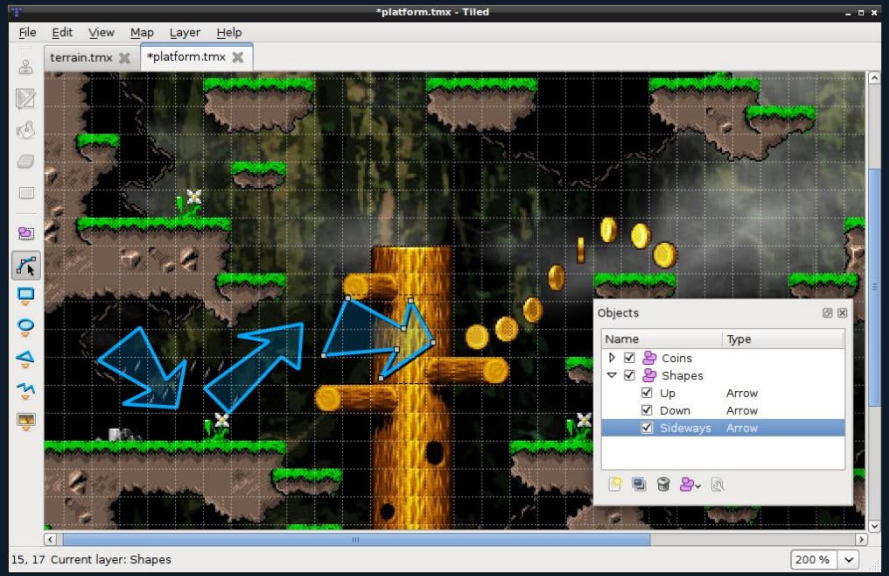
Loading a Level with JSON

Topics

- Tiled Map Editor
- Creating a tileset
- Creating a level
- Exporting a level
- The JSON format
- Drawing the level in JavaScript

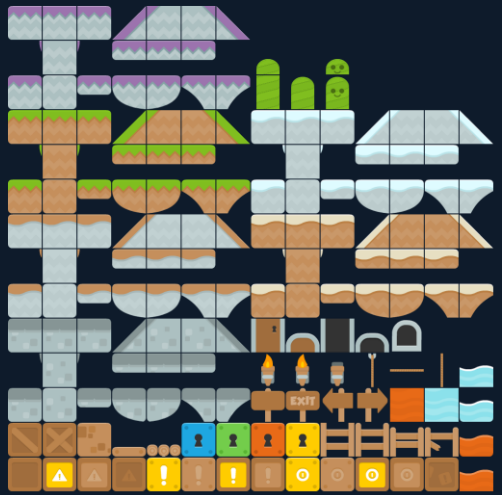
Tiled Map Editor

- Available for Windows, Mac and Linux
- http://www.mapeditor.org/



Creating a Tileset

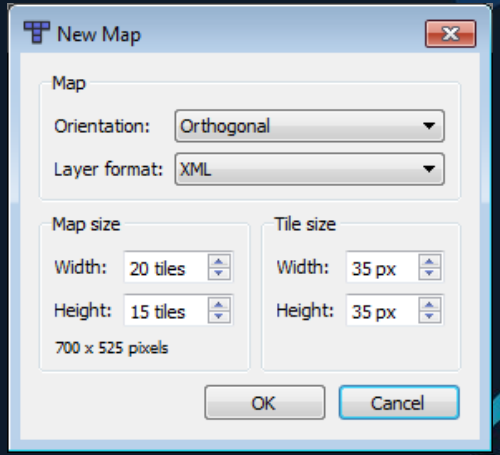
- Find or make a tileset
  - All tiles should be the same dimensions (square)
  - Tiles spaced evenly in a single PNG image



http://opengameart.org/content/kenney-platformer-base-pack-for-tiled

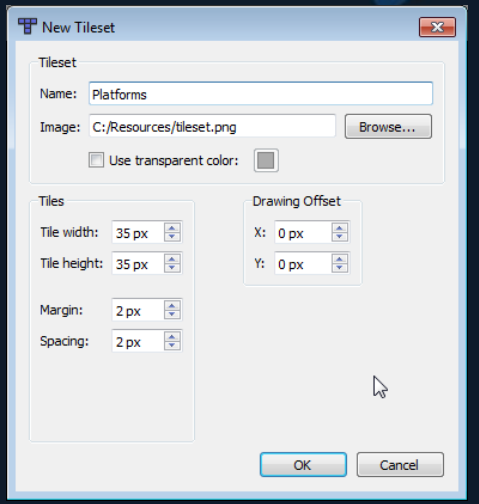
Creating a Tileset

- In Tiled, make a new map
  - Map Size: 20 tiles x 15 tiles, Tile Size 35 px x 35 px
  - (Map tile size half of tileset tile size)
- This generates an empty level
- But first, we need a tileset



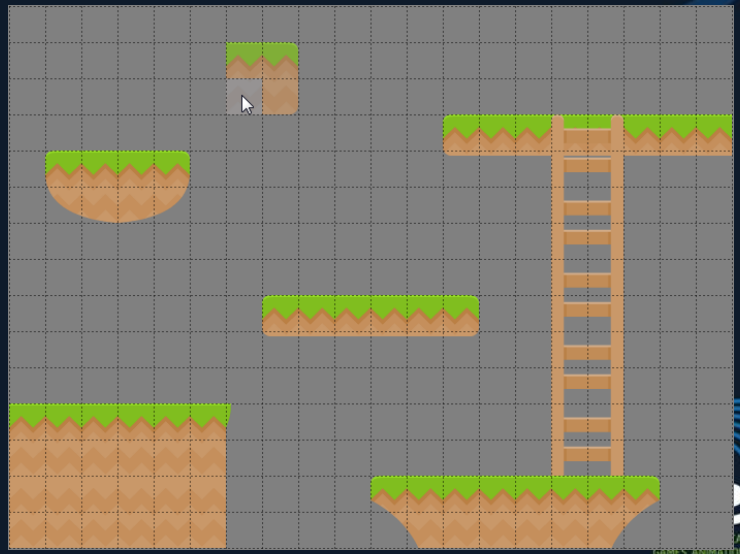
Creating a Tileset

- From the Map menu, select New Tileset
- Enter name and image location
  - Tiles 70 px x 70 px, Margin 2 px, Spacing 2 px
- The tileset should appear in the tileset window



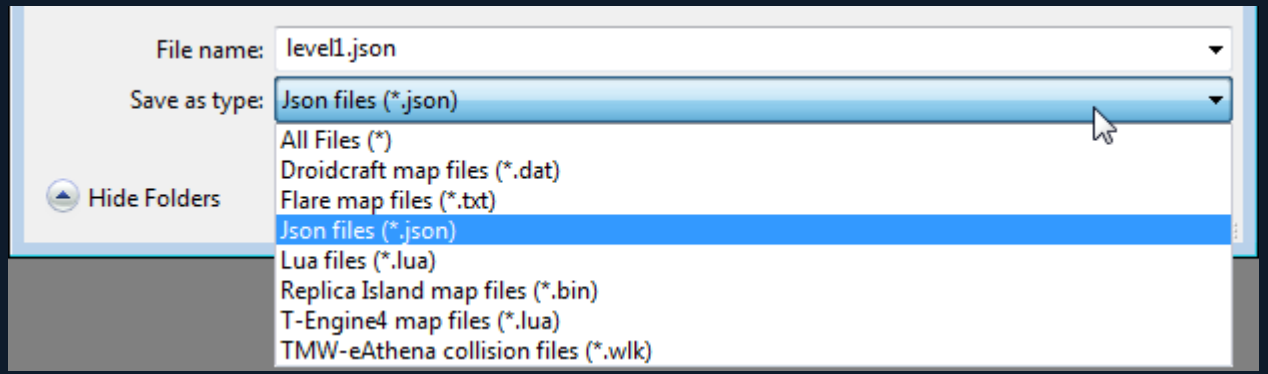
Creating a Level

- In the tileset window, select a tile
- In the level, select where to place the tile
- Add a new layer for ladders
  - Platforms on layer 1
  - Ladders on layer 2
- Save your work



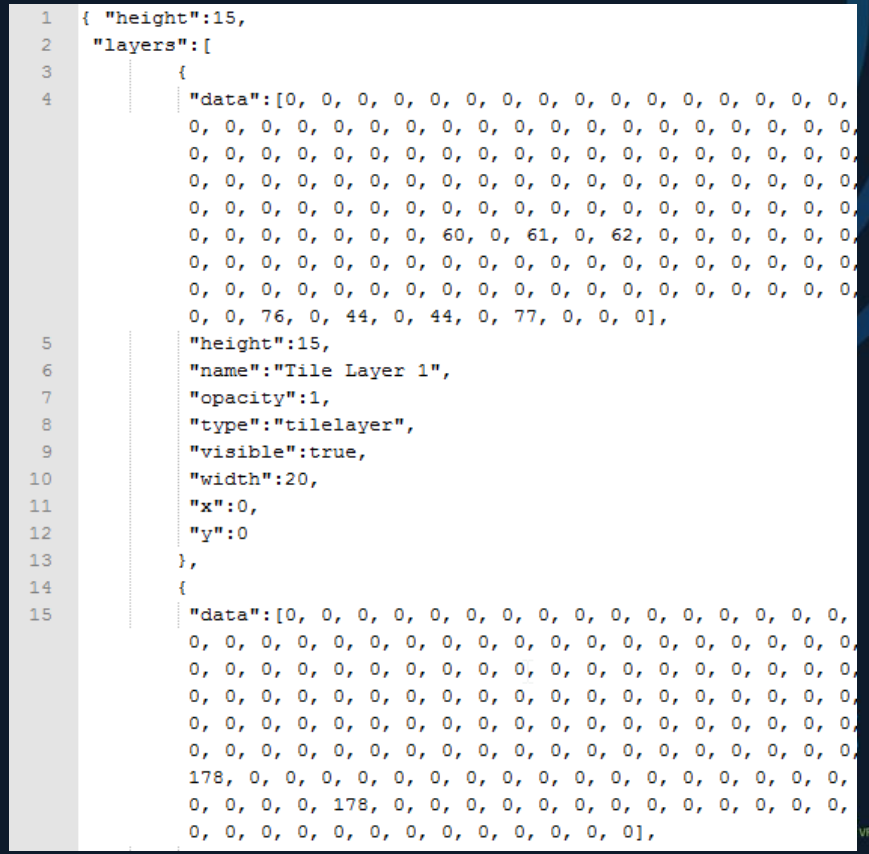
Exporting a Level

- From the File menu, select Export
- From the ‘Save as type’ dropbox, select Json files (*json)
- Save as level1.json



The JSON Format

- Open your JSON file
- It should look something like this
- Do you notice any similarities with how we defined an object in JavasScript?



```
{ "height":15,
 "layers":[
        {
         "data":[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,
          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, 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, 60, 0, 61, 0, 62, 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, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 76, 0, 44, 0, 44, 0, 77, 0, 0, 0],
         "height":15,
         "name":"Tile Layer 1",
         "opacity":1,
         "type":"tilelayer",
         "visible":true,
         "width":20,
         "x":0,
         "y":0
        },
        {
         "data":[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,
          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, 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, 0, 0, 0, 0,
          178, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 178, 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],
```

The JSON Format

- JavaScript Object Notation
- Lightweight text-data interchange format
- Language independent*
- “self-describing” and easy to understand
  - Uses JavaScript syntax for describing data objects

JSON - Evaluates to JavaScript Objects

- Syntactically identical to code for creating JS objects

```
{
   “NinjaTurtles”: [
      {      “name”: “Leonardo” ,
             “weapon” : “swords”  },
      {      “name” : “Michelangelo” ,
             “weapon” : “nunchucks”  },
      {      “name” : “Raphael” ,
             “weapon” : “sai”  },
      {      “name” : “Donatello” ,
             “weapon” : “bo staff”  }
    ]
}
```

JSON Syntax

- Subset of JavaScript
- Name/Value pairs, separated by colon

```
“firstName” : “Bubba”
```

Equals to the JavaScript statement:

```
firstName = “Bubba”
```

- Objects written inside curly brackets

```
{ “firstName” : “Bubba”, “lastName” : “Ho-tep” }
```

- Square brackets hold arrays
- The file type for JSON files is ‘.json’

Using JSON Data

- To make level1.json into a usable JavaScript Object:
  - Rename to level1.js
  - Add the text *level1 =* to the start of the file
  - Include a reference to level1.js in index.html

index.html

```
<html>
  <body>
    <canvas id="gameCanvas" width="640" height="480">
    </canvas>
    <script src="level1.js"></script>
    <script src="player.js"></script>
    <script src="main.js"></script>
  </body>
</html>
```

level1.js

```
level1 = { "height":15,
 "layers":[
        {
         "data": ...
```

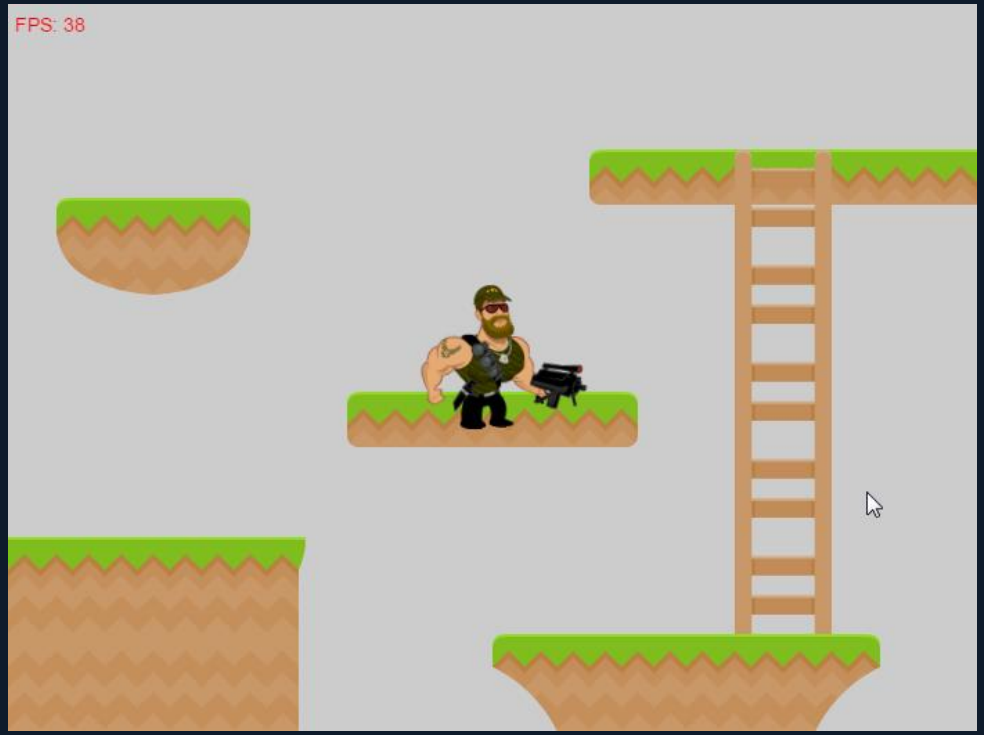
Drawing the Level in JavaScript

- Load the PNG tilemap used in Tiled
- Use the level1 object we defined in level1.js
- Loop through the level data, drawing the correct tile for the corresponding level data
- drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) draws the source rectangle (from image) to the destination rectangle (screen)

Drawing the Level in JavaScript

```
function drawMap()
{
   for(var layerIdx=0; layerIdx<LAYER_COUNT; layerIdx++)
   {
        var idx = 0;
        for( var y = 0; y < level1.layers[layerIdx].height;  y++ )
        {
            for( var x = 0; x < level1.layers[layerIdx].width;  x++ )
            {
                if( level1.layers[layerIdx].data[idx] != 0 )
                {
                    // the tiles in the Tiled map are base 1 (meaning a value of 0 means no tile), so subtract one from the tileset id to get the
                    // correct tile
                    var tileIndex = level1.layers[layerIdx].data[idx] - 1;
                    var sx = TILESET_PADDING + (tileIndex % TILESET_COUNT_X) * (TILESET_TILE + TILESET_SPACING);
                    var sy = TILESET_PADDING + (Math.floor(tileIndex / TILESET_COUNT_Y)) * (TILESET_TILE + TILESET_SPACING);
                    context.drawImage(tileset, sx, sy, TILESET_TILE, TILESET_TILE, x*TILE, (y-1)*TILE, TILESET_TILE, TILESET_TILE);
                }
                idx++;
            }
        }
    }
}
```

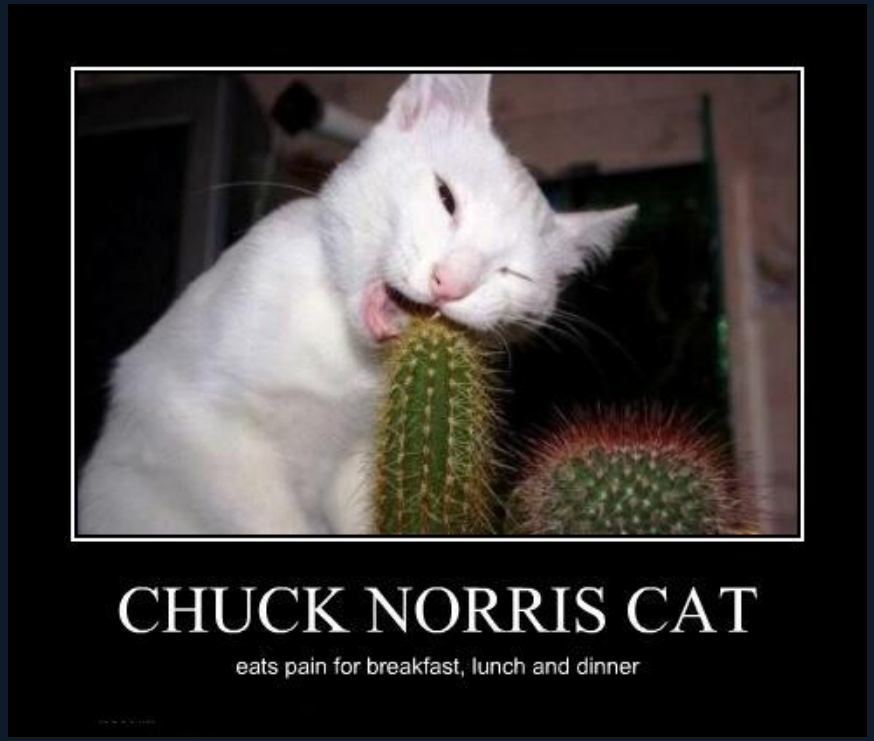
Drawing the Level in JavaScript



Summary

- Use Tiled Map Editor to make your levels
- Export to JSON
- Modify the JSON file (level1 = …)
- Access the level1 object data when drawing the map

Questions



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX