the top and bottom only scale horizontally, the left and right only scaling vertically, while the centre is free to stretch as much as it wants in order to achieve smooth stretching of UI elements.

This slicing does not happen to the image itself, rather the image just needs to be designed in a way that can be 9-sliced later on if re-use and re-scaling of the image is desired while preserving smooth corners and borders. The actual slicing occurs automatically in development, which is out of scope of this document.

Fig. 6

**Main Assets Needed**

**Background panels:**

Background panels only require a single image, which is then 9-sliced by the game engine so it can be re-used at different sizes.

**Properties panel-** Currently just using the standard default with a tint and alpha effects from the standard shader.

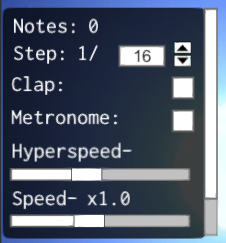
**

Fig. 7

**Tools panel**- Currently using the image displayed in figure 5. Used to hold all the different tools such as placing notes, adding sections, etc. A secondary panel pops up labeled “mode” (fig. 9) if the note tool is selected and keyboard controls are enabled, which currently shares the same background panel at a different scale.

**

Fig. 9

Fig. 8

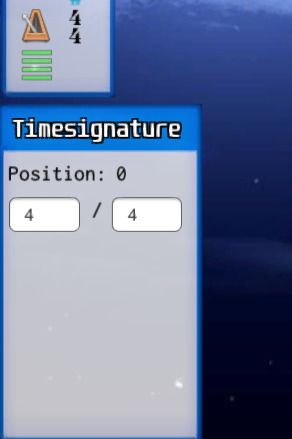
**Inspectors-** These panels appear in the bottom left when an object is clicked on to be inspected and/or edited, like a note or a time signature. They all share the background panel from figure 5 and the tools panel, mostly out of laziness out of creating a new panel. It doesn’t matter if the new panel for this is the same as the tools panel or not if it still looks fine. 

Fig. 11

Fig. 10

**Menu panel-** This panel is mainly for all the menus, such as settings, about, credits, song properties, etc. Pictured in figure 12 is the current panel being used, with examples provided in the settings and song properties menus (fig 13 and 14 respectively).

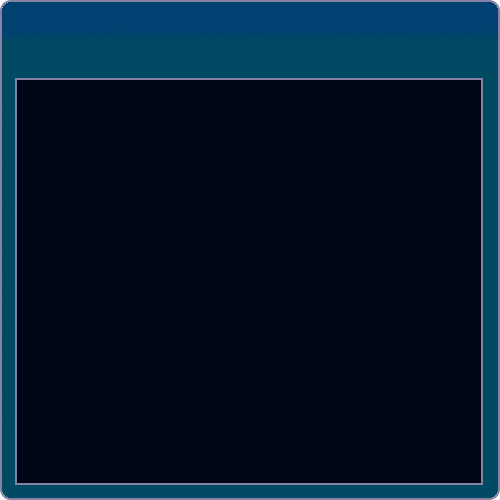


Fig. 12

Fig. 13

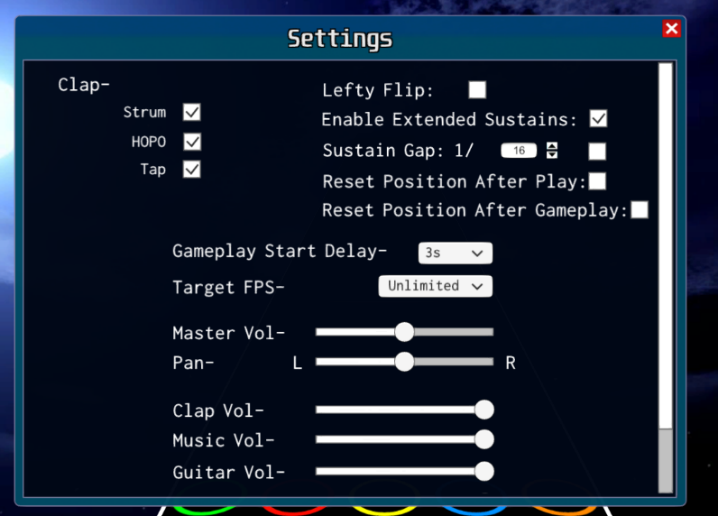


Fig. 14

**Gameplay Stats-** This panel appears when the user enters the gameplay mode in order to playtest their creations. It keeps track of the total notes they’ve hit, the total notes that have appeared and how long their last note-streak was.



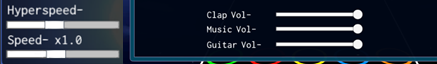
**Standard GUI:**

**Buttons-** Your standard click and a thing happens button. TODO

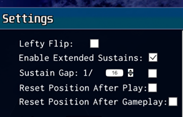
**Scrollbar-** Used for content navigation, typically for scrolling down a page, the scrollbar uses 2 sprites: a background image that displays the full scrolling area and the actual handle that indicates how far down a user is into a scroll. Will only need one single set to apply to all scrollbars. If in your honest opinion you feel these current scrollbars work anyway feel free to leave them as is.

****

**Sliders-** Used to select a specific value, a slider uses 3 sprites: one for the background, one for the handle and one for the filled area. The fill area is an optional sprite and does not necessarily need to be required as it depends on the look of the slider desired.



**Toggles-** Unity’s default toggle is currently a checkmark that uses 2 sprites: one for the background image box and one for the actual checkmark that appear/disappears to indicate when the toggle in on/off respectively.



Total toggles in the program (sprites to be reused between most toggles obviously):

* Properties panel: Clap, Metronome
* Keys Mode
* Note inspector: Tap, Forced
* Settings: Strum, HOPO, Tap, Lefty Flip, Extended Sustains, Sustain Gap, Reset After Play, Reset After Gameplay
* Export menu: Forced, Copy Empty Diff.

**Dropdowns-**

**Optional**

Draft and suggest better ways of organising the UI on the menus, as well as suggest pretty much anything that could look better, such as fonts, colours, etc. just as long as it doesn’t change or require additional functionality