Tutorial – UI and HUD Design

This week we're going to mix things up a little. In the tutorial for this lesson we're going to make two different Heads Up Display (HUD) designs for our game using a graphics editing program.

This tutorial will use GIMP, since it's open source (i.e., free) and available on Windows, Mac and Linux. If you have a different graphics program installed on your machine (like Photoshop) then feel free to use that instead.

If you don't have GIMP and want to install it, download the latest version from http://www.gimp.org/. The download page should take you to the correct version for your operating system.

One you have your graphic editor of choice loaded up, you're ready to begin

Create a HUD:

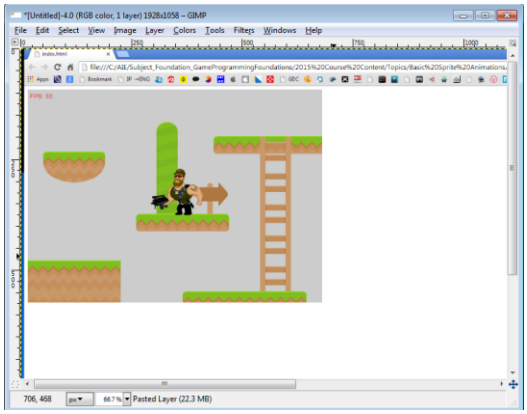
1. Start by taking a screenshot of your game as it is now. Hopefully you're already using the final sprite animations and tilesets that you want in your game, so in terms of graphics it shouldn't change too much.

   To take a screen shot in Windows, activate the window you want to capture (in this case it will be the browser window running our game) and press Alt + Print Screen.
   This captures only the active window.
   Your screen shot will be saved to the clipboard.

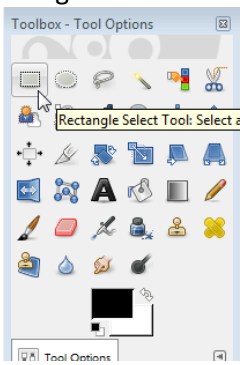
   To take a screen shot on a Mac, open the Grab program and select the *selection* option from the *Capture* menu.
   You should copy your screen shot to the clipboard.

2. In GIMP, select *File ->Create ->From Clipboard*

   This will create a new image using your screenshot

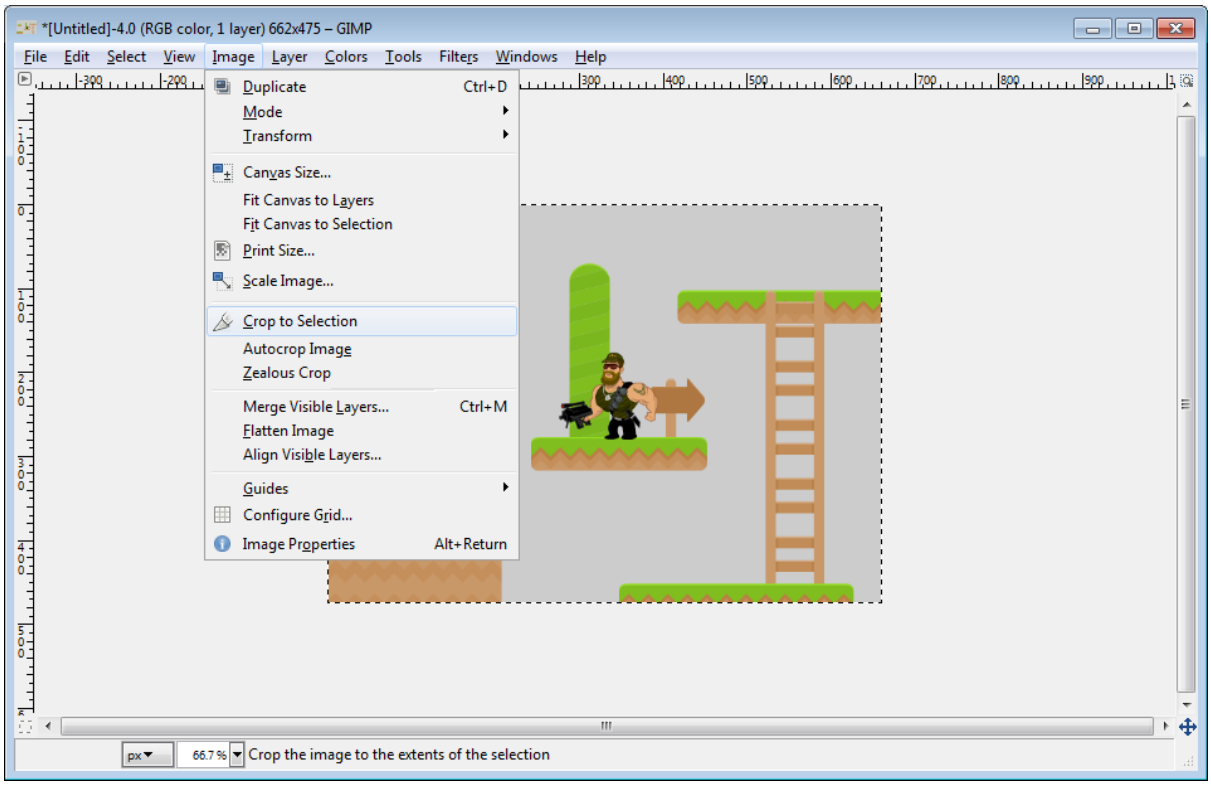


3. Using the selection tool, select just the game view.



4. With the region in your image selected, from the *Image* menu choose *Crop to Selection*.

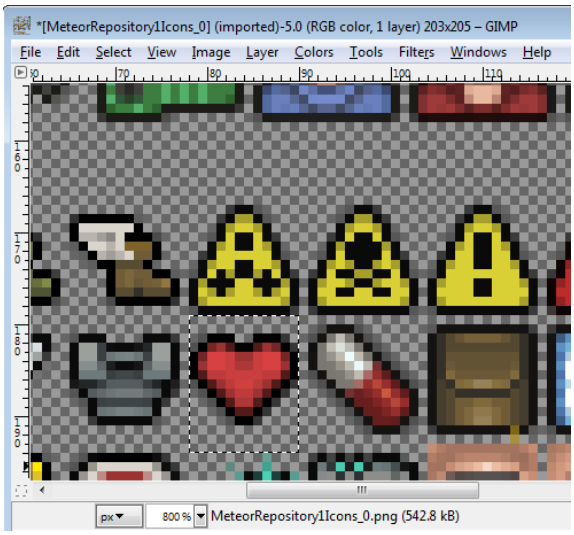
   You should now have an image that shows just your game (not the surrounding browser window)



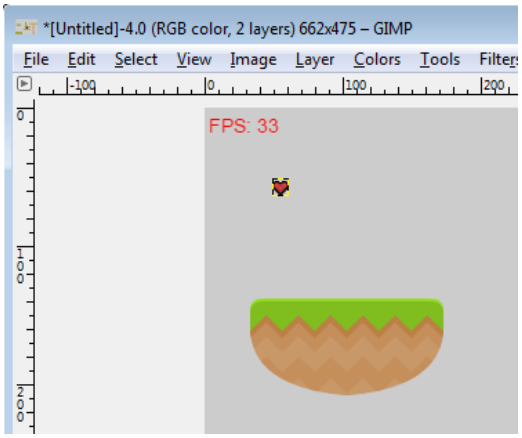
5. Search Google to find some icons and graphics that you would like to use to display information in your HUD.

   I've chosen a military icon set from OpenGameArt.org:
   http://opengameart.org/content/141-military-icons-set
   They're a little small, but we can enlarge them and they should work fine.

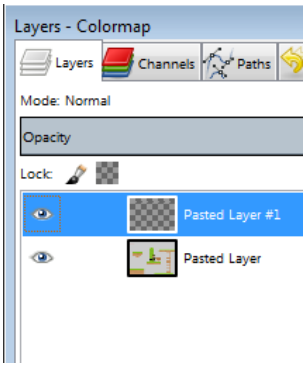
6. I've loaded my military icons in another GIMP window, and I'm going to select the images I need (using the selection tool) and then copy and paste them into the window containing my HUD mock-up



Select and copy…



and paste



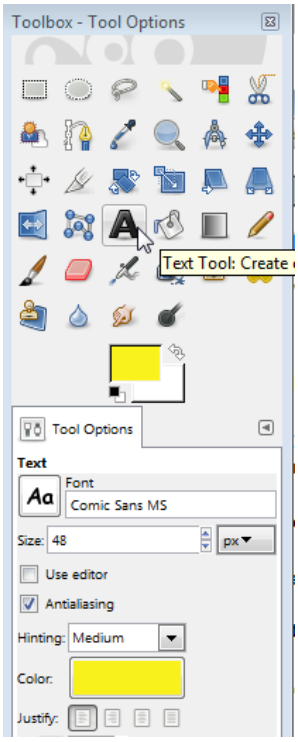
The pasted image will appear as a new layer

7. If you're pasted image is a little small (like mine is), select the layer and choose *Layer ->Scale Layer*. You can enter new pixel dimensions, or scale by a percentage.

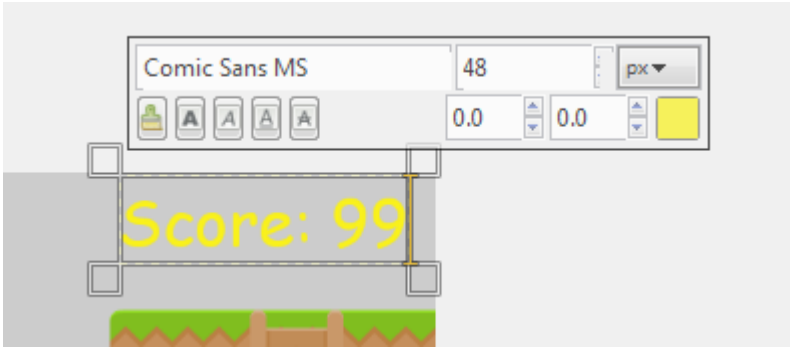
8. Copy and paste images into your HUD mock-up until you get something looking good.

   You will need to make sure that you have, at a minimum, a score and a life/health counter.

9. To insert text, select the *Text Tool* and set the appropriate font, colour and size.

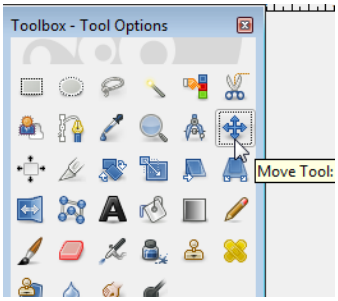


Select the *Text Tool* from the toolbox.



Click on your image where you want the text to appear and start typing.

Press *esc* when you've finished.



If your text is in the wrong position you can move it using the *Move Tool*.

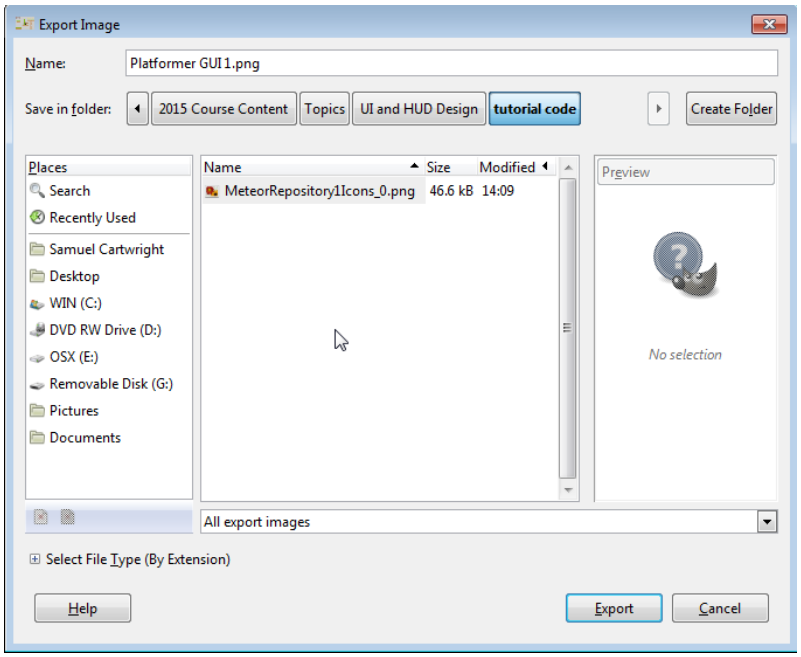
10. Once your image looks the way you want it, select *File ->Export As…*

    Enter the name and location for your image file to save to.

This will export your image as a PNG image. Note that if you export to a PNG, you won't be able to modify the PNG image – all layer information will be lost.

If you think you would like to come back and edit your image later, you should select the *Save* option from the *File* menu and save your image as a GIMP XCF image.

You will need to submit your HUD designs as PNG images.



The final HUD mock-up as a PNG image… booyeah!

Adding the HUD to your Game:

Now comes the part that's a bit trickier, actually implementing the HUD in your game.

When we think about the information the HUD is displaying, we need to think about the underlying game mechanics and how the game needs to dynamically update the information on the screen.

In my HUD mock-up I have a life counter (Chuck Norris will have 3 lives), and a score.

The score will be the easy one. It is all text and will only need to show the score variable as it changes values (that is, as our score increases the text will change and be drawn to the screen).

The hearts however require a bit more thought. When Chuck dies, we need to be able to respawn him from somewhere. Up until now we haven't thought about this.

A larger game may use checkpoints, or allow saving and loading. But for us, when Chuck dies (say, by falling off the screen), we'll respawn him at the start of the level.

Which brings us to my next point, we currently don't kill Chuck when he falls of the screen. We'll need to add that too.

And finally, if Chuck dies with no lives left, we'll need to switch to the game-over state.

So you can see that sometimes adding HUD components can affect our game's code in quite dramatic ways.

Adding the Score:

1. Create the score variable in main.js. I like to put any variable that we may need to access from another part of the game (say, from the player.js script) inside main.js, but really it

```
var score = 0;
```

   could live anywhere.
2. Next we draw the score to the screen during the run() function. If you've broken your run() function up into Update and Draw functions, then put it in the draw function.
   I've put my score text in the same general position as in my HUD design. You should change the font, colour and position according to your own design.

```
	// score
	context.fillStyle = "yellow";
	context.font="32px Arial";
	varscoreText = "Score: " + score;
	context.fillText(scoreText, SCREEN_WIDTH - 170, 35);
```

The only thing left to do will be to increment (and possibly decrement) the score according to your game logic.

Adding the Life Counter:

Our life counter is a little different from the score. Instead of drawing text, we're going to draw a series of heart images. The more hearts, the more lives the player has.

1. As with the score, add a variable to track the number of lives:

```
var lives = 3;
```

2. Now in the run() function, just below our code for drawing the score, use a loop to draw the number of hearts needed:

```
// life counter
for(vari=0; i<lives; i++)
{
context.drawImage(heartImage, 20 + ((heartImage.width+2)*i), 10);
}
```

As with the score, all that remains is to add the game logic that will decrement the life counter when the player loses a life.

Exercises:

1. Complete the implementation of the score. This will depend on the game mechanics that you have decided to implement. For example, you could increase the score every time the player shoots an enemy.

   You may find you need to add enemies, bullets, or other objects to your game before your score will work. Attempt this yourself, but remember we will also briefly cover these topics during the lesson titled 'Finishing the Platformer'.

2. Complete the implementation of the life counter. As with the score, this will depend on the game mechanics you are implementing. You player could, for example, lose a life if they fall off the screen or get hit my an enemy.