How to create tracks for BitRacer

Version 1.0

0: Contents

1: Introduction

This track creation guide is the initial version of this document. If you have problems with it please contact me. There is no dedicated editor for BitRacer, instead I have made use of Blender, an existing program, to create my tracks. Blender is not an easy program to initially use, especially with a one button mouse, but it is possible. If you have not made use of Blender before I suggest learning to use it before attempting to create a BitRacer track.

2: Track format

BitRacer uses a bundle format with the type "brtracks". The standard tracks that are used when you run BitRacer are stored in "Tracks.brtracks". Control or right click on this file to show its contents. Figure 1 shows what you see

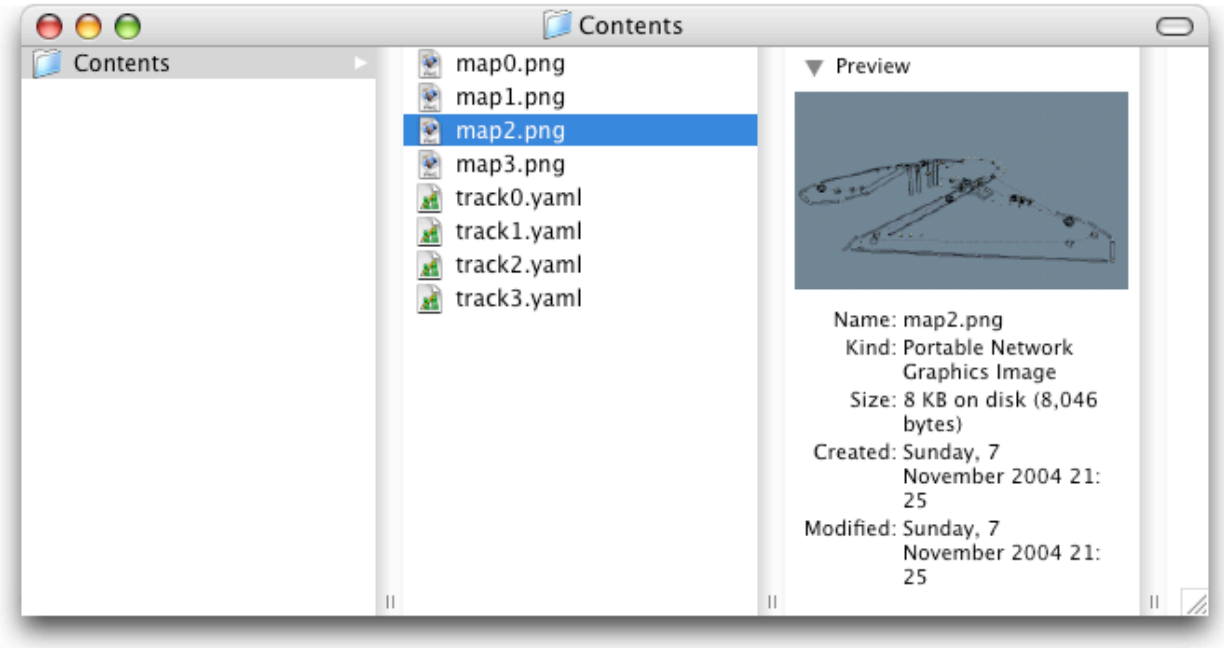

Figure 1—Tracks.brtracks content

The contents of a brtracks file consists of several map graphics and several track data files. These are named in sequence as map0, map1 ... and track0.yaml, track1.yaml ... The map files can be any file format that is normally recognised by Cocoa, png, jpg, psd etc. The data files are written in a text description language YAML (YAML Ain't Markup Language), for more information on this format see YAML.org and also will.thimbleby.net/yaml.html for source code to use in your own programs.

The YAML structure is very simple and a basic track will look something like this:

```
---
  name: induction
  laps: 5
  boxes:
    -
      posx : 142.192581177
      posy : 121.053916931
      posz : 0.224667325616
      sizex: 87.1602859497
      sizey: 4.63000011444
      sizez: 4.54579687119
      qx: 0.0
      qy: 0.0
      qz: -0.756712675095
      qw: 0.653747618198

      ......

  objects:
    -
      posx : 183.351013184
      posy : -313.958892822
      posz : 16.047712326
      type : cone

      .....

  checkpoints:
      -
        size: 2.32168984413
        posx : 27.7234096527
        posy : 11.7392082214
        posz : 0.245826676488
        previous :
          -
            posx : -332.495544434
            posy : -2.73822784424
            posz : 0.585751771927
            previous :
              .....
          -
            posx : 127.800086975
            posy : -272.891326904
            posz : 0.585752785206
      -
        posx : 324.541625977
        posy : 219.664077759
        posz : 1.77476871014
        previous :
          .....
      .....
```

The structure is the same for all tracks, and should be mostly self explanatory. Notice how indenting is used for grouping, don't use tabs. Boxes can either use quaternions as they do above for rotation, or you can specify an axis with ? ? ?. Objects are specified by their position and type (which currently can only be "cone"). Checkpoints are more complicated. They are given in order round the track as a position and size. For the computers however more info is needed and these waypoints (which you don't see when you race) are specified as a tree of previous points.

Checkpoints and how computers race

The computer AI is very simple, but can be made effective with good placement of checkpoints. An AI will choose the nearest waypoint of the checkpoint it is aiming for, and then try to drive straight at that waypoint's parent. Sound complicated? Well this method allows a track designer to specify where a computer is trying to go to at any point. The tree structure allows for recovery routes if a computer falls off a bridge or misses a turning.

3: Creating tracks in Blender

Blender is a powerful, complex, confusing and completely free 3D program. You can find it at blender.org. I used Blender to create the levels that BitRacer uses. This allows for simple placement of blocks, cones and waypoints.

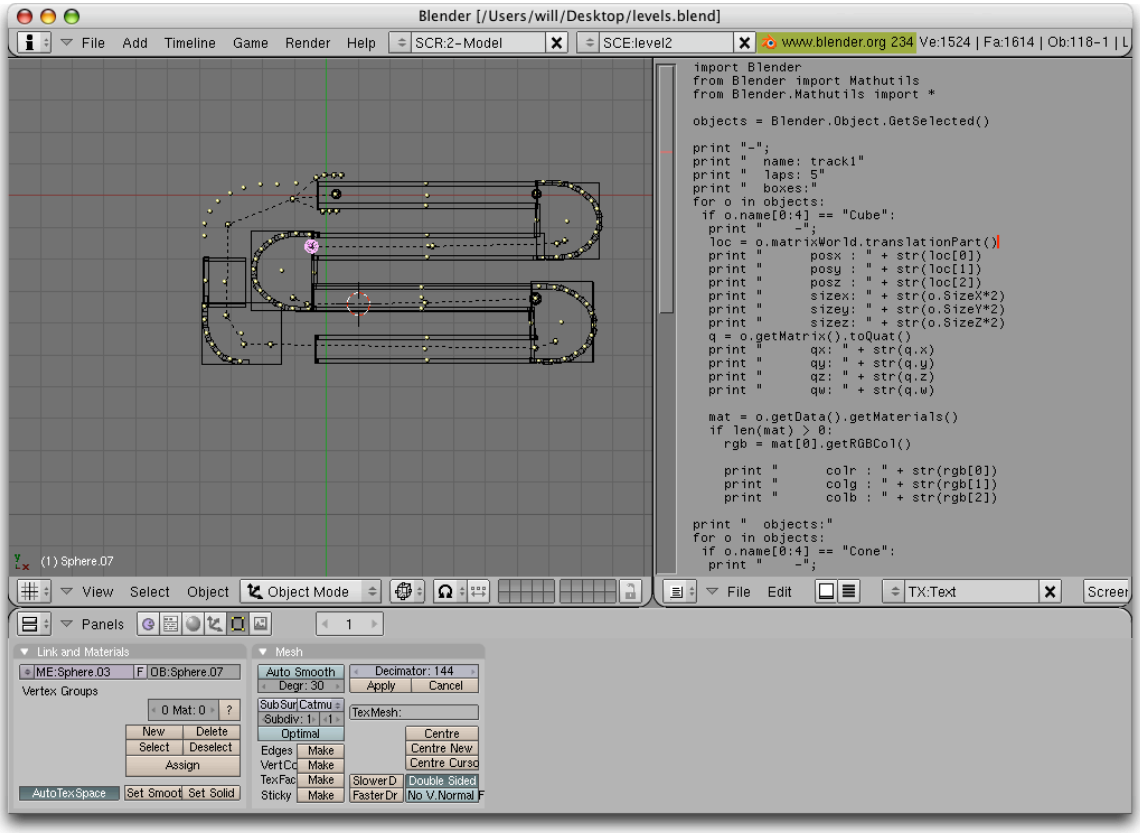

Figure 2 — Screenshot of a track being created in Blender

If you haven't used Blender before try some of the tutorials here http://www.blender3d.org/cms/Using_Blender.80.0.html. Blender allows you to use python scripts to interact with its objects and I use a script to output BitRacer levels. You can find the script and the Blender file I use to generate the standard levels for BitRacer with this file. The example Blender file contains 4 scenes each containing a track.

How to use the script
Simply run the script from within Blender with all the objects selected. The script will ask you for a file and then save that data there. Make sure you have selected all the objects you want the script to output.

How to use Blender for race tracks
Objects all use their names to identify themselves. Cubes are identified by their name, any cube you create will automatically be named Cube.###. Your cubes must be named with a name that starts "Cube", the rest is irrelevant. The same is true of cones, checkpoints and waypoints.

Racers:
The race starts at the origin, facing along the positive x-axis. Racers are about 4 units wide, which should give you an idea of scale.

Boxes:
Boxes are created as cube meshes in Blender, you can rotate, scale and position them wherever you want. As long as you do not alter the cubes in edit mode you can do what you want. Blender is automatically in edit mode when you create a cube -- make sure the first thing you do is switch to object mode (tab).

Cones:
Cones are created as cone meshes in blender, only their location is important. You can rotate and scale to make the cones visible and clear, it wont have any affect on the track.

Checkpoints:
Checkpoints are created as sphere meshes. As with cones their rotation is not considered, however their scale is. The scale determines how large the checkpoint is in the track. Try to make them big enough so that racers can't easily miss

them.

Checkpoints must be named "Sphere.#". Numbered sequentially with no gaps from some number, the lowest numbered checkpoint is the initial checkpoint.

Waypoints:
Waypoints are created as cylinder meshes. Neither their rotation or size is important. Waypoints are parented to either another waypoint or to a checkpoint. These must create a tree structure that has a checkpoint as its route node. This allows the AI to take different routes and to recover from errors. Figure 3 shows this structure for a checkpoint in the first track. The dashed lines show the parenting of the waypoints.

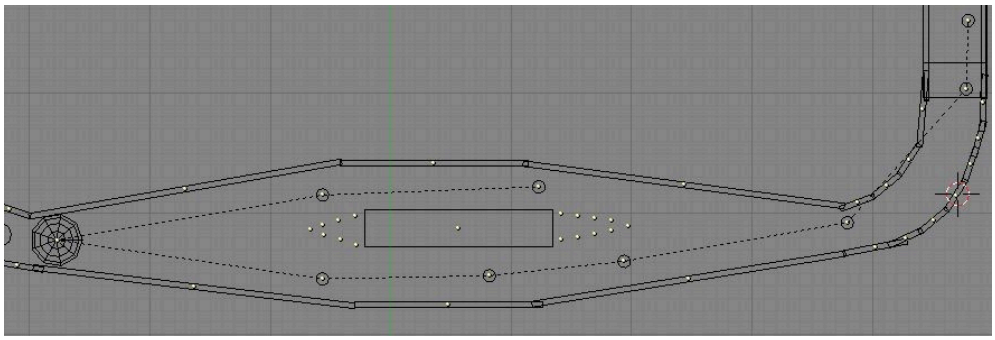

Figure 3 — Waypoint tree

4: Putting it all together

After you have created you tracks in Blender you need to use the script to output the tracks, saved as track#.yaml. Edit the files to alter the track's name and the number of laps it is. Either take a screenshot of it in Blender or create a map graphic in some other way. Place both these files in the contents of your tracks files, and you are away.

5: Contact & feedback

If you have feedback, problems or created any super cool new tracks let me know will@thimbleby.net.