# A Computational Model based on Symmetry for Generating Visually Pleasing Maps of Platform Games

**Julian R. H. Mariño** and **Levi H. S. Lelis**
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, Minas Gerais, Brazil

**Abstract**

In this paper we introduce a computational model based on the concept of symmetry to generate visually pleasing maps of platform games. We cast the problem of generating symmetrical maps as an optimization task and propose a heuristic search algorithm to solve it. A user study using a platform game shows the advantage of our method over other approaches in terms of visual aesthetics and enjoyment. Another user study shows that our method is able to generate maps as visually pleasing as maps created by professional designers.

## Introduction

Automatic content generation is a major challenge in Artificial Intelligence (AI). That is, how can AI systems create stories, game maps, and sport commentaries with quality similar to those created by professional designers?

A common problem in content generation is to place a set of objects to form a game map.[1] For example, in physical-based games such as Angry Birds (Ferreira and Toledo 2014), how should one place a set of objects to form an interesting and visually pleasing level of the game? We call the problem of placing objects to form a game level the *object placement problem* (OPP). Although OPP is general and appears in application domains outside computer games (e.g., construction of graphical user interfaces), in this paper we focus on solving OPPs to create visually pleasing levels of 2D platform games such as *Super Mario Bros.* (SMB).

Our work is inspired by the works of Ngo et al. (2000; 2003) and Bauerly and Liu (2006). Ngo et al. proposed visual aesthetics metrics rooted at theories of graphical design (e.g., symmetry) for analyzing graphical user interfaces. Bauerly and Liu showed empirically that the symmetry of objects composing an image could positively correlate with people's perceived visual aesthetics – people tended to perceive as visually pleasing images that were symmetrical.

In this paper we formalize the problem of generating symmetrical solutions to OPP as an optimization problem and present a heuristic search approach to solve it optimally. That is, for a given set of input objects, our approach generates a solution to the OPP that is as symmetrical as possible according to our definition of symmetry.

[1]We use the terms level and map interchangeably in this paper.

We evaluate our algorithm with two user studies. In the first user study we employ our system to create levels of *Infinite Mario Bros* (IMB), a clone of SMB, and compare the participants' perceived visual aesthetics of the levels created by our system with those created by other approaches. Our results suggest that our system is competitive with a system that accounts for human input while creating IMB levels, and is superior to another system. The results of our second study suggests that our system is able to generate levels as visually pleasing as levels created by professional designers.

## Related Works

Approaches for automatic content creation are known as Procedural Content Generation (PCG) systems. PCG approaches to platform games include those presented by Smith et al. (2010), Sorenson et al. (2011), Snodgrass and Ontañón (2014), among others – see Togelius et al. (2011), Hendrikx et al. (2013), and Shaker et al. (2015) for reviews.

Other researchers have also built on the work of Ngo et al. (2000; 2003), and Bauerly and Liu (2006). Lai et al. (2010) evaluated Bauerly and Liu's metrics with text-overlaid images. Lai et al. did not observe in their experiment a correlation between symmetry and visual appeal, which suggests that symmetry might be an ineffective proxy to human's perceived visual aesthetics in some domains. Despite Lai et al.'s result, we show that symmetry can be used to successfully guide a search algorithm to create visually pleasing game maps. Salimun et al. (2010) tested some of Ngo et al.'s metrics in graphical interface layouts to find that people preferred symmetrical interface layouts.

In the domain of PCG for games, Liapis et al. (2012; 2014) presented evolutionary approaches to create spaceships and maps for real-time strategy games that use symmetry as a visual aesthetic metric. By contrast, we present exact algorithms to find symmetrical solutions to the OPP.

## Problem Formulation

The *level generation problem* is divided into two problems. First, one chooses a set of rectangle-shaped objects $G = \{o_1, o_2, \cdots, o_m\}$. Then, one solves $G$'s OPP. That is, one has to define the objects' $x$ and $y$ coordinates in a grid space of size $L \times k \cdot W$ ($L$ rows and $k \cdot W$ columns). We assume $G$ is provided as input, and our goal is to find visually pleasing solutions to the OPP.

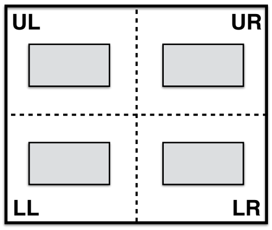


Figure 1: An example of a symmetrical placement of objects

Some objects composing a level can be placed on "top of each other" by occupying the same cells in the grid. For example, in SMB multiple mountains may occupy the same cell, while blocks must not. We assume that there is enough space in the grid to place all $m$ objects.

We transform the problem of generating a level of size $L \times k \cdot W$ into the problem of generating $k$ smaller levels of size $L \times W$. Then, the $k$ smaller levels are concatenated into a level of size $L \times k \cdot W$ with a method similar to the one introduced by Reis et al. (2015) (described below).

## Symmetry as Objective Function

In this section we describe the objective function we optimize while placing objects in a grid of size $L \times W$. The grid is divided into four regions of equal dimensions by a vertical and a horizontal line, as illustrated by the dashed lines in Figure 1. The regions are named Upper Left (UL), Upper Right (UR), Lower Left (LL), and Lower Right (LR), as shown in Figure 1. Figure 1 also shows the placement of four objects (gray rectangles), one in each of the regions. Intuitively, the object placement shown in Figure 1 results in a perfectly symmetrical image: there are identical objects on both sides of the vertical dashed line, as well as on both sides of the horizontal dashed line. We define next an objective function that captures this intuitive notion of symmetry.

We define as $G_{UL}$, $G_{UR}$, $G_{LL}$, and $G_{LR}$ the set of objects in regions $UL$, $UR$, $LL$, and $LR$, respectively. Note that an object $o$ can be placed simultaneously in more than one region. For example, half of a rectangle $o$ could fall in $LL$ and the other half in $LR$. If this happens, we replace $o$ by two objects $o_1$ and $o_2$, where $o_1$ is in $LL$ and $o_2$ in $LR$.

We define a symmetry value of a level in terms of functions $X(\cdot)$, $Y(\cdot)$, and $A(\cdot)$, where, $X(G_{LL}) = \sum_{o \in G_{LL}} dx(o)$. Here, $dx(o)$ is the distance between the center of the rectangle-shaped object $o$ and the vertical line. $Y(G_{LL})$ is the sum the values of $dy(o)$ (the distance between the center of $o$ and the horizontal line), and $A(G_{LL})$ is the sum of the values of $a(o)$ (the area of $o$), for all $o \in G_{LL}$.

### The Vertical Symmetry Function ($S_v$)

One formulation of symmetry we use is defined in terms of $X_v(G)$, $Y_v(G)$, and $A_v(G)$, where,

$$X_v(G) = |X(G_{UL}) - X(G_{UR})| + |X(G_{LL}) - X(G_{LR})| \,.$$

$Y_v(G)$ and $A_v(G)$ are defined analogously, but using functions $Y(\cdot)$ and $A(\cdot)$ instead of $X(\cdot)$, respectively. We define $S_v(G) = X_v(G) + Y_v(G) + A_v(G)$. $S_v(G)$ accounts for the symmetries across the vertical line, i.e., the differences in terms of $dx$, $dy$, and $a$-values between regions UL and UR, and between regions LL and LR.

### The All Symmetries Function ($S_a$)

Another objective function we consider is defined in terms of $X_a(G)$, $Y_a(G)$, and $A_a(G)$, where,

$$\begin{aligned} X_a(G) = {} & |X(G_{UL}) - X(G_{UR})| + |X(G_{LL}) - X(G_{LR})| \\ & + |X(G_{UL}) - X(G_{LL})| + |X(G_{UR}) - X(G_{LR})| \\ & + |X(G_{UL}) - X(G_{LR})| + |X(G_{UR}) - X(G_{LL})| \,. \end{aligned}$$

$Y_a(G)$ and $A_a(G)$ are defined analogously, but using functions $Y(\cdot)$ and $A(\cdot)$ instead of $X(\cdot)$, respectively. We define $S_a(G) = X_a(G) + Y_a(G) + A_a(G)$. In addition to the symmetries accounted by $S_v$, $S_a$ accounts for horizontal (between UL and LL, and UR and LR) and radial symmetries (between UL and LR, and UR and LL).

Images that are intuitively symmetrical, such as the one shown in Figure 1, have the desired property of having their values of $S_v$ and $S_a$ equal to zero. In this paper we test the hypothesis that by minimizing the values of $S_v$ and $S_a$ one is able to generate visually pleasing solutions to OPP in the context of map generation for 2D platform games.

## The Symmetry Problem

In the *symmetry problem* one receives as input a set of objects $G$ and integers $L$ and $W$ defining the grid's size. We assume that the height and width of the objects in $G$ are no larger than $L$ and $W$, respectively. The task is to place on the grid the objects in $G$ while minimizing $S(G)$, where $S$ could be either $S_v$ or $S_a$. In IMB, the application domain we test the algorithms we introduce in this paper, we assume that the grid has the bottom row occupied by the game's "ground", which is not accounted for in the symmetry formulas.

We define the symmetry problem as a state-space search problem and present a brute-force search algorithm to solve it. Then, we introduce a branch-and-bound (B&B) search procedure that is able to prune unpromising nodes in the search tree while still finding optimal solutions.

### Brute-Force Search (BFS)

Each level of the search tree of the symmetry problem defines the assignment of the $x$ and $y$ coordinates of one of the objects in $G$. The root of the tree is an empty partial solution to the problem, i.e., no objects have their coordinate values assigned yet. Each child of the root accounts for one possible position on the grid for a given object in $G$. Each leaf node of this search tree represents a solution to the symmetry problem. We are interested in finding an optimal solution.

We consider a set of domain-specific constraints while placing the objects on the grid. The constraints reduce the search tree branching factor by reducing the number of different positions one can place the objects. For IMB, some objects such as pipes, cannons, and mountains cannot be aloft (i.e., the object has to be placed on the ground or on top of another object). Also, every aloft object can be at most at a distance of four grid cells from another object or the level's ground to ensure that the player can interact with all objects placed on the level.

The brute-force search (BFS) algorithm traverses the search tree in a depth-first manner and returns a solution with lowest symmetry value.

## Branch and Bound (B&B) Search

Our B&B algorithm performs a depth-first search in the search tree described above and uses a *heuristic function* $H(n)$ to decided whether node $n$ can be pruned. $H(n)$ provides an estimate of the symmetry value of a complete solution obtained from partial solution $n$. $H$ is called *admissible* if it never overestimates the lowest symmetry value encountered amongst all leaf nodes in the subtree rooted at node $n$, for all $n$. Let $B$ be the symmetry value of the best incumbent solution encountered by B&B. If $H$ is admissible and $H(n) \geq B$, then node $n$ can be safely pruned as solutions reachable from $n$ are no better than $B$. B&B returns an optimal solution if employing an admissible heuristic.

## Heuristic Function

In this section we describe a heuristic function $H(n)$ for the objective function $S_a$. For node $n$, we denote as $P_n \subseteq G$ the set of objects already placed on the grid (objects that had their $x$ and $y$ coordinates assigned by the search procedure). The set of objects yet to be placed is defined as $M_n = G - P_n$. A lower bound on the symmetry value of $n$ is computed as follows.

$$H(n) = S_a(P_n) - 3 \times \left( |M_n| \times \left( \frac{W}{2} + \frac{L}{2} \right) + \sum_{o \in M_n} a(o) \right),$$

where $a(o)$ is the area of the rectangle-shaped object $o$, $\frac{W}{2}$ is an upper bound on the largest possible distance from the center of an object to the vertical line, and $\frac{L}{2}$ is an upper bound on the largest possible distance from the center of an object to the horizontal line. The term $S_a(P_n)$ is the symmetry value of the objects already placed on the grid. $H(n)$ considers that the remaining objects $M_n$ can be placed in a way that they decrease the symmetry value of the partial solution $n$ by at most the negative term of $H(n)$.

Note that $H$ could be negative, and since $S_a(G) \geq 0$, $H(n)$ is trivially set to zero if it becomes negative.

The following theorem states that $H$ is admissible.

**Theorem 1.** *Let a symmetry problem be defined by a grid of size $L \times W$ and a set of rectangle-shaped objects $G$, where the height and the width of all $o \in G$ are at most $W$ and $L$, respectively. Also, let $n$ be a partial solution to the symmetry problem where the set of objects $P_n$, with $P_n \subseteq G$, are already placed on the grid. There is no placement of $M_n = G - P_n$ that yields $S_a(G) < H(n)$.*

*Proof.* Our proof works by showing that when placing one object on the grid one is able to reduce the partial symmetry value $S_a(G)$ by at most $3 \times \left(\frac{W}{2} + \frac{L}{2} + a(o)\right)$.

A rectangle-shaped object can be placed completely inside one of the four regions ($UL$, $UR$, $LL$ or $LR$), partially in two regions, or partially in all four regions. If $o$ is placed partially in more than one region, then we treat $o$ as though it was replaced by multiple objects, one for each region it was placed on. For example, if $o$ is placed partially in two regions, then we replace $o$ by objects $o_1$ and $o_2$ with $a(o) = a(o_1) + a(o_2)$. The area $a(o)$ of an object $o$ entirely placed in one of the four regions is accounted for in three terms of $A_a$. Thus, when $o$ is placed on the grid, $a(o)$ can reduce $A_a$ by at most $3 \times a(o)$. This is true even if $o$ is placed in two or four regions, since $o$ is replaced by objects with total area of $a(o)$.

Since the grid is $W$ wide, each region is $\frac{W}{2}$ wide. Any object $o$ fully placed in one of the regions reduces $S_a$ by at most $3 \times \frac{W}{2}$ in terms of $X_a$ as the center of $o$ cannot be placed farther than $\frac{W}{2}$ from the vertical line and $dx(o)$ is accounted for three times in $X_a$. If $o$ is placed in two regions and $o$ has width $D$, then $o$ is replaced by two objects, one with width $p$ and another with width $q$ such that $D = p + q$. Since the center point of the objects have distance $\frac{p}{2}$ and $\frac{q}{2}$ from the vertical line, the two objects can reduce at most $3 \cdot \frac{p}{2} + 3 \cdot \frac{q}{2} = 3 \cdot \frac{D}{2}$ from $X_a$. Since $D \leq W$, an object placed in two regions can decrease $X_a$ by at most $3 \times \frac{W}{2}$.

If $o$ is placed in all four regions, $o$ is replaced by objects $o_1 \in UL$, $o_2 \in UR$, $o_3 \in LL$, and $o_4 \in LR$. Again let $o$ have width $D = p + q$. Since $o$ is rectangle-shaped, then $o_1$ and $o_3$ have the same width $p$ and $o_2$ and $o_4$ have the same width $q$. Note that $dx(o_1)$ cancels out with $dx(o_3)$ and $dx(o_2)$ cancels out with $dx(o_4)$ in $X_a$. Thus, $X_a$ can be written considering the four non-cancelled terms as $4 \cdot \frac{(|p-q|)}{2} = 2 \cdot |p-q|$, which is maximized when $p = \frac{W}{2}$ and $q = 0$ yielding $W$. Thus, $3 \times \frac{W}{2}$ is an upper bound on how much $X_a$ can be reduced by placing $o$ in all four regions.

The proof for the term $3 \times \frac{L}{2}$ is analogous to the proof just shown for $3 \times \frac{W}{2}$ when placing an object $o$. $\square$

## Object and Region Orderings

Since B&B prunes all nodes $n$ for which $H(n) \geq B$, it will tend to prune more nodes if $H$ provides values that are closer to perfect (i.e., larger values if $H$ is admissible). We now describe how the order in which the objects are placed on the grid might affect the accuracy of the $H$-values.

The negative term of $H(n)$ accounts for the area of the objects yet to be placed, $M_n$. Thus, if B&B places the objects with larger $a$-values early in search, it reduces the negative term of $H$ thus increasing the $H$-value of nodes at deeper levels of the tree. Aiming at obtaining more accurate heuristic estimates, in our implementation of B&B we sort the objects in $G$ according to their areas and place larger objects before smaller ones.

Another factor that affects how much pruning B&B is able to perform is the cost of the incumbent solution $B$. Intuitively, B&B is able to prune larger portions of the search tree if it quickly finds an optimal or near-optimal solution to the symmetry problem. This is because smaller $B$-values allow more nodes $n$ to be pruned by satisfying the $H(n) \geq B$ condition. In addition to defining an ordering of objects to improve the estimates returned by $H$, we define an ordering of regions to try to reduce the B&B search tree size.

Once the search procedure defines which object $o$ is going to be placed next during search, B&B verifies which region $R \in \{UL, UR, LL, LR\}$ has the smallest sum $X(G_R) + Y(G_R) + A(G_R)$. B&B tries to place $o$ in $R$ before any other region. The intuition is that B&B tries to keep all regions with similar $X$, $Y$, and $A$-values, so that the values cancel out and B&B finds good solutions early in search.

## Node Expansion and Time Experiment

We ran experiments with BFS and B&B using $H$ and the object and region ordering strategies described above with sets $G$ of sizes 4, 5, 6, 7, 8 and 9. We used 20 different sets $G$ of each size and grids of size $15 \times 15$. The objects used in our experiment had their areas chosen randomly from 1 grid cell to 225 grid cells. We compared B&B's and BFS's number of nodes expanded and running time. The detailed results of this experiment are omitted for space; in this section we highlight our main findings.

Our experiment showed that B&B expands on average from 2.54 ($|G| = 9$) to 13.87 ($|G| = 4$) times fewer nodes than BFS. In terms of running time, B&B is 5.53 times faster than BFS for $|G| = 4$ and 1.83 times faster than BFS for $|G| = 9$. We observed a larger difference between B&B and BFS in terms of nodes expanded than in terms of running time because B&B's time-per-node is more expensive than BFS's due to B&B's heuristic computation. As an example of running time, for $|G| = 5$ B&B takes 3.13 seconds on average to find an optimal solution, while BFS takes 12.14 seconds on average for sets of the same size.

The difference in running time of B&B and BFS is important because we envision our system being used to generate content in real time as the player goes through the level. This way our system could account for player's preferences in addition visual aesthetics metrics.

## Representative Small Levels

In this section we show a few representative small levels generated by our approach while minimizing $S_v$ and $S_a$. In this experiment we use four different sets $G$. The results are shown in Figures 2 and 3, for $S_v$ and $S_a$, respectively. Enemies and coins were added by following simple rules: enemies were placed randomly and coins were placed on top of some of the objects composing the level.

Often the search procedure outputs exactly the same small level for both $S_v$ and $S_a$ for a given set $G$; see the images on the left-hand side of Figures 2 and 3 for a representative case. We also observed that the search procedure tends to place more objects in the upper part of the level when minimizing $S_a$ than when minimizing $S_v$; see the images on the right-hand side of Figures 2 and 3 for a representative case. When minimizing $S_a$ the search procedure tries to balance the objects that must be placed in the lower part of the level (e.g., mountains have to be placed on the ground) by placing objects in the upper part of the level.

## Generating Complete Game Maps

B&B can be used to generate small symmetrical game levels. Next, we explain how one can generate complete game maps out of small levels by using an approach similar to the one introduced by Reis at al. (2015). We name the PCG system described in this section as `Symmetry`.

We generate a complete level of a platform game by concatenating small levels together according to a mathematical function called *tension arc*. A tension arc receives as input the position $j$ of a small level within the complete level and returns the difficulty value of the small level occupying the $j$-th position. For example, for $j = 1$ (i.e., the first small level composing the complete level) the tension arc might return a difficulty value indicating that the first small level composing the full-sized level must be an easy one.

Reis et al. use human computation to label a set of small levels according to people's perceived visual aesthetics, enjoyment, and difficulty. Then, they use a tension arc and the labels provided by humans to construct complete levels of a platform game. Instead of using human computation to select small levels with good visual aesthetics for a given value of difficulty, we use our B&B procedure to generate a set of small symmetrical levels $\mathbf{L}$, and the metric of *leniency* (Smith and Whitehead 2010) to replace the human-rated difficulty values. The leniency values of levels in $\mathbf{L}$ are normalized to values between 1 and 7, where levels with leniency value of 1 are expected to be much more difficult than levels with leniency value of 7. We use the tension arc defined by the following ordered set: $\{7, 6, 5, 4, 4, 3, 3, 4, 5\}$, which states, for example, that the first small level composing the complete level has a leniency value of 7 and the last has a leniency value of 5.

## Comparison with Other Systems

We use three different IMB PCG systems in our experiment: `Symmetry`, Human-Computation Tension Arc-Based (HCTA) with a parabolic tension arc, as described by Reis et al. (2015), and Occupancy-Regulated Extension (ORE) generator (Mawhorter and Mateas 2010).

We chose to use HCTA and ORE in our experiment because the former was shown to outperform other approaches (Reis, Lelis, and Gal 2015) and the latter was the winner of the 2011 Mario AI Competition.[2]

In addition to a Turing test, the systems are evaluated according to the following criteria: enjoyment, visual aesthetics, and difficulty. Each participant was asked whether they agreed, in a 7-Likert scale, with the following sentences: "This level is enjoyable to play"; "this level has good visual aesthetics"; "this level is difficult"; "this level was developed by a machine". A score of 1 for enjoyment and visual aesthetics means that the participant strongly agrees that the level played is enjoyable and has good visual aesthetics; a score of 1 for difficulty means that the participant strongly agrees that the level is difficult; a score of 1 for the Turing criterion means that the participant strongly agrees that the level was designed by a machine.

Subjects were instructed about the controls of the game and played one practice level before playing one level generated by each system. After playing each level, the subjects

[2]see http://www.marioai.org/LevelGeneration.

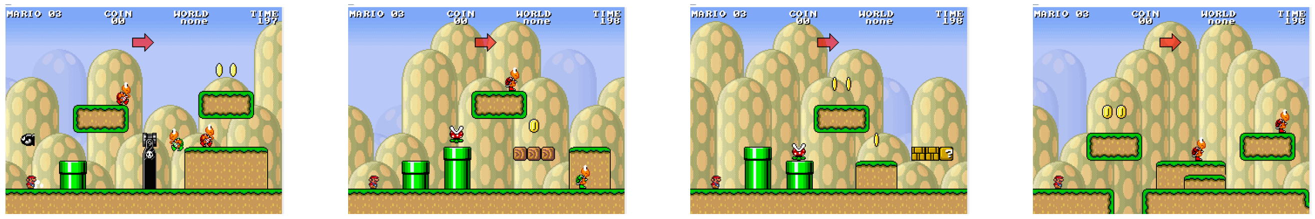


Figure 2: Representative small levels generated while minimizing $S_v$.

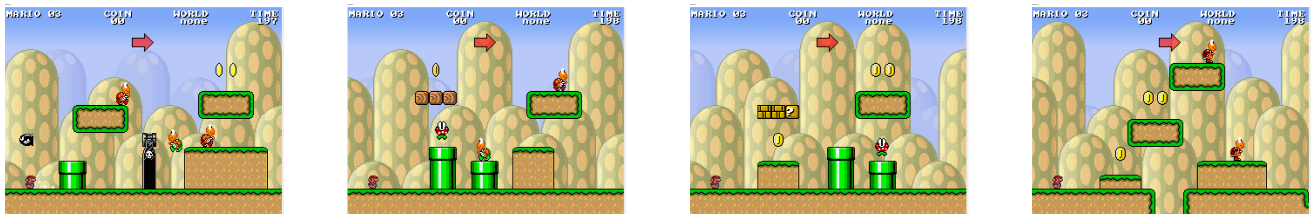


Figure 3: Representative small levels generated while minimizing $S_a$.

| | SYM | HCTA | ORE |
|---|---|---|---|
| enjoyment | $2.70 \pm 1.69^a$ | $2.97 \pm 1.68^a$ | $3.89 \pm 2.01^b$ |
| aesthetics | $2.72 \pm 1.69^a$ | $2.78 \pm 1.72^a$ | $3.36 \pm 1.79^b$ |
| difficulty | $3.74 \pm 1.85^a$ | $4.06 \pm 1.78^a$ | $3.29 \pm 2.12^a$ |
| Turing | $3.06 \pm 1.83^a$ | $3.95 \pm 2.12^b$ | $2.31 \pm 1.93^c$ |

Table 1: Lower values are better for enjoyment and (visual) aesthetics; larger values are better for Turing. Different letters in a given row indicate statistically significant results.

gave scores according to the criteria described above. In addition to the scores, the subjects could inform us of technical issues they might have had during the experiment. At the end of the experiment the subjects filled a questionnaire informing their age, gender, and if they had played SMB before.

Our within-subject experiment considered only the data of participants who finished playing all levels. Considering those, the experiment had 47 participants: 43 males and 4 females with an average age of 27.53 and standard deviation of 5.59. All participants had played SMB before. The experiment was carried out online: our system was made available in the Internet and our experiment advertised in different mailing lists. Participation was anonymous.

Since all participants played one level generated by each system, we used a balanced Latin square design to counteract possible ordering effects. The tested levels were generated during the experiment by the evaluated systems – we did not pre-select a set of levels to be tested. All systems generated levels of $160 \times 15$. The systems were manually tuned to generate levels with similar difficulty. Our experimental design follows the one suggested by Mariño et al. (2015).

The mean results and standard deviations are shown in Table 1. Different letters in a given row indicate statistical significance. Shapiro-Wilk tests showed that our data is unlikely to be normally distributed ($p < .01$). Thus, we used the non-parametric Friedman test which showed significant differences in all criteria but difficulty ($p < .01$).

We used Wilcoxon signed-rank tests to perform pairwise comparisons of the results. We present the effect size of the comparisons ($r$-values) in addition to $p$-values. `Symmetry` generates levels that are significantly more enjoyable to play than those generated by ORE ($p < .001$, $r = 0.50$). HCTA also generates levels that are significantly more enjoyable to play than those generated by ORE ($p < .01$, $r = 0.42$). There is no statistical difference between the levels generated by `Symmetry` and HCTA with respect to enjoyment.

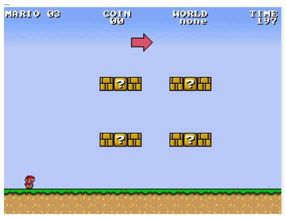
Pairwise comparisons on visual aesthetics showed that `Symmetry` and HCTA generate levels with significantly better visual aesthetics than those generated by ORE ($p < .05$, $r = 0.37$ for `Symmetry` and $p < .01$, $r = 0.39$ for HCTA). There is no statistical significance between `Symmetry` and HCTA in terms of visual aesthetics.

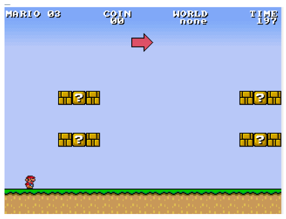
Pairwise comparisons on Turing showed that the levels HCTA generates trick more people into thinking they were generated by human designers than the levels generated by `Symmetry` ($p < .05$, $r = 0.30$) and by ORE ($p < .0001$, $r = 0.56$). Also, the levels `Symmetry` generates trick more people into thinking they were generated by human designers than the levels generated by ORE ($p < .05$, $r = 0.31$).

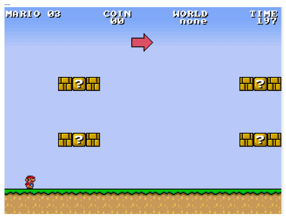
All pairwise comparisons reported as significant have effect sizes around the medium size mark of 0.3, indicating substantial differences among the systems. Some of the pairwise comparisons show large effect sizes ($r$-values $> 0.50$).

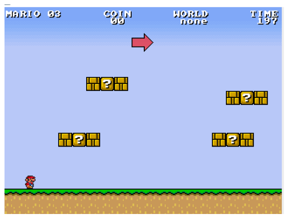
Our results suggest that `Symmetry` generates levels that are as enjoyable and as visually pleasing as those generated by HCTA. HCTA is a method that uses human workers to select the set of small levels composing the IMB level. `Symmetry` does not use any sort of human input: it solely relies on the quality of the symmetrical levels generated by our B&B approach.

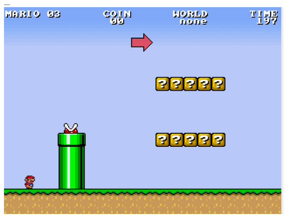
The only criterion that `Symmetry` and HCTA differ is Turing. This shows that although the levels generated by both systems have comparable visual appeal, the participants were able to notice that the levels generated by `Symmetry` were created by a computer program.

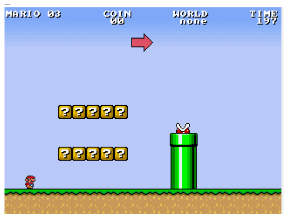

(a) Original.

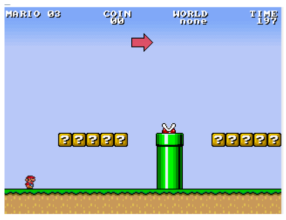

(b) $S_a$

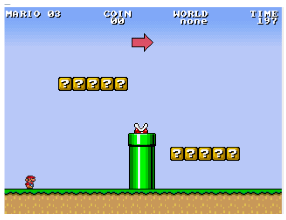

(c) $S_v$

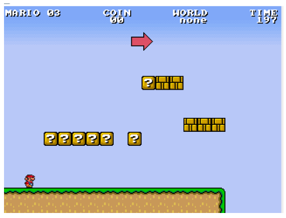

(d) Random.

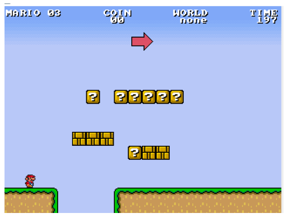

(e) Original.

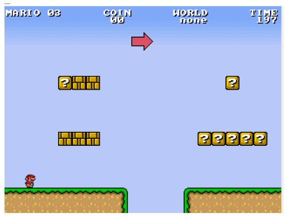

(f) $S_a$


(g) $S_v$


(h) Random.


(i) Original.


(j) $S_a$


(k) $S_v$

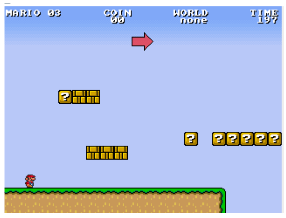

(l) Random.

Figure 4: Representative sets of objects $G$ used in our user study.

Finally, `Symmetry` was able to generate levels that are significantly and substantially more enjoyable, and more visually pleasing than those generated by ORE, a system that also does not account for human input while creating levels.

## Comparison with Human Designers

In this section we compare the visual aesthetics of levels generated by our system with those created by professional designers for fixed sets of objects $G$.

We reproduced in IMB 10 chunks of original SMB levels of size $15 \times 15$. These small levels were arbitrarily chosen from an online repository of game maps.[3] We collected the set of objects $G$ in each small SMB level and provided it as input to B&B. B&B then produced levels, also of size $15 \times 15$, with the same set of objects as the original ones.

We ran a user study in which we presented images of the original small levels side by side with those generated by our B&B procedure while minimizing $S_a$ and $S_v$. In addition to the small levels created by professional designers and those generated by our system, we also displayed images of levels created by randomly placing the objects but accounting for the domain-specific constraints used with B&B.

In contrast with the IMB game, the images of the levels were presented to the participants with a blue background to ensure that the random background generated by the game would not bias the participants' perceived visual aesthetics. Figure 4 shows three representative sets of objects $G$.

[3]http://ian-albert.com/games

| $S_a$ | $S_v$ | Original | Random |
|---|---|---|---|
| $3.26 \pm 1.74^a$ | $3.08 \pm 1.91^a$ | $3.08 \pm 1.85^a$ | $5.04 \pm 1.96^b$ |

Table 2: Lower values are better. Different letters in a given row indicate statistically significant results.

Each participant visualized all four images at once: one created by a professional designer (Original), two generated by our system ($S_a$ and $S_v$), and one generated by the random approach (Random). The participants answered for each image if they agreed with the statement "This level has good visual aesthetics" in a 7-Likert scale. After evaluating all 4 images, the participant could choose to either quit the experiment or to evaluate another set of images. Each participant could evaluate from 1 to 10 sets $G$. In order to account for ordering effects, we randomized the order in which the sets $G$ and the images were presented. Our experiment was performed online. At the end of the experiment the subjects filled a questionnaire informing their age, gender, and if they had played SMB before. Our within-subject experiment had 23 participants: 19 males and 4 females with an average age of 26.38 and standard deviation 3.81. The 23 participants evaluated 215 sets $G$. They all had played SMB before.

Table 2 shows the average score and standard deviation of each approach. Lower values mean that the participants agreed more strongly that the levels have good visual aesthetics. Different letters indicate that two means are significantly different. A Shapiro-Wilk test showed that our data is unlikely to be normally distributed ($p < .0001$). The non-parametric Friedman test showed a significant difference on the means ($p < .0001$). Pairwise comparisons with Wilcoxon signed-rank tests showed statistical difference between Random and all the other approaches ($p < .0001$ and $r > 0.50$ for all comparisons). There was no statistical difference between the levels generated by our system and those created by professional designers.

Our system often generated levels that were similar to those created by professionals (see Figure 4 for examples). The user study results suggest that our system using either $S_a$ or $S_v$ is able to generate small levels as visually pleasing as those created by professionals, and that have far better visual aesthetics than those generated by Random.

These results support our hypothesis that our system is able to generate visually pleasing game maps.

## Conclusions

In this paper we introduced a computational model based on the concept of symmetry for generating visually pleasing game maps. We cast the problem of generating symmetrical maps as an optimization task and propose a B&B approach to solve it. We tested our approach with two user studies. In the first user study the participants evaluated the levels generated by our system as enjoyable and as visually pleasing as the levels created by a system that requires human input, and more enjoyable and more visually pleasing than the levels generated by another system. In the second user study we compared the visual aesthetics of the small levels generated by our system with those created by professional designers. The participants rated the levels generated by our approach as visually pleasing as those created by the professionals.

Although we applied our method to game design, we believe that our computational model and search procedure are general and applicable to other domains such as the placement of buttons and icons in graphical user interfaces.

## Acknowledgements

The authors gratefully thank Claudio Toledo, Leonardo Pereira, Lucas Ferreira, and Sandra Zilles for their constructive comments on an earlier version of this paper. Financial support for this research was provided by Brazil's CAPES (Science Without Borders), FAPEMIG, and CNPq.

## References

Bauerly, M., and Liu, Y. 2006. Computational modeling and experimental investigation of effects of compositional elements on interface and design aesthetics. *International Journal of Human-Computer Studies* 64(8):670–682.

Ferreira, L., and Toledo, C. F. M. 2014. A search-based approach for generating angry birds levels. In *Proceedings of the Conference on Computational Intelligence and Games*.

Hendrikx, M.; Meijer, S.; Van Der Velden, J.; and Iosup, A. 2013. Procedural content generation for games: A survey. *ACM Transactions Multimedia Computing, Communications. Applications* 9(1):1:1–1:22.

Lai, C.-Y.; Chen, P.-H.; Shih, S.-W.; Liu, Y.; and Hong, J.-S. 2010. Computational models and experimental investigations of effects of balance and symmetry on the aesthetics of text-overlaid images. *International Journal of Human-Computer Studies* 68(1):41–56.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2012. Adapting models of visual aesthetics for personalized content creation. *Transactions on Computational Intelligence and AI in Games* 4(3):213–228.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Designer modeling for sentient sketchbook. In *Proceedings of the Conference on Computational Intelligence and Games*.

Mariño, J. R. H.; Reis, W. M. P.; and Lelis, L. H. S. 2015. An empirical evaluation of evaluation metrics of procedurally generated Mario levels. In *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment*, 44–50.

Mawhorter, P. A., and Mateas, M. 2010. Procedural level generation using occupancy-regulated extension. In *Proceedings of the Conference of Computational Intelligence and Games*, 351–358.

Ngo, D. C. L.; Samsudin, A.; and Abdullah, R. 2000. Aesthetic measures for assessing graphic screens. *Journal of Information Science and Engineering* 16(1):97–116.

Ngo, D. C. L.; Teo, L. S.; and Byrne, J. G. 2003. Modelling interface aesthetics. *Information Sciences* 152:25–46.

Reis, W. M. P.; Lelis, L. H. S.; and Gal, Y. 2015. Human computation for procedural content generation in platform games. In *Proceedings of the Conference of Computational Intelligence and Games*, 99–106.

Salimun, C.; Purchase, H. C.; Simmons, D. R.; and Brewster, S. 2010. Preference ranking of screen layout principles. In *Proceedings of the BCS Interaction Specialist Group Conference*, 81–87. British Computer Society.

Shaker, N.; Togelius, J.; and Nelson, M. J. 2015. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer.

Smith, G., and Whitehead, J. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the Workshop on Procedural Content Generation in Games*, 1–7. ACM.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: a mixed-initiative level design tool. In *Proceedings of the International Conference on the Foundations of Digital Games*, 209–216. ACM.

Snodgrass, S., and Ontañón, S. 2014. A hierarchical approach to generating maps using Markov chains. In *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment*, 59–65.

Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computing Intelligence and AI in Games* 3(3):229–244.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.