EUROGRAPHICS 2022/ N.N. Short Paper

# Scene Synthesis with Automated Generation of Textual Descriptions (Supplementary Material)

J. Müller-Huschke, M. Ritter, and M. Harders

Interactive Graphics and Simulation Group at the Department of Computer Science, University of Innsbruck, Austria.

\*corresponding author

#### 1. Scene evaluation function

To build the evaluation function from the set of constraints the approach of [YYW\*12] is utilized. In their work they build a function that scores a scene layout between 0 and 1 by taking the product of many factors. Each factor represents a logical connective. So, if constraints on the scene layout can be formulated with logical connectives, this method allows to easily build an evaluation function from constraints. [YYW\*12] define these logical connective factor functions based on the Gaussian density function  $\mathcal{N}(x,\mu,\sigma^2)$  and a sigmoid function  $\mathrm{Sig}(x,h) = \frac{1}{1+e^{-hx}}$ :

Equals
$$(x, y, \sigma^2) = \mathcal{N}(0, ||x - y||, \sigma^2) / \mathcal{N}(0, 0, \sigma^2)$$
  
Greater $(x, y, h) = \text{Sig}(x - y, h)$   
Less $(x, y, h) = \text{Sig}(y - x, h)$ 
(1)

 $\operatorname{Range}(x, y_{\min}, y_{\max}, h) = \operatorname{Greater}(x, y_{\min}, h) \operatorname{Less}(x, y_{\max}, h).$ 

The constants h and  $\sigma^2$  define their steepness. In this work they are chosen the same as in [YYW\*12], i.e. h = 3.0 and  $\sigma^2 = 0.1$ . These logical constraints can be used to define spatial constraints. For example, if a scene contains two objects A and B, which should be at a perpendicular angle to each other and two meters apart from each other, the scene evaluation function could look like this:

$$f = \text{Equals}(angleBetween(A, B), 90.0, \sigma^{2}) \cdot$$

$$\text{Greater}(distanceBetween(A, B), 2.0, h).$$
(2)

The evaluation function takes scene layout as input and produces an evaluation score between 0 and 1, with values close to 1 representing the condition that all constraints are perfectly satisfied. Ideally, no scene layout should ever be scored with 0, since then calculating the acceptance ratio for Metropolis Hastings becomes undefined. Additionally, it is likely that neighboring scene layouts in the search space will also be scored with 0, removing any sort of gradient that could allow the Markov chain to quickly converge. The evaluation function f is defined and computed in log space using an energy function v. This is in order to avoid underflows, e.g. the evaluation function becoming 0 due to floating point precision limitations when taking the product of many very small numbers. In log space these products become sums mitigating the issue to some extent.

$$f = e^{-\nu}. (3)$$

The energy function v consists of four parts concerning different types of constraints:

$$v = v_r + v_c + v_p + v_q. \tag{4}$$

- $v_r$  results from spatial relationships between objects.
- $v_c$  states objects should not collide with one another.
- $v_p$  states objects should not collide with pathways.
- $v_q$  encourages generation of believable pathways.

For each edge of the scene relationship graph, an evaluation factor is computed and summed up to form  $v_r$ . This factor is built from one or multiple factor functions from [YYW\*12] (see equation 1). However, these factor functions are usually instantiated in log space, which is denoted here by writing their names in lower case. Table 1 shows how to compute the evaluation factors for all possible types of edges in the scene relationship graph. To keep this table compact, an additional factor function enforces that two unit vectors  $\vec{v}$ ,  $\vec{w}$  point in the same direction:

**pointingSameDirection**
$$(\vec{v}, \vec{w}) = \text{equals}(\cos^{-1}(\vec{v} \cdot \vec{w}), 0).$$
 (5)

A number of functions are used in table 1 in order to get information about the relative transformation between scene objects and their collision geometries:

- distanceBetween(A,B) gives the minimum distance that can be found between the collision geometries of objects A and B.
- *directionFromTo*(*A*, *B*) gives the normalized direction vector pointing from the position of *A* to the position of *B*.
- $direction(A, \vec{x})$  gives the direction vector  $\vec{x}$  transformed to the local coordinate system of object A.
- boundary $(A, \vec{x})$  gives the most extreme point in direction  $\vec{x}$  on the collision geometry of A. The resulting point is given in the global coordinate system.

Edge Label from A to B	Resulting factor
NextTo	less(distanceBetween(A,B),2)
NextToChooseDirection	
NextToChooseSide	less(distanceBetween(A, B), 2) + max(
	pointingSameDirection(directionFromTo(B, A), direction(B, left)),
	pointingSameDirection(directionFromTo(B, A), direction(B, right)))
NextToNorth	less(distanceBetween(A, B), 2) +
	pointingSameDirection(directionFromTo(B,A), north)
NextToEast	less(distanceBetween(A, B), 2) +
	pointingSameDirection(directionFromTo(B, A), east)
NextToSouth	less(distanceBetween(A,B),2) +
	pointingSameDirection(directionFromTo(B, A), south)
NextToWest	less(distanceBetween(A, B), 2) +
	<b>pointingSameDirection</b> ( $directionFromTo(B,A)$ , west)
NextToLeft	less(distanceBetween(A, B), 2) +
	pointingSameDirection(directionFromTo(B, A), direction(B, left))
NextToRight	less(distanceBetween(A,B),2) +
	pointingSameDirection(directionFromTo(B, A), direction(B, right))
NextToFront	less(distanceBetween(A,B),2) +
	pointingSameDirection(directionFromTo(B,A), direction(B, forward))
NextToBehind	less(distanceBetween(A,B),2) +
	pointingSameDirection(directionFromTo(B, A), direction(B, backward))
OnTopOf	$\textbf{equals}(boundary(A, \mathbf{up})_y, boundary(B, \mathbf{down})_y) +$
IsPartOf	$\mathbf{range}(\operatorname{center}(A)_x, boundary(B, \operatorname{left})_x, boundary(B, \operatorname{right})_x) +$
	$\mathbf{range}(\operatorname{center}(A)_z, boundary(B, \operatorname{forward})_z, boundary(B, \operatorname{backward})_z)$
OnTopOfMiddle	$\textbf{equals}(boundary(A, \mathbf{up})_y, boundary(B, \mathbf{down})_y) +$
IsPartOfMiddle	equals( $\ \operatorname{center}(A)_{xz} - \operatorname{center}(B)_{xz}\ , 0$ )
OnTopOfEdge	$\mathbf{equals}(boundary(A, \mathbf{up})_y, boundary(B, \mathbf{down})_y) + \mathbf{equals}(\min(B, \mathbf{up})_y) + \mathbf{equals}(\mathbf{up})_y) + \mathbf{equals}(\mathbf{up})_y + \mathbf{equals}(\mathbf{up})_$
IsPartOfEdge	distance To Boundary(A, B, forward), distance To Boundary(A, B, backward)
	distanceToBoundary(A, B, left), distanceToBoundary(A, B, right), 0)
OnTopOfFrontEdge	$\mathbf{equals}(boundary(A, \mathbf{up})_y, boundary(B, \mathbf{down})_y) +$
IsPartOfFrontEdge	$\mathbf{equals}(distanceToBoundary(A, B, forward), 0)$
FacingTowards	pointingSameDirection(directionFromTo(A,B), direction(A, forward))
ImplicitFacingTowards	
FacingAwayFrom	$\mathbf{range}(direction(A, \text{forward}) \cdot direction(B, \text{forward})), -1, 0)$
ImplicitFacingAwayFrom	
FacingSameDirection	pointingSameDirection(direction(A, forward), direction(B, forward))
ImplicitFacingSameDirection	
FacingOrthogonalDirection	equals $(\cos^{-1}(direction(A, forward) \cdot direction(B, forward))), \frac{\pi}{2})$
ImplicitFacingOrthogonalDirection	
PathConnectedTo	0

**Table 1:** Translation of edge types from the scene relationship graph to evaluation factors in log space. Each edge points from an object A to an object B. Factor functions are printed in bold. They are log space versions of the functions [YYW\*12] employed (see Equation 1).

For these functions no further implementation is detailed, since they are highly dependent on the collision geometries used.

Objects in the scene should not collide with each other. The function  $v_c$  penalizes collisions between each object pair from the set of objects in the scene O and can be written as:

$$v_{c} = \sum_{(A,B) \in O \times O, A \neq B} collisionPenalty(A,B)$$

$$collisionPenalty(A,B) = \begin{cases} log((1-penetrationDepth(A,B))^{c}) \\ & \text{if } A \text{ and } B \text{ collide} \\ 0 & \text{otherwise.} \end{cases}$$
(6)

The function penetrationDepth(A,B) calculates the percentage that the collision geometries of objects A and B penetrate into each other. In other words, penetrationDepth(A,B) returns the minimum distance that one of the two objects needs to move so that their collision geometries do not intersect anymore, normalized to [0,1] using the summed up size of both objects. An intensity constant c controls how much more large intersections should be penalized compared to small intersections. Since  $penetrationDepth(A,B) \in [0,1]$ , a larger c will penalize big penetration depths much more. The choice of c additionally affects, which priority the avoidance of collisions should have relative to other constraints.

The paths in the scene layouts are made up of path segments with control points,  $v_p$  and  $v_q$  of the evaluation function are defined to yield reasonable path layouts. Given a set of path segments P and objects O,  $v_p$  penalizes collisions between path segments and objects similar to  $v_c$ :

$$v_p = \sum_{A \in O.B \in P} collisionPenalty(A, B). \tag{7}$$

Here, the function collisionPenalty(A,B) is defined the same way as in equation 6 above.  $v_q$  encourages each path segment from P to be in a specific shape. For each path segment  $A \in P$  consisting of  $n_A$  control points,  $C_{A,i}$  with  $1 < i < n_A$  denotes the individual control points of the segment and  $l_A$  the desired maximum length;  $v_q$  is defined as:

$$v_{q} = \sum_{A \in P} \mathbf{less}(totalLength(A), l_{A}) + \sum_{A \in P} \sum_{1 < i < n_{A}} \mathbf{less}(angleBetween(C_{A,i-1}, C_{A,i}, C_{A,i+1}), \gamma) + \sum_{\alpha \in anglesIntersectionPaths(P)} \mathbf{greater}(\alpha, \gamma).$$
(8)

The factors applied in  $v_q$  can be explained the following way:

- Σ<sub>A∈P</sub> less(totalLength(A), l<sub>A</sub>) encourages the path segment to be shorter than its assigned maximum length, with totalLength(A) summing up the length of all lines the path segment draws.
- $\sum_{A \in P} \sum_{1 < i < n_A} \mathbf{less}(angleBetween(C_{A,i-1}, C_{A,i}, C_{A,i+1}), \gamma)$  penalizes angles which are bigger than some constant  $\gamma$  (here chosen to be  $60^{\circ}$ ). The angles are between lines connecting successive control points  $C_{A,i-1}, C_{A,i}, C_{A,i+1}$ .

•  $\sum_{\alpha \in anglesIntersectionPaths(P)}$  **greater**( $\alpha, \gamma$ ) makes all outgoing paths segments of an intersection object have at least a 60 degree angle between them. A function with the name anglesIntersectionPaths(P) returns the set of all angles between neighboring outgoing paths segments from all intersection objects. For scenes without path intersections it yields an empty set.

Notably missing from constraints  $v_q$  and  $v_c$  is collision between path segments. This is acceptable, since given the above limitations on path segment lengths and angles, collisions of path segments can only happen rarely unless many path segments in the scene have a large number of control points.

## 2. Additional Details about the Pilot User Study

Participants were informed about the purpose of the study and gave their informed consent. They received no financial compensation and the study was conducted online. The developed web interface allowed to explore the scenes by rotating the camera and zooming in and out.

At the beginning of the questionnaire, subjects were asked to assess on a Likert scale their familiarity with the overall project as well as with 3D virtual worlds (video games, simulations) in general. No significant correlation between these self assessments and the later study results were found. An optional post-hoc questionnaire allowed participants to rate difficulty and clarity of the study, and to provide remarks in a free-form text field. Due to a technical mishap, in a few rare cases participants were able to rate descriptions they themselves provided in the previous session. Nevertheless, excluding these cases did not change significance of the results or other findings.

### 3. Scene Variety, Model Database and Further examples

To generate a wide variety of scene layouts 250 models were obtained from the Unity Asset Store (https://assetstore.unity.com/packages/3d/environments/fantasy/fantasy-village-pack-140303) and annotated with metadata to populate the scenes. For the graph grammar, over 100 different production rules were hand-designed in about 30 person-hours.

Further examples of generated scenes with descriptions demonstrating some of the variety and complexity possible are presented below. For each scene the automated description, the highest rated and the lowest rated user description is shown. Examples where the developed method worked well are given in Figures 1 and 2. An example where both the scene layout and the the textual description exhibit major flaws is shown in Figure 3. Failure cases are presented in Figures 4 and 5.

#### References

[YYW\*12] YEH Y.-T., YANG L., WATSON M., GOODMAN N. D., HANRAHAN P.: Synthesizing open worlds with constraints using locally annealed reversible jump MCMC. *ACM Trans. on Graphics 31*, 4 (July 2012). 1, 2

There is a dirt path with a crossroads. Next to it there is a small forest with thicket. The path has a place with two meal counters and one plants counter.

### Human A

a mostly empty field with a few trees and two streets that intersect. on one street is a small. there is a small square on one road with two tables and three poleposts



**Generated 3D Scene** 

The scene consists of three roads meeting at an intersection, a group of trees, an oak tree and three market stands. The three market stands are next to the first road. The group of trees consists of three pine trees and three bushes. The first market stand consists of a sign to the right of a table. A big pot of stew is in the middle of this table. The second market stand consists of a sign besides of a table. A big pot of stew is in the middle of this table. The third market stand consists of three flowerpots on top of a table and a sign. This sign is to the right of this table.

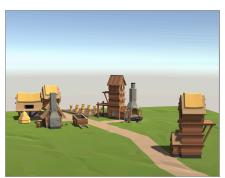
Human B Machine

Figure 1: Procedurally generated 3D scene with textual descriptions. (Left:) Two descriptions by human users – rated as most (top) and least (bottom) accurate in our pilot study; (Right:) automatically generated description with our framework.

A road crosses thorugh a plain. Near the middle of the road, there is a small village, containing a three story smithy, a grain fiels and two houses on the opposite side of the road. One of the two houses is a one story building, the other one a two story building with a small cart and flower pot in front. Further down the road, there is an additional three story house.

#### Human A

I see more houses as in the first scene. There is wheat field with straw bales. A farm house and a hand cart.



**Generated 3D Scene** 

The scene consists of a farmstead, a road, two towers, a blacksmith and a clothes line. The farmstead, the blacksmith and the first tower are next to the road. The blacksmith is besides of the other tower. The clothes line and a sign are behind of this tower. The farmstead consists of a wheat sheaf besides of a shed, a field, a house and an empty wagon. A flowerpot is north of the house. Besides of it is the empty wagon. A barrel is next to the house. A blacksmith forge is facing away from the other tower. The blacksmith consists of a brazier in front of it and the sign. An anvil is in front of the brazier. A bucket of coal is besides of the brazier.

Human B Machine

Figure 2: Procedurally generated 3D scene with textual descriptions. (Left:) Two descriptions by human users – rated as most (top) and least (bottom) accurate in our pilot study; (Right:) automatically generated description with our framework.

I see paths crossing a green lawn. On the paths there are lights or lanterns on wooden poles. On an intersection of the paths a house is placed. Next to the house I see a wagon with some boxes inside. On the lawn three trees are standing.

# Human A

**Human B** 

I can see a house, three trees, six lantern masts and a wheelbarrow. There is a path with two turnings. Everything seems to be made out of wood.



**Generated 3D Scene** 

The scene consists of four roads meeting at an intersection, a group of trees, a house, a wagon of crates and six street lamps. The wagon of crates is next to the first road. The first street lamp is next to the intersection. The house is next to the second road. Four of the street lamps are next to the third road. The sixth street lamp is next to the fourth road. The group of trees consists of three pine trees.

Machine

Figure 3: Procedurally generated 3D scene with textual descriptions with major flaws. The scene layout displays too many street lanterns placed chaotically and the description incorrectly describes the presented trail layout. (Left:) Two descriptions by human users - rated as most (top) and least (bottom) accurate in our pilot study; (Right:) automatically generated description with our framework.

A street runs from the middle lower side in direction of the uper right side but makes a curve to the uper left side in the uper half of the scene. Two houses are positioned on the left side of the street: a smith and a mill. The main house of both of them is small and three floors tall with balconies on the back side. The smith as a forging furnace on its right side, with a additional coal pan, an anvil an a table. The mill has a wind wheel attached to the top of the main house. On the right side are some bundles of corn, on the left side some filled bags and a hand cart with what seemed to be flour.

## Human A

Human B

There is a mil, a normal house, a oven, a lantern mast and a wheelbarrow in this scene. There are more details like the fire or the hay in this scene.



**Generated 3D Scene** 

intersection, a mill, a tower, a blacksmith and a street lamp. The mill and the street lamp are next to the first road. The blacksmith is next to the second road. It is besides of the tower. The mill consists of a windmill building, a wagon of grain, two pallets of grain sheafs and four sacks of flour. Both pallets of grain sheafs and two of the sacks of flour are besides of the windmill building. The wagon of grain and two of the sacks of flour are to the left of the windmill building. A blacksmith forge is facing away from the tower. A sign is in front of the tower. The blacksmith consists of a brazier in front of the blacksmith forge, the sign and a table. An anvil is in front of the brazier. On top of it is a hammer. The table is in front of the sign.

The scene consists of three roads meeting at an intersection, a tower, an empty wagon and four market stands. The tower and the four market stands are next to the first road. The empty wagon is next to the second road. The first market stand consists of a sign to the left of a

table, two boxes of wine bottles and two bottles.

Two of the boxes of wine bottles and two of the

bottles are on top of this table. The second

market stand consists of two books on top of a

table and a sign. This sign is besides of this

table. A chair is behind of this table. The third market stand consists of two books on top of a table and a sign. A chair is behind of this table.

The sign of the third market stand is to the right of this table. The fourth market stand consists of

a sign to the right of a table, two boxes of wine

bottles and two bottles. Two of the boxes of wine

bottles and two of the bottles are on top of this

The scene consists of three roads meeting at an

Machine

**Figure 4:** Procedurally generated 3D scene example with textual descriptions. The generated trails in the scene overlap each other, resulting in a scene with an incorrect description. (Left:) Two descriptions by human users – rated as most (top) and least (bottom) accurate in our pilot study; (Right:) automatically generated description with our framework.

One can see a road. At the end of the road, there is an empty cart. In the center of the scene, there are some sign posts. Next to each sign post there is a table with either books or bottles on top of it. At the beginning of the road there is a building next to it.

# Human A

**Human B** 

I see a green lawn crossed by a path. Next to the path is a wagon and a bit further there are some sales stalls. On the other end of the path there is a small tower. The sky is blue.

**Generated 3D Scene** 

Machine

**Figure 5:** Procedurally generated 3D scene example with textual descriptions. Many generated objects and the trails clip into each other, resulting in a not plausible scene with an incorrect description. (Left:) Two descriptions by human users – rated as most (top) and least (bottom) accurate in our pilot study; (Right:) automatically generated description with our framework.