DungeonBreath Manual

Making a Game

Nick Valentine

Contents

Part One

1. General

1.1 Input

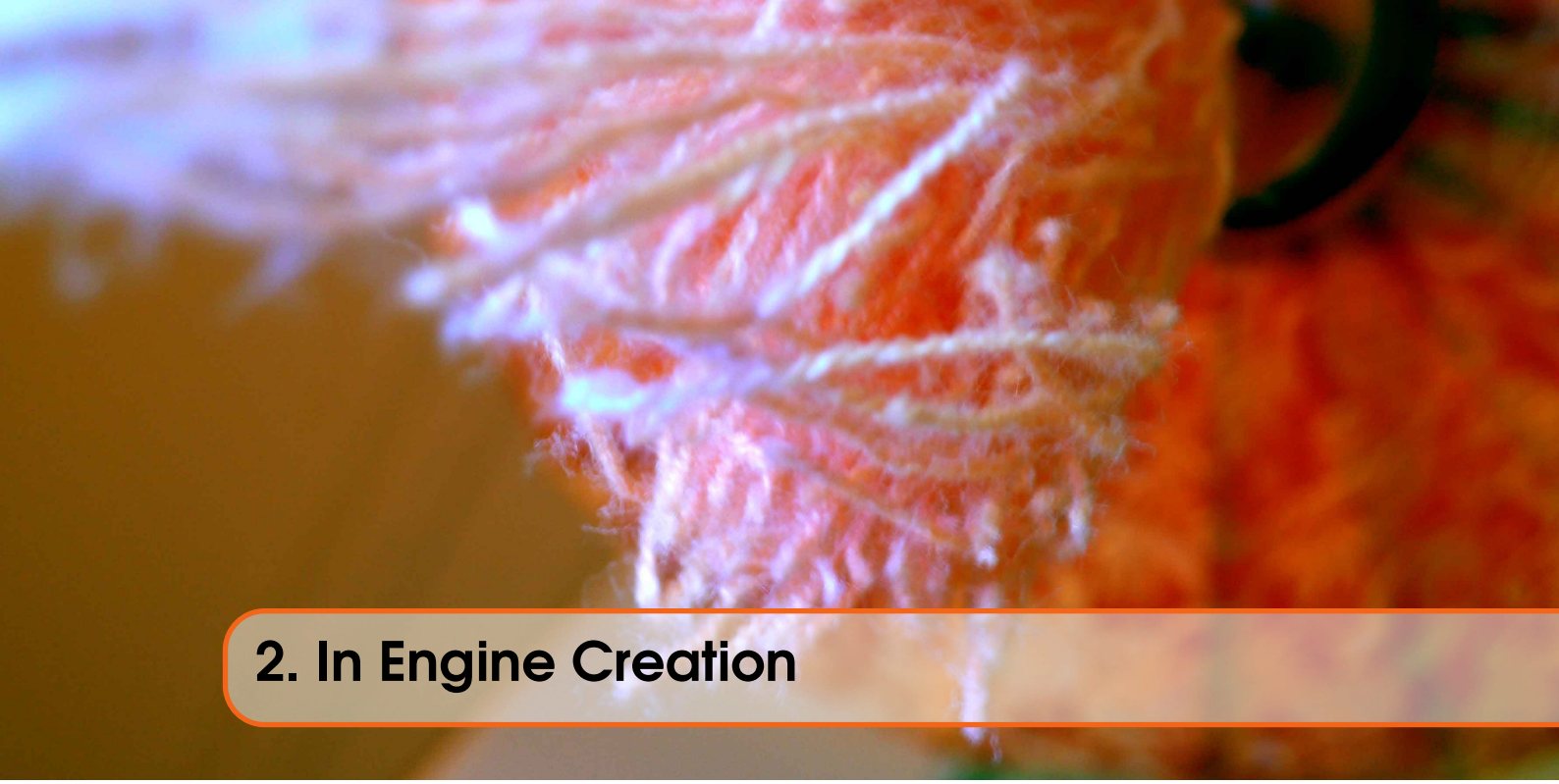
In order to interact with the game in any way, shape or form, you will need to pass input into it. These inputs are defined in

Theorem 1.1.1 — Config file. GameDir/GameData/config.txt

This file is a simple key value store where the first word on any line is the key, and the second is the value. Input definitions follow a very simple domain specific language in order to be defined.

Theorem 1.1.2 — Input DSN. device:[input number]:[axis or button]:[min value]:button

device may be either k for keyboard, or j for joystick. If the device is keyboard, then don't pass any of the optional values. The configuration would look something like k:d. However if the input device is a joystick, all options are required as the input number is which joystick to look at (1-8)[hopefully this will be refactored out soon] axis or button tells how to treat the input, and min value is useful in the case of axis to present a dead zone which will prevent false positive triggering.



2. In Engine Creation

2.1 Tile Sets

2.1.1 Image layout

The engine for DungeonBreath allows for variable size sprites, and variable size tilesets. However it will tend to work best when all images are of a size that is a power of 2, and no smaller than 16x16. Example sizes can be 32x32, 64x64, 256x256, and they don't have to be square, they can also be 32x128 if necessary. In order to import your sprites into the engine, lay them out into one single sprite sheet per use case. A hero should have their own sheet, separate from other NPCs. The most restrictive use case of separating sprites by use case is levels, each level may only have one tileset in use at a time. Once the tilesheet is layed out properly, add it to:

Theorem 2.1.1 — Image data directory. GameDir/GameData/img/

2.1.2 The tileset tool

Then you may launch the game in dev mode and press the "Tile Sets" button. Choose New and provide an easy symbolic name, the name of the spritesheet with none of the path before the img directory (images may be placed inside of sub folders). And provide as the base size, the smallest unit of measure necessary to accurately select the smallest sprite.

Once finished press next and you will be in the tileset edit screen, accept will commit the accumulated tiles up to this point as a tileset ending with the tile selected by hitting accept. If there are no previously accumulated tiles, this will become a static tile. Static tiles do not have animations and will get some engine optimizations. In order to accumulate tiles for animations, use fire while hovering over the tile you wish to select. Use this method to select all except for the very last tile, use enter on the last tile to commit as a single tileset.

2.2 Level Creation

Launch the game in dev mode and press the "Level Editor" button. You may choose new, or from a list of existing levels. Once in the level editor:

1. accept + fire will remove a tile
2. accept + alt_fire will enter actor select mode
3. alt_fire will enter tile select mode
4. fire will place the currently selected item at the cursor's position
5. escape will enter the escape menu where the user may set the layer, collision type, save, exit, or return to editing
6. up will move the cursor up one block
7. down will move the cursor down one block
8. left will move the cursor left one block
9. right will move the cursor right one block
10. next will place a collision cube at the cursor

At this time in order to choose what events are triggered by collisions, you must manually edit the world file and add to that section. The dsn is:

Theorem 2.2.1 — action dsn. <type> <action> <target>

Where type is the integer associated with the collision type, action is either teleport or call, and target is either the new world to load, or the lua function to call.