Evolving Planets User Manual

1 Functionalities Overview

This software produces a collection of 3D models that represent planets of arbitrary shapes. The software can generate both realistic-looking planetoids, e.g. asteroids, and absurd shapes. The user can customize the process through a variety of parameters. The planets are first randomly generated through an initialization stage; then an evolutionary algorithm can be executed for optimising their landing difficulty. The generated planets can be visualised through a built-in renderer, and exported as Wavefront obj files.

2 UI Overview

As shown in figure 1, the screen is split into three parts: one on the left, one at the bottom and a central one. The first contains two tabs, *EA parameters* and *Controls*; *EA parameters* lists all the parameters, while *Controls* contains the controls for actually using the software's functionalities; the bottom part lists some parameters related to the planets' visualisation; the central part is the viewport of the build-in renderer, where the generated planets will be displayed, one at a time.

3 Keyboard Controls

After the planets have been generated, the user can switch the displayed planet with the left and right arrows on the keyboard; on the *Controls* tab, a label shows the index of the current planet.

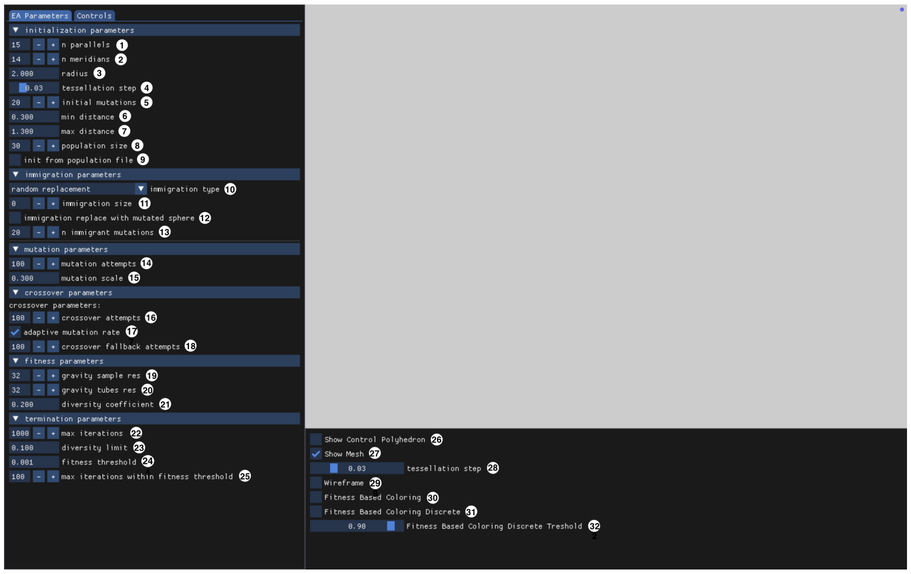
4 Detailed Parameters Description

Every element of the user interface will be described in detail; in figure 1 and 2 each element is numbered so that it is referenced by the list below.

1. number of parallels of the control polyhedron; take into consideration that the first 3 and the last 3 parallels are dedicated to the poles and their plateau, thus they do not participate in shaping the middle portion of the shape. (default: 14; recommended: [12, 20])

2. number of meridians of the control polyhedron; the number of parallels and meridians determines the degree of complexity of the shape; the recommended values are a compromise between simplicity and complexity; (default: 14; recommended: [10, 20])

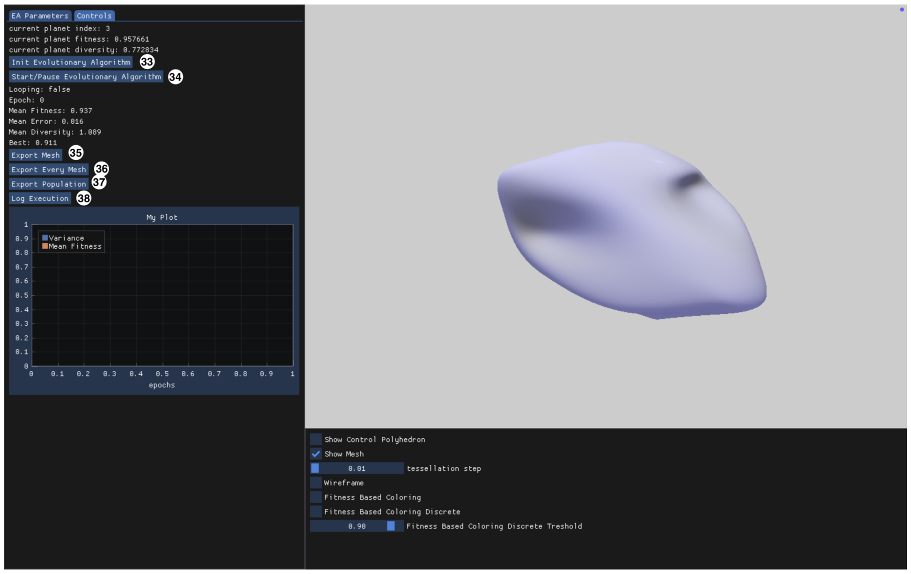
3. radius of the starting spheres: each planet is created as a sphere, and then it is subject to multiple mutations during initialization; it should not be changed (default: 2.0; recommended: 2.0)



```
15      n parallels 1
14      n meridians 2
2.000   radius 3
0.03    tessellation step 4
20      initial mutations 5
0.300   min distance 6
1.300   max distance 7
30      population size 8
[ ] init from population file 9
random replacement   immigration type 10
0       immigration size 11
[ ] immigration replace with mutated sphere 12
20      n immigrant mutations 13
100     mutation attempts 14
0.300   mutation scale 15
crossover parameters:
100     crossover attempts 16
[✓] adaptive mutation rate 17
100     crossover fallback attempts 18
32      gravity sample res 19
32      gravity tubes res 20
0.200   diversity coefficient 21
1000    max iterations 22
0.100   diversity limit 23
0.001   fitness threshold 24
100     max iterations within fitness threshold 25

[ ] Show Control Polyhedron 26
[✓] Show Mesh 27
0.03    tessellation step 28
[ ] Wireframe 29
[ ] Fitness Based Coloring 30
[ ] Fitness Based Coloring Discrete 31
0.90    Fitness Based Coloring Discrete Treshold 32
```

Figure 1: Main Screen



```
current planet index: 3
current planet fitness: 0.957661
current planet diversity: 0.772834
Looping: false
Epoch: 0
Mean Fitness: 0.937
Mean Error: 0.016
Mean Diversity: 1.089
Best: 0.911

[ ] Show Control Polyhedron
[✓] Show Mesh
0.01    tessellation step
[ ] Wireframe
[ ] Fitness Based Coloring
[ ] Fitness Based Coloring Discrete
0.90    Fitness Based Coloring Discrete Treshold
```

Figure 2: Algorithm Controls

4. tessellation step used by the auto-intersection test procedure: the surface is turned into triangles by sampling the square domain; the tessellation step determines the sampling resolution (the step is the distance between

two samples). If generating production-ready planets, it should be set to 0.01, because lower values can lead to invalid shapes. At 0.03 precision is still decent, and the process becomes a lot faster. (default: 0.03; recommended: [0.01, 0.03])

5. number of mutations each planet is subject to during the initialization process; each single mutation is a gaussian terraformation; if the user intends to use the optimization algorithm to generate a single planet, a value of few tens is fine, while if the user wants to employ initialization as the only process, the higher the number, the more complex and more optimized the planets will be; instead, low values like 20 can generate smooth shapes that resemble real asteroids. (default: 30)

6. minimum intensity for the gaussain terraformations; each mutation has a random intensity between a minimum and a maximum value (default: 0.3; recommended: [0.1, 0.5])

7. maximum intensity for the gaussian terraformations; (default: 1.3; recommended: [0.7, 1.5])

8. size of the generated population; if initialization is the only generative process, this parameter is arbitrary; otherwise, it should be as high as possible; (default: 30)

9. check if the population should be loaded from disk instead of being initialised as a brand new one. (default: unchecked)

10. select the type of immigration: *random replacement* or *replace the most similar*: *random replacement* puts new individuals in place of random individuals of the population; *replace the most similar* puts the new individuals in place of the planets that have lower diversity. The second setting should help fighting the premature shape convergence by breaking the similarities; (default: *random replacement*; recommended: *replace the most similar*)

11. number of brand new planets that are injected during each cycle in place of planets of the current population; if 0 the immigration system is completely turned off; immigration size should be set at 0, or set at low values, like 5% of the population size; (default: 0; recommended [0, $0.1 \cdot population\_size$])

12. check if the immigrants should be mutated spheres; otherwise immigrants are the very same planets that are going to be substituted but mutated for making them different from before; (default: checked; recommended: checked)

13. number of mutations immigrants (whether they are spheres or the planets that are substituting) are subject to; (default: 20)

14. number of mutation attempts: each cycle, each planet is subject to a mutation process; when a planet is modified or generated, it must pass the auto-intersection test: if it fails, the procedure should be repeated; if the number of mutation attempts goes beyond this parameter, then the planet is not mutated; the parameter should be set as high as possible; (default: 100)

15. mutation scaling factor; it should be between 0 and 1; the higher the value, the higher the mutation intensity; (default: 0.3)

16. same as mutation attempts; when the number of attempts is exceeded, the crossover procedure switches from uniform to continuous, which is a safer technique; it the continuous crossover attempts are also exceeded, the child will simply be a mutated planet;

17. if checked, the mutation rate becomes adaptive, making the crossover procedure progressively safer; should be left checked.

18. number of continuous crossover attempts;

19. resolution value for the fitness computation procedure; fitness is computed for a number of locations sampled on the planet surface; each domain axis is subdivided by this value, for example at 32, the planet is sampled 32*32=1024 times; this value should be as high as possible, but it severely hinders performance; keep at 32 if unsure, increase if possible; keep in mind that this value is related to the optimization process; it doesn't affect the initialization process results, but it still affects its performance, so keep it low if only random inizialization is desired; (default: 32; recommended: 64 or higher)

20. internal parameter for gravity computation; (default: 32; recommended: 64 or higher)

21. multiplicative coefficient between 0 and 1 that adds diversity value to the fitness computation; (default: 0.2; recommended: [0.0, 0.3])

22. number of iterations after which the aglorithm is terminated (default: 1000)

23. diversity values under which the algorithm is terminated; should be 0 if diversity is not a concern; in general the algorithm is able to produce a single optimized planet and cannot efficiently maintain diversity across the population; therefore, it should be set to 0; (default: 0.1; recommended: [0, 0.2])

24. threshold that refers to the change in average fitness between two epochs; if the change remains under this threshold for a set number of consequential epochs, then the algorithm is terminated; (default: 0.001)

25. number of consequential iterations after which, if the improvement is within the fitness threshold, the algorithm is terminated; (default: 100)

26. show the B-Spline control polyhedron

27. show the triangle mesh

28. tessellation step by which the mesh is sampled starting from the B-Spline; (default: 0.03; recommended: [0.01, 0.03])

29. show the mesh as wireframes instead of a solid object

30. enable fitness based coloring

31. make fitness coloring discrete

32. fitness threshold for fitness discrete coloring

33. initialize the algorithm (it only executes the initialization process: after that the user can choose if launching the actual evolution algorithm or keeping the planets as they are); it can be clicked again after a population has been initialized for deleting the current population and generating a new one; if *init from population file* is checked, a file browser will open, and a .population file must be selected

34. start/pause the algorithm; if the algorithm is running, clicking it will actually pause the execution after the current iteration terminates

35. export the current planet as a triangle mesh (as displayed, change the tessellation step if desired) as a Wavefront .obj file.

36. export the whole population as triangles meshes.

37. export the population as a .population file, it can be loaded during initialization instead of creating a brand new population.

38. log a file that lists every parameter and the performance values (fitness, error, diversity) for each executed cycle.