As detailed previously, there are two main ways to initialise a GE individual: by generating a [[genome|Representation#linear-genome-representation]], or by generating a [[derivation tree|Representation#derivation-tree-representation]]. Generation of a genome does not take tree size/shape into account. Population initialisation via derivation tree generation on the other hand allows for fine control over many aspects of the initial population, e.g. the distribution of depth limits or derivation tree shape. Unlike with genome initialisation, there are a number of different ways to initialise a population using derivation trees. Currently implemented methods are detailed below.

Genome

Random

The simplest and worst-performing initialisation approach is to generate individuals from uniformly generated genomes. Use command-line argument:

```
--initialisation uniform_genome
```

or set the parameter `INITIALISATION` to `uniform_genome` in either a parameters file or in the params dictionary.

NOTE *that random genome initialisation in Grammatical Evolution should be used with caution as poor grammar design can have a negative impact on the quality of randomly initialised solutions due to the inherent bias capabilities of GE* [[[Fagan et al., 2016|References]]; [[Nicolau & Fenton, 2016|References]]].

An improved approach, *random with valids only and no duplicates* (named RVD and described by Nicolau) just discards invalid individuals and those with duplicate phenotypes. According to Nicolau it performs well, while still being easy to implement:

```
--initialisation rvd
```

or set the parameter `INITIALISATION` to `rvd` in either a parameters file or in the params dictionary.

By default in PonyGE2, genomes of length 200 codons are generated when using random genome initialisation. However, this parameter can be changed using the argument:

```
--init_genome_length [INT]
```

or by setting the parameter `INIT_GENOME_LENGTH` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the length of genomes to be initialised.

Derivation Tree

There are currently three options provided in PonyGE2 for initialising a population of individuals using derivation tree methods. You can either initialise a population of random derivation trees, or you can use various "smart" initialisation methods implemented here.

Random

Random derivation tree initialisation generates individuals by randomly building derivation trees up to the specified maximum initialisation depth limit.

Activate with:

```
--initialisation uniform_tree
```

or by setting the parameter `INITIALISATION` to `uniform_tree` in either a parameters file or in the params dictionary.

NOTE *that there is no obligation that randomly generated derivation trees will extend to the depth limit; they will be of random size* [[[Fagan et al., 2016|References]]].

NOTE *that randomly generated derivation trees will have a tendency towards smaller tree sizes with the use of a grammar-based mapping* [[[Fagan et al., 2016|References]]].

Ramped Half-Half

Ramped Half-Half initialisation in Grammatical Evolution is often called "Sensible Initialisation" [[[Ryan and Azad, 2003|References]]]. Sensible Initialisation follows traditional GP Ramped Half-Half initialisation by initialising a population of individuals using two separate methods: `Full` and `Grow`.

`Full` initialisation generates a derivation tree where all branches extend to the specified depth limit. This tends to generate very bushy, evenly balanced trees [[[Fagan et al., 2016|References]]].

`Grow` initialisation generates a randomly built derivation tree where no branch extends *past* the depth limit.

NOTE *that* `Grow` *is analogous to random derivation tree initialisation, i.e. no branch in the tree is* ***forced*** *to reach the specified depth. Depending on how the grammar is written, this can result in a very high probability of small trees being generated, regardless of the specified depth limit.*

Activate with:

```
--initialisation rhh
```

or by setting the parameter `INITIALISATION` to `rhh` in either a parameters file or in the params dictionary.

RHH initialisation generates pairs of solutions using both `full` and `grow` methods for a ramped range of depths. The maximum initialisation depth is set with the argument:

```
--max_init_tree_depth [INT]
```

or by setting the parameter `MAX_INIT_TREE_DEPTH` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the maximum depth to which derivation trees are to be initialised. The default value is set at 10.

By default in PonyGE, initialisation ramping *begins* at a depth where sufficient unique solutions can be generated for the number of required solutions at that depth [[[Nicolau & Fenton, 2016|References]]]. However, this value can be over-written in favor of a user-defined minimum ramping depth. This can be set with the argument:

```
--min_init_tree_depth [INT]
```

or by setting the parameter `MIN_INIT_TREE_DEPTH` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the minimum depth from which derivation trees are to be initialised.

NOTE *that RHH initialisation with the use of a grammar-based mapping process such as GE can potentially result in a high number of duplicate individuals in the initial generation, resulting from a potentially high number of very small solutions* [[[Nicolau & Fenton, 2016|References]]; [[Fagan et al., 2016|References]]]. *As such, caution is advised when using RHH initialisation in grammar-based systems, as particular care needs to be given to grammar design in order to minimise this effect* [[[Fagan et al., 2016|References]]].

Position Independent Grow (PI Grow)

Position Independent Grow (PI Grow) initialisation in Grammatical Evolution mirrors Sensible/Ramped Half-Half initialisation by initialising a population of individuals over a ramped range of depths. However, while RHH uses two separate methods `Full` and `Grow` to generate pairs of individuals at each depth, PI Grow eschews the `Full` component and only uses the `Grow` aspect. There are two further differences between traditional GP `Grow` and PI Grow [[[Fagan et al., 2016|References]]]:

1. At least one branch of the derivation tree is forced to the specified maximum depth in PI Grow, and
2. Non-terminals are expanded in random (i.e. position independent) order rather than the left-first derivation of traditional mappers.

Activate with:

```
--initialisation PI_grow
```

or by setting the parameter `INITIALISATION` to to `PI_grow` in either a parameters file or in the params dictionary.

As with RHH initialisation, PI Grow initialisation generates individuals for a ramped range of depths. The maximum initialisation depth is set with the argument:

```
--max_init_tree_depth [INT]
```

or by setting the parameter `MAX_INIT_TREE_DEPTH` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the maximum depth to which derivation trees are to be initialised. The default value is set at 10.

By default in PonyGE, initialisation ramping *begins* at a depth where sufficient unique solutions can be generated for the number of required solutions at that depth [[[Nicolau & Fenton, 2016|References]]]. However, this value can be over-written in favor of a user-defined minimum ramping depth. This can be set with the argument:

```
--min_init_tree_depth [INT]
```

or by setting the parameter `MIN_INIT_TREE_DEPTH` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the minimum depth from which derivation trees are to be initialised.