Teeter Level Encoding

Stefan Neubert

January 19 to March 26, 2014

1 Introduction

In contrast to the wooden teeter game, our implementation enables us to create interactive levels which change their layout while playing.

These levels have to be stored in such way that they provide enough possibilities for creative level design and are easy to en- and decode. Additionally, we would like them to be accessible without special software and interpretable without programming knowledge.

A typical data format would be xml, which we found to be too hard to read by humans, or an ascii-art representation, which would make it complicated to encode all needed information in a interpretable way.

We chose PNG graphics to be our source format for teeter levels. By basing the levels on an image format, it is possible to open the level file in a standard image viewer and still see the most important level structures. Because each pixel in RGBA is encoded with $4*8 = 32bit$ we have enough bytes to store the three properties a block of the Teeter-board is defined by:

Firstly, blocks may have different shapes. Generally speaking these include walls, holes and ground. But to be able to build ramps etc. the precise shape shall be defined by the height of the four corners of the block. A block with all heights being 0 is considered to be a hole and won't be put in the scene or the physics engine.

Secondly, blocks may define special appearance or actions on ball-contact.

Thirdly, blocks may have an identifier to provide inter-block interaction for those actions.

2 Encoding

Apart from the most important criteria for the encoding - to be able to represent all three properties - the encoding has to meet a second criteria which is graphical representation of the most important properties via the pixel's color, for a viewer should be able to recognize a level's basic structure by looking at the image in a standard image viewer. This basic structure is given by the shape of a block and top-level type differentiations such as standard blocks vs. finish area.

Additional properties concerning appearance or behavior customization are second-level properties, whereas third-level properties include the block's personal id or id references to blocks affected by this block's actions. Those properties do not need to be instantly visible to a viewer, but should be encoded in such way, that the decoding process (loading of level data in C++) is understandable, and the encoding process (level design in an 3^{rd} party image editor) is easy.

2.1 General structure

To be able to control the properties separately and still be able to do visual interpretation, we use the HSB[1] color model and map top-level differentiations to Hue. By encoding all additional properties with Saturation and Brightness, a viewer will still be able to tell

[1]HSB is equivalent to HSV within our usage, we use the term "Brightness" to avoid confusion between the numeric value of a property and the Value property.

from the pixel's color *what* the corresponding block does, he might though not be able to tell *how* the block behaves, for Saturation and Brightness are much harder to interpret.

By mapping the shape to Brightness, we can achieve that the most common shapes (hole, ground, wall) are distinguishable.

To ensure visual clarity and a sufficient numeric stability during conversion between the interpretable model (HSB) and the storage model of PNG (RGB), the possible values are restricted to the rules in the respective component sections.

2.2 Available Types (Hue)

As explained, types are mapped to the pixel's Hue. Because different editors use different ranges to represent Hue, we restrict possible values to common integral factors of those ranges.

Typical ranges are $\{0 \dots 359\}$ (360), $\{0 \dots 239\}$ (240) and $\{0 \dots 255\}$ (256). To ensure a floating point representation in IEE 754 of the value in the range of $\{0 \dots 1\}$ (1.0) we split the ranges in $2^3 = 8$ parts as shown in table 1.

Color example	Hue				Type	Description
	360	240	256	1.0		
🟥	0	0	0	0.000	Trap	Be victim of something extremely terrible.
🟧	45	30	32	0.125	Item	Lose or gain points, lives, become weightless, ...
🟩	90	60	64	0.250	Standard	A block without any abilities for a simple wall or ground.
🟩	135	90	96	0.375	Checkpoint	Start, Finish and Checkpoints between them.
🟦	180	120	128	0.500	Elevator	Move up or down periodically or button-triggered.
🟦	225	150	160	0.625	Ghost	Appear or disappear periodically or button-triggered.
🟪	270	180	192	0.750	Button	Trigger the action of all blocks with the same ID.
🟪	315	210	224	0.875	Portal	Fall through a wormhole to a target Portal with the same ID.

Table 1: Overview of possible Hue values for different block types.

Because the Hue cannot be converted to and from rgb without precision loss, decoded values of ± 10 are accepted.

2.3 Shape (Brightness)

Brightness is usually given in percent, most tools offer an integral selection out of $\{0 \dots 100\}$. For the four vertices of a block's height map we want to be able to set a value out of 0 (no height), 1, 2 (standard height of ground), 3 and 4 (standard height of walls). Each height will always apply to two adjacent vertices, making it possible to build ramps of all gradients in all directions.

Holes receive a Brightness of 0, for the resulting black color is easy to spot as a hole. Additionally, a Brightness of 0 per definition results in no behavior (0 Hue) and no properties (0 Saturation) - what is exactly the way holes should be. If the user sets a height of 0 for a block, it will be interpreted as hole (= no block) when decoded again.

The possible block shapes are enumerated according to table 2.

S	E	Dir	Brightness
0	0	-	00
0	1	t	57
0	1	l	58
0	1	b	59
0	1	r	60
0	2	t	61
0	2	l	62
0	2	b	63
0	2	r	64
0	3	t	65
0	3	l	66
0	3	b	67
0	3	r	68
0	4	t	69
0	4	l	70
0	4	b	71
0	4	r	72
1	1	-	73
1	2	t	74
1	2	l	75
1	2	b	76
1	2	r	77
1	3	t	78
1	3	l	79
1	3	b	80
1	3	r	81
1	4	t	82
1	4	l	83
1	4	b	84
1	4	r	85
2	2	-	86
2	3	t	87
2	3	l	88
2	3	b	89
2	3	r	90
2	4	t	91
2	4	l	92
2	4	b	93
2	4	r	94
3	3	-	95
3	4	t	96
3	4	l	97
3	4	b	98
3	4	r	99
4	4	-	100

Table 2: Overview of possible Brightness values for different block shapes.

Tests have shown, that only values above 23 will be decoded properly and will only be decoded properly if the limitations on Saturation and Hue are respected.

2.4 Type specifics (Saturation)

Values in Saturation are interpreted according to the block's type. If possible, only high Saturation values should be used, so that the type's color remains recognizable.

Saturation is usually given in percent, most tools offer an integral selection out of $\{0 \ldots 100\}$.

Due to the rgb-hsb conversion, only values ≥ 50 are permitted, all types have to accept a decoded value of ± 1. Therefore, type specifics have to be encoded using a sparse distribution of values within that range.

Durations Some types specify duration values. The values' unit are seconds, all non-user-triggered durations are synchronized across all game objects.

2.4.1 Trap (Hue 0)

Traps are subdivided into five Subtypes using the most significant digit as specified in table 3.

Color example (standard ground)	Base Saturation	Trap type
🟥	90	Mine
🟥	80	Spring
🟥	70	Trapdoor
🟥	60	*unused*
🟥	50	Black hole

Table 3: Overview of Trap blocks

Mine The ball explodes. The offset out of $\{0, 3, 6\}$ specifies the delay between the collision of ball and mine and the explosion. 0 leads to immediate explosion, 3 delays for a second and 6 for two seconds.

Spring The ball is catapulted in the air. The offset out of $\{0, 3, 6\}$ specifies the strength of the spring with higher being stronger.

Trapdoor An invisible hole opens. The offset out of $\{0, 3, 6\}$ specifies the delay between the collision of ball and trapdoor and the opening. 0 leads to immediate opening of the door, 3 delays for a second and 6 for two seconds.

Black hole The ball is attracted to the block. The offset out of $\{0, 3, 6\}$ specifies the strength and radius of the attraction with higher being stronger and farther.

2.4.2 Item (Hue 45)

Items are subdivided into five Subtypes using the most significant digit as specified in table 4.

Color example (standard ground)	Base Saturation	Item type
🟨	90	Gold bonus
🟨	80	Gold penalty
🟨	70	Life bonus
🟨	60	Life penalty
🟨	50	Anti-gravity

Table 4: Overview of Item blocks

Gold The player receives extra points. The offset out of $\{0, 3, 6\}$ specifies the amount of points awarded/lost with higher being more.

Life The player gets extra lives. The offset out of $\{0, 3, 6\}$ specifies the amount of lives granted/lost with higher being more.

Anti-gravity The ball ignores gravity for a while. The offset out of $\{0, 3, 6\}$ specifies the duration of the anti-gravity effect with higher being longer.

2.4.3 Standard (Hue 90)

A Standard block has no behavior and no properties except for it's shape.

2.4.4 Checkpoint (Hue 135)

Checkpoints are subdivided into three kinds of blocks using the most significant digit as specified in table 5.

Color example (standard ground)	Base Saturation	Type
🟩	100	Start
🟩	80	Checkpoint (mandatory)
🟩	70	Checkpoint (voluntary)
🟩	50	End

Table 5: Overview of Checkpoint blocks

Start There can only be one block of type Start on the whole level. It specifies the coordinate where the ball is spawned.

Checkpoint Mandatory Checkpoints have to be visited before the level can be completed. Voluntary Checkpoints can be visited to collect additional points. The offset out of $\{0, 3, 6\}$ specifies the amount of awarded points with higher being more.

Finish As with Checkpoints there is no limit on Finish blocks. When the ball collides with a Finish block, the player mastered the level. There might be Checkpoint blocks, which he must visit before, though.

The offset out of $\{0, 3, 6, 9, 12, 15, 18\}$ specifies the amount of awarded points for completing the level with higher being more.

2.4.5 Elevator (Hue 180)

A block of type Elevator moves up and down. It may be used to temporarily block or unblock a path, or to lift the ball on a higher plane.

The Brightness-specified shape always applies to the initial position of the elevator.

The customization describes the mode of moving up and down and the movement interval as specified in table 6.

All Elevators will react to buttons in the same group, however those which move duration-based will additionally move up and down periodically.

One of the following values of table 7 is added to the base value to specify the direction and magnitude of the movement.

Color example (standard ground)	Base Saturation	movement
🟦	75	only triggered
🟦	50	duration-based

Table 6: Overview of Elevator blocks

Offset value	0	3	6	9	12	15	18	21
Movement value	-4	-3	-2	-1	1	2	3	4

Table 7: Overview of movements for Elevators

2.4.6 Ghost (Hue 225)

A block of type Ghost appears and disappears. It may for example be used to toggle a coordinate between hole and ground.

The customization describes the initial state and the mode of (dis-)appearing (table 8).

Color example (standard ground)	Base Saturation	Initial state	state changes
🟦	100	present	only triggered
🟦	80	present	duration-based
🟦	70	hidden	only triggered
🟦	50	hidden	duration-based

Table 8: Overview of Ghost blocks

The higher the value the longer is a state's duration. Therefore the highest values can be seen as "‘infinite"’. All Ghosts will react to buttons in the same group.

The base value of initially present Ghosts is 80, the base value of initially hidden Ghosts is 50. One of the possible offsets of table 9 is added to the base value to specify the duration.

Offset value	0	3	6	9	12	15	20
Duration value	0.5	1	1.5	2	2.5	3	Infinite

Table 9: Overview of duration offsets for Ghosts

2.4.7 Button (Hue 270)

A Button triggers the effect of another Block with the same identifier.

Buttons are subdivided into three kinds of blocks using the most significant digit as specified in table 10.

Trigger Once activated, a trigger will become non-reactive.

Color example (standard ground)	Saturation	Type
🟪	100	Trigger
🟪	6X-8X	Push-Button
🟪	50	Switch

Table 10: Overview of Button blocks

Push-Button A push-button snaps back in it's initial state after a short time.

The base value of a push-button is 60. One of the following offsets of table 11 is added to the base value to specify the duration.

Offset value	0	3	6	9	12	15	18	21	24	27
Duration value	1	2	3	4	5	6	7	8	9	10

Table 11: Overview of duration offsets for push-buttons

Switch A switch toggles it state. To avert flickering, there is a short non-reactive time after a switch was toggled.

2.4.8 Portal (Hue 315)

A Portal creates a wormhole between two Blocks. The wormhole lets a ball that falls into it jump out of the connected second Portal. Note that if there are more than two Portals with the same identifier, the target portal will be randomly chosen.

Customization via Saturation specifies, on which of the block's sides the portal's entry/exit is, as specified in table 12.

Color example (standard wall)	Base Saturation	Active side (OSG axis)
🟪	90	+x (right)
🟪	80	-x (left)
🟪	70	+y (back)
🟪	60	-y (front)
🟪	50	+z (top)

Table 12: Overview of Portal blocks

Additionally, an offset value out of $\{0, 3, 6\}$ added to the base Saturation specifies how much the portal attracts balls near to its entrance. 0 deactivates attraction.

Portal blocks should preferably be height blocks (without gradient) of heights >2, because the entrance will always be about 2 high (downward from the block's maximum height on the entrance's side) and occupy the space in the given direction. Portals with top orientation may have any shape, because their entrance is always extruded in $+z$ direction, where it should not interfere with any other block.

Deviation from this recommendation might result in very unexpected behavior.

2.5 Identifiers (Alpha)

Identifiers are encoded using the Alpha channel. If sufficient, only high values should be used, to keep the color as opaque as possible. Normally the Alpha values are ranged in $\{0 \dots 255\}$ with 255 being opaque and 0 being transparent.

Identifiers are not thought to identify one block, but are designed to group related blocks, such as a Button and a Ghost or two Portals.