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

J. Müller-Huschke, M. Ritter, and M. Hardera
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 $N(x, \mu, \sigma^2)$ and a sigmoid function $\text{Sig}(x,h) = \frac{1}{1+e^{-x}}$:

$$f = e^{-v}.$$  \hspace{1cm} (3)

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

$$v = v_r + v_c + v_p + v_q.$$  \hspace{1cm} (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:

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

- $\text{distanceBetween}(A, B)$ gives the minimum distance that can be found between the collision geometries of objects $A$ and $B$.
- $\text{directionFromTo}(A, B)$ gives the normalized direction vector pointing from the position of $A$ to the position of $B$.
- $\text{direction}(A, \vec{x})$ gives the direction vector $\vec{x}$ transformed to the local coordinate system of object $A$.
- $\text{boundary}(A, \vec{x})$ 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.
- $\text{distanceToBoundary}(A, B, \vec{x})$ gives the distance between the collision geometry of $A$ and the boundary in $\vec{x}$ direction of the collision geometry of $B$. For convex collision shapes this boundary would most likely be a point, but for box colliders it can be a plane.

---

\*Corresponding Author:
<table>
<thead>
<tr>
<th>Edge Label from A to B</th>
<th>Resulting factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextToNextToChooseSide</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + max($\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, left$))), $\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, right$)))</td>
</tr>
<tr>
<td>NextToNextToNorth</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), north)</td>
</tr>
<tr>
<td>NextToNextToEast</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), east)</td>
</tr>
<tr>
<td>NextToNextToSouth</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), south)</td>
</tr>
<tr>
<td>NextToNextToWest</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), west)</td>
</tr>
<tr>
<td>NextToNextToLeft</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, left$))</td>
</tr>
<tr>
<td>NextToNextToRight</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, right$))</td>
</tr>
<tr>
<td>NextToNextToFront</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, forward$))</td>
</tr>
<tr>
<td>NextToNextToBehind</td>
<td>$\text{less}(\text{distanceBetween}(A, B), 2)$ + $\text{pointingSameDirection}$(directionFromTo($B, A$), direction($B, backward$))</td>
</tr>
<tr>
<td>OnTopOfOnTopOf</td>
<td>$\text{equals}$(boundary($A, up$), boundary($B, down$)) + range(center($A, x$), boundary($B, left$), boundary($B, right$)) + range(center($A, z$), boundary($B, forward$), boundary($B, backward$))</td>
</tr>
<tr>
<td>IsPartOfIsPartOf</td>
<td>$\text{equals}$(boundary($A, up$), boundary($B, down$)) + $\text{equals}$(min(distanceToBoundary($A, B, forward$), distanceToBoundary($A, B, backward$)), distanceToBoundary($A, B, left$), distanceToBoundary($A, B, right$), 0)</td>
</tr>
<tr>
<td>OnTopOfFrontEdge</td>
<td>$\text{equals}$(boundary($A, up$), boundary($B, down$)) + $\text{equals}$(distanceToBoundary($A, B, forward$), distanceToBoundary($A, B, backward$), 0)</td>
</tr>
<tr>
<td>IsPartOfFrontEdge</td>
<td>$\text{pointingSameDirection}$(directionFromTo($A, B$), direction($A, forward$))</td>
</tr>
<tr>
<td>FacingTowardsFacingAwayFrom</td>
<td>$\text{range}$(direction($A, forward$) · direction($B, forward$), $-1, 0$)</td>
</tr>
<tr>
<td>ImplicitFacingAwayFrom</td>
<td>$\text{pointingSameDirection}$(direction($A, forward$), direction($B, forward$))</td>
</tr>
<tr>
<td>FacingSameDirectionImplicitFacingSameDirection</td>
<td>$\text{equals}$(cos$^{-1}$(direction($A, forward$) · direction($B, forward$)), $\frac{\pi}{2}$)</td>
</tr>
<tr>
<td>FacingOrthogonalDirectionImplicitFacingOrthogonalDirection</td>
<td>0</td>
</tr>
</tbody>
</table>

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 \( v_p \) penalizes collisions between each object pair from the set of objects in the scene \( O \) and can be written as:

\[
\begin{align*}
  v_p &= \sum_{A \in O, B \in \mathcal{P}} \text{collisionPenalty}(A, B) \\
  \text{collisionPenalty}(A, B) &= \begin{cases} 
  \log((1 - \text{penetrationDepth}(A, B))^c) & \text{if } A \text{ and } B \text{ collide} \\
  0 & \text{otherwise}.
  \end{cases}
\end{align*}
\]

The function \( \text{penetrationDepth}(A, B) \) calculates the percentage that the collision geometries of objects \( A \) and \( B \) penetrate into each other. In other words, \( \text{penetrationDepth}(A, B) \) 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 \( \text{penetrationDepth}(A, B) \in [0, 1]\), 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 \( \mathcal{P} \) and objects \( O \), \( v_p \) penalizes collisions between path segments and objects similar to \( v_c \):

\[
  v_p = \sum_{A \in \mathcal{P}, B \in O} \text{collisionPenalty}(A, B).
\]

Here, the function \( \text{collisionPenalty}(A, B) \) is defined the same way as in equation 6 above. \( v_q \) encourages each path segment from \( \mathcal{P} \) to be in a specific shape. For each path segment \( A \in \mathcal{P} \) 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:

\[
\begin{align*}
  v_q &= \sum_{A \in \mathcal{P}} \text{less}(\text{totalLength}(A), L_A) + \\
  \sum_{A \in \mathcal{P}, 1<i<n_A} \text{less}(\text{angleBetween}(C_{A,i-1}, C_{A,i}, C_{A,i+1}), \gamma) + \\
  \sum_{\alpha \in \text{anglesIntersectionPaths}(\mathcal{P})} \text{greater}(\alpha, \gamma).
\end{align*}
\]

The factors applied in \( v_q \) can be explained the following way:

- \( \sum_{A \in \mathcal{P}} \text{less}(\text{totalLength}(A), L_A) \) encourages the path segment to be shorter than its assigned maximum length, with \( \text{totalLength}(A) \) summing up the length of all lines the path segment draws.
- \( \sum_{A \in \mathcal{P}, 1<i<n_A} \text{less}(\text{angleBetween}(C_{A,i-1}, C_{A,i}, C_{A,i+1}), \gamma) \) penalizes angles which are bigger than some constant \( \gamma \) (here chosen to be \( 60^\circ \)). The angles are between lines connecting successive control points \( C_{A,i-1}, C_{A,i}, C_{A,i+1} \).
- \( \sum_{\alpha \in \text{anglesIntersectionPaths}(\mathcal{P})} \text{greater}(\alpha, \gamma) \) makes all outgoing paths segments of an intersection object have at least a 60 degree angle between them. A function with the name \( \text{anglesIntersectionPaths}(\mathcal{P}) \) 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 \( v_p \) and \( v_q \) 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 textual description exhibit major flaws is shown in Figure 3. Failure cases are presented in Figures 4 and 5.

### References

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.

**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 street runs from the middle lower side in direction of the upper right side but makes a curve to the upper left side in the upper 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 an additional coal pan, an anvil and 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.

The scene consists of three roads meeting at an 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.

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.

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.