Urban space design has been of continued interest for several centuries and procedural modelling is a tool that successfully addresses the challenges in the generation of cities. Procedural production systems are traditionally grammar-based, each one virtually describing a new programming language. Content generators using these tools need not only to have certain artistic sensitivity but also an ability to write computer programs.

There have been two recent directions that aim to ease the use of procedural systems. The first one, inverse procedural modelling, includes both systems that generate the production rules from square one [3, 5], and techniques that fit the parameters of existing grammars [6, 7]. The second one can be denoted interactive procedural modelling and introduces tools that leverage the troublesome process of parameter tuning of existing grammars, such as handles and local edits [1, 2].

We introduce in this paper the concept of differentiable procedural modelling, which enhances inverse modelling with tools from interactive methods. Procedural models have lots of confusing parameters which only the original programmer understands. We introduce an interactive tool which lets users perform direct modifications on the geometry of models, while it fits the parameters of a grammar to match those modifications. Formally, a differentiable procedural system is a parametrised production system [8], that is, a system composed of:

- An algebra of objects $U$, defined as $(U, +, \cdot, F \leq)$, where the set $U$ is closed under the insertion $\cdot$ and deletion $-$ operations and all transformations $f \in F$. An object $u \in U$ is said to occur in another object $v \in U$ if and only if there exists a transformation $f \in F$ such that $f(u) \leq v$.
- A set of production rules $R \subseteq U \times U$.
- A set of initial objects $I \subseteq U$.
- An interpretative mechanism, by which objects are generated by the repeated application of production rules to an initial object. Given an initial object $w \in U$, a production rule $u \rightarrow v$, and a variable assignment $g$, if there exists a transformation $f \in F$ such that $f(u) \leq w$, then the production rule can produce the new object:

$$[w - f(g(u))] + f(v)$$

such that:

- for each shape $w \in U$, each production rule $u \rightarrow v \in R$, each transformation $f \in F$ and each variable assignment $g$, we have that for every object $\tau \in U$ with

$$\tau \leq [w - f(g(u))] + f(v) \in U$$

we can compute its rate of change (a.k.a., differentiate $\tau$) with respect every variable assignment $g^i$ applied in the generation of the shape

$$[w - f(g(u))] + f(v) \in U$$

For a geometric interpretation of a differentiable procedural system, we develop a C++ implementation of CGA Shape grammars [4]. CGA Shape is a language for the procedural modelling of architecture. Our system works as follows. Given a set $p = \{p_i\}_{i=1}^n$ of $n$ input parameters, the procedural systems produces a geometry $G$. The user can change the parameters $p$ to compute $G$, as in forward modelling. Interactively, the user can select with the mouse a vertex $v$ in $G$ and displace it to a different location $w$ (inverse modelling), and our system automatically updates $p$ to satisfy such a modification. We solve the minimization problem

$$\mathbf{p}' = \arg \min_{\mathbf{p} \in \mathbb{R}^n} \mathcal{d}(v(p), w)$$

for some distance measure $\mathcal{d}$, which defaults to an Euclidean metric. Optionally, each parameter value $p_i$ can be constrained within a plausible range $[p_{i_{\text{min}}}, p_{i_{\text{max}}}]$ or fixed to the current value if it corresponds to some feature of the geometry $G$ that should stay constant. The (local) optimum is computed by a standard gradient-based non-linear function optimisation routine using a quasi-Newton strategy.