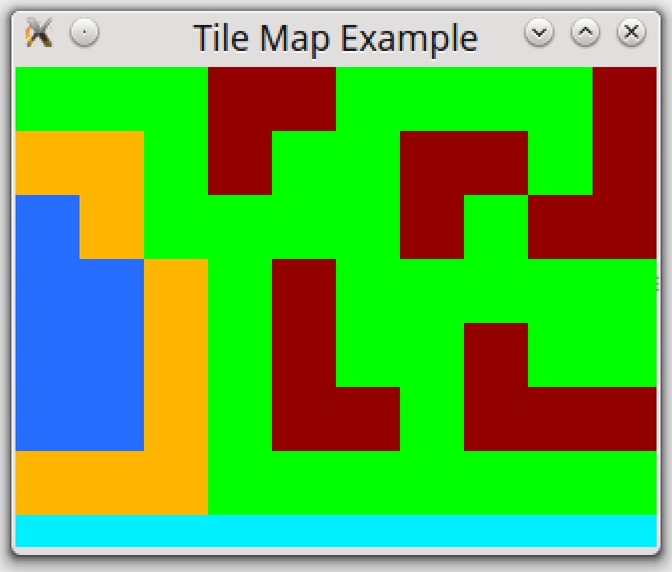
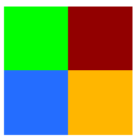
Project 2: Tiled Map Generation
Dr Alan Crispin

Many 2D and isometric role playing games (RPGs) use a playing area which consists of small rectangular, square or hexagonal graphic images, referred to as tiles. Such games are called tile-based games and there is a dedicated Wikipedia page about them http://en.wikipedia.org/wiki/Tile-based_video_game

A tile is a small image, usually rectangular or isometric, that is a building block for creating larger images. We can define a map as a grouping of tiles. In this mini project we will display a simple map using rectangular tiles as shown below.



Tile Set (tiles.png)



64x64 image created using rgbpaint (Linux paint program).
Each colour is a 32x32 square area.

Tile Map (map.txt)

```
tiles.png
0,0 0,0 0,0 1,0 1,0 0,0 0,0 0,0 0,0 1,0
1,1 1,1 0,0 1,0 0,0 0,0 1,0 1,0 0,0 1,0
0,1 1,1 0,0 0,0 0,0 0,0 1,0 0,0 1,0 1,0
0,1 1,1 1,1 0,0 0,0 0,0 0,0 0,0 0,0 0,0
0,1 0,1 1,1 0,0 1,0 0,0 0,0 0,0 0,0 0,0
0,1 0,1 1,1 0,0 1,0 0,0 0,0 1,0 0,0 0,0
0,1 0,1 1,1 0,0 1,0 1,0 0,0 1,0 1,0 1,0
1,1 1,1 1,1 0,0 0,0 0,0 0,0 0,0 0,0 0,0
0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0 0,0
```

Step 1: Setting up the project

Create a new directory for your project called “cpp-tiles”. Copy the map.txt and tiles.png files into the project folder.

Step 2: Create the source code

Using Geany create a new C++ source code file called tile.cpp in the project directory. Write the following C++ code for the project.

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <fstream>
#include <cctype>
#include <sstream>
#include <string>

int main()
{
      std::ifstream openfile("map.txt");

      sf::Texture tileTexture;
      sf::Sprite tiles;

      sf::Vector2i map[100][100];
      sf::Vector2i loadcounter=sf::Vector2i(0,0);

      if(openfile.is_open())
      {
            std::string tileLocation;
            openfile >>tileLocation;
            tileTexture.loadFromFile(tileLocation);
            tiles.setTexture(tileTexture);
            while(!openfile.eof())
            {
                  std::string str;
                  openfile >>str;
                  char x =str[0], y=str[2];
                  if (!isdigit(x) || !isdigit(y))
                  {
                        map[loadcounter.x] [loadcounter.y] = sf::Vector2i(-1,-
1);
                  }
                  else
                  {
                        map[loadcounter.x] [loadcounter.y] =
sf::Vector2i(x-'0',y-'0'); //ASCII values e.g.49-48
                  }
                  if(openfile.peek() == '\n')
                  {
                        loadcounter.x=0;
                        loadcounter.y++;
                  }
                  else
                  {
                        loadcounter.x++;
                  }
            }
      }

      sf::RenderWindow window(sf::VideoMode(320,240, 32), "Tile Map Example");
```

```
	while(window.isOpen())
	{
		sf::Event event;
		while(window.pollEvent(event))
		{
			if(event.type ==sf::Event::Closed)
				window.close();
		}
		window.clear(sf::Color(0,240,255));

		for(int i=0; i<loadcounter.x; i++)
		{
			for(int j=0; j<loadcounter.y; j++)
			{
				if(map[i][j].x !=-1 && map[i][j].y !=-1)
				{
					tiles.setPosition(i*32, j*32);
					tiles.setTextureRect(sf::IntRect(map[i][j].x *32, 
map[i][j].y *32, 32,32));
					window.draw(tiles);
				}
			}
		}

		window.display();
	}
}
```

For more information on loading tile maps using C++ and the sfml library watch the Youtube video at https://www.youtube.com/watch?v=O7lVymlZMy0

Step 3: Makefile

Using a text editor create the following makefile for the project.

```
sfml-app: tile.o
	g++ tile.o -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio -o 
sfml-app
tile.o:
	g++ -c tile.cpp
clean:
	rm -rf *.o
```

Step 4: Compiling and executing the project to observe the tile map

To compile the project use the menu command Build→Make. You should see the following output.

```
make (in directory: /home/alan/cpp-tiles)
g++ -c tile.cpp
g++ tile.o -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio -o sfml-app
Compilation finished successfully.
```

To run the application use the “execute sfml” command set up in the previous project or use the terminal command ./sfml-app.

Step 5: Changing the map.txt file to change the scene
Experiment by changing the values in the map.txt file to change the tiled map scene.

Dr Alan Crispin (31/05/15)