Tom Birdsong
tbirdso
ECE 4420/6420 Section 001
Takehome #2

December 4, 2019

1 Takehome #2 PROPOSAL for a Fuzzy Procedural Generation Engine

1.1 Description

Virtual reality simulations take place in three-dimensional environments which must be constructed and visualized to deliver a full experience. Procedural generation allows developers to construct environments based on present rules and relations among data that can be tailored to a desired environment while eliminating the need to carefully design every facet of the world, in consequence reducing development overhead. For instance, a procedurally generated forest might have many open spaces and different clusters of trees, while a procedurally generated cave might have very few available paths for traversal.

In Takehome #1 an engine was developed to assemble 3D tiles into a traversable dungeon map based on preset adjacency metadata. As this engine produced maps with narrow corridors and tight environments, Takehome #2 aims to generate maps for an alternate case centering on open environments. The project will use fuzzy methods in FuzzyCLIPS to produce an openly-traversable mid-level "boss room" environment from tile presets and properties.

1.2 Resources

- The Project Corridor development team will be consulted to help develop specifications for the generated room landscape outlining what elements may be included and how they should be generated. For example, tiles close to walls may be likely to be "rocky", while tiles near the center of the room may be likely to become "water" tiles.

- The webpage by Isaac Dykeman referenced for procedural design in Takehome #1 will be again referenced for Takehome #2. Dykeman develops a second algorithm, "Most Constrained Placement with Local Information Propagation", that will be adapted to allow deterministic local confidence factor propagation in FuzzyCLIPS. See https://ijdykeman.github.io/ml/2017/10/12/wang-tile-procedural-generation .html [1].

1.3 Goals

- Outline tile elements and adjacency rules for map elements.

- Design checks to ensure preconditions allow for map generation.

- Design local uncertainty propagation rules to inform fuzzy tile sets.

- Design rules to convert fuzzy tile data to crisp tile labels.

1.4 Deliverable

Development will yield a FuzzyCLIPS engine accepting a limited range of input data and returning a crisp m x n labelled grid representing a fully traversable map.

2 Discussion

2.1 Narrowing Scope

The first goal of the project was to develop specifications for how tiles should be procedurally selected to fill an openly-traversable "boss room" map. Rather than allowing the application to procedurally select all tiles, conversations with the Project Corridor development team yielded an alternative vision of a rules-based tool to assist guided map development by "filling" grid regions in a procedural manner. Specifications for the application are as follows:

- The application should generate an $m \times n$ grid of tiles.
- The application should accept user input for the m and n parameters. To narrow the scope of the project soft limits are set such that $0 \leq x \leq 20$ and $0 \leq z \leq 20$.
- The application should accept between zero and $m \times n$ defining ("key") tiles from user input.
- Tiles must take on exactly one of at least three possible labels. For Takehome 2 labels were constrained to the distinct colors "red", "yellow", and "blue", but the label set could be expanded in the future.
- Tiles near to key tiles are more likely to take on the same label.
- Tiles farther away from key tiles are less likely to take on the same label.

2.2 Checking Preconditions

Rules are developed to ensure that appropriate preconditions are met to begin labelling operations.

1. Key tiles may not occupy the same position on the grid. If two key tiles share an (x,z) coordinate then the less recent fact is discarded.
2. Key tiles may not occupy a position outside the stated bounds of the grid. If a key tile is found outside the grid it is discarded.
3. All tiles on the grid must be present in order to be evaluated and labelled by the engine. The program asserts a single fact for each tile position (x,z).

2.3 Set Membership

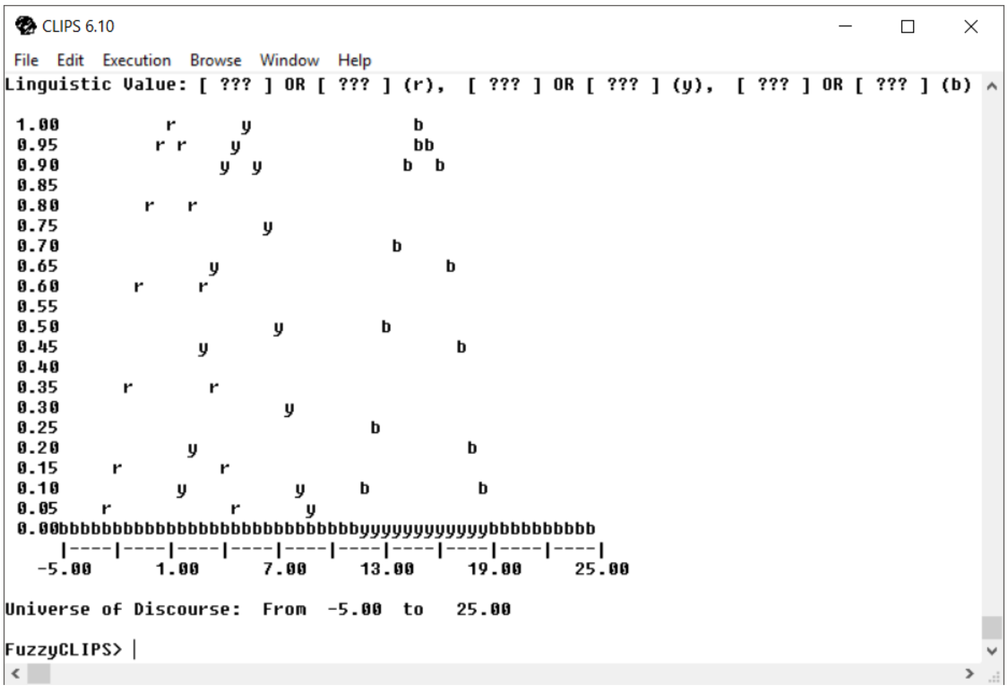
The rules engine application takes inspiration from Dykeman's localized information propagation algorithm to make constrained placements [1]. Rather than backtracking from invalid tile placements the application uses fuzzy logic to allow any one tile to be a simultaneous member of multiple possible variant sets. Set membership is collapsed to assign exactly one crisp label to individual tiles. The degree of membership in each set is utilized to assign confidence to each tile label to inform future assignments.

FuzzyCLIPS 6.31 supports membership functions operating along one-dimensional data, with the User Guide referencing examples on a temperature scale [2] [3]. The room labeling problem relates data in two dimensions, so a solution is implemented to iterate over one-dimensional rows and apply fuzzy logic sequentially. In order to consider two-dimensional relationships a "scanning" solution is implemented in which the scanner iterates over the map grid in rows and then in columns, requiring $m + n$ scanning iterations. Any given iteration considers fuzzy membership on only one dimension, assigning membership based on a tile's position in that row or column.

On each iteration the scanner generates a new membership function for each of the "red", "yellow", and "blue" labels. Initial development efforts attempted to define fuzzy labels for each set using only the `deftemplate` construct; however, requirements for dynamic updates in membership functions necessitated an approach in which each label is asserted as a fact with a dynamic membership function slot. Each membership function is updated to reflect key tiles of its type present in the row or column that are asserted by the user. For example, if a user were to assert a red key tile at position (1,1) with certainty factor 1.0 then when the engine scans row 1 it would join the "red" membership function with a PI function at $x = 1$. The application translates a key tile's certainty factor to inform the width of the base of the PI function. A key tile always has a membership of 1.0 in its respective label set, but its adjacent tiles on the row or column will have greater membership in the set if the key tile has a higher certainty. A sensitivity factor is chosen so that the membership of a tile at most approximately 3 tiles away from a key tile may be influenced by its PI function; this hyperparameter may be adjusted to reflect how much of an "impact" a key tile tends to have on its surrounding tiles.

Sample output from the FuzzyCLIPS window is shown in Figure 1. In this example a red key tile is placed at (0,1), a yellow key tile is placed at (0, 5), and a blue key tile is placed at (0, 15). Note that the universe for fuzzy membership is on the domain [-5,25] which is beyond the tile x-z domain outlined in Section 2.1. This allows PI functions to be written at the endpoints 0 and 20 without overflowing the universe. Note that soft limits on the x-z domain could be expanded by manually editing the universe to include a larger range. Either dimension can be restricted within the [0,20] domain at runtime without changing the universe.

Memberships are collapsed with approximately equal weight to each possible label such that the tile will become a crisp member of whichever set it has the most membership in. The example in Figure 1 represents a single scan across row 0. In this case the tile in column 0 clearly has the most membership in the "red" set so it is assigned the "red" label. In the case of ties across memberships such as at column 3 an order of precedence



```
Linguistic Value: [ ??? ] OR [ ??? ] (r),  [ ??? ] OR [ ??? ] (y),  [ ??? ] OR [ ??? ] (b)

 1.00          r      y               b
 0.95         r r    y                bb
 0.90               y  y             b  b
 0.85
 0.80        r   r
 0.75                   y
 0.70                               b
 0.65              y                     b
 0.60       r     r
 0.55
 0.50                    y         b
 0.45             y                       b
 0.40
 0.35      r       r
 0.30                     y
 0.25                             b
 0.20            y                         b
 0.15     r         r
 0.10           y          y     b          b
 0.05    r           r      y
 0.00bbbbbbbbbbbbbbbbbbbbbbbbbbbbyyyyyyyyyyyybbbbbbbbbb
     |----|----|----|----|----|----|----|----|----|----|
   -5.00      1.00      7.00     13.00     19.00     25.00

Universe of Discourse:  From  -5.00  to   25.00

FuzzyCLIPS> |
```

Figure 1: Sample row membership functions

is followed where red is prioritized, then blue, then yellow. The tile at column 3 has equal membership in both the red and yellow sets and so is assigned a ”red” label. Likewise the tile at column 9 is not a member of any of the three sets and so is assigned the ”red” label.

Under the scanning algorithm fuzzy sets are defined along rows and columns of the grid, but not radially. Additional rules are defined to inform tiles diagonally adjacent to key tiles about labelling by visiting a tile on the edge of a key tile and then scaling down its set memberships. The degree to which the set is scaled is an arbitrary value that could be altered as a hyperparameter to tune the engine. The current engine allows radial information to propagate only by a distance of one tile; future work could define additional rules with farther-reaching propagation relying on similar logic.

2.4 Assigning Confidence to Labels

One disadvantage of ”scanning” through rows and columns is that a tile is likely to be revisited more than once. As a result it is necessary to inform the application as to whether new information should cause a tile label to change. This is represented with calculated certainty factors, where labels assigned to tiles far away from key tiles are typically less certain than labels assigned to tiles close to key tiles.

$$CF_{red} = RedMembership \times (1 - max\{BlueMembership, YellowMembership\}) \quad (1)$$

Equation 1 shows the certainty factor formula for a tile that is assigned the "red" label during scanning, which takes into account 1) the degree to which the tile is a member in the red set and 2) the degree to which the tile is more of a member in the red set than in the blue or yellow sets. The formula is relatively simple and provides the following desired behavior:

1. A tile that is a crisp member of the red set and not a member of the blue or yellow sets will have a certainty factor of 1.0;
2. A tile that is extremely a member of the red set will have a higher certainty factor than that of a tile that is only somewhat a member of the red set;
3. A tile that is a member of at least two sets will have a lower certainty than a tile that is only a member of the red set;
4. A tile that is a member of no sets will have a certainty factor of 0.0.

Where a tile is selected to be labelled as yellow or blue a similar formula to Equation 1 is used to calculate certainty in the respective label. Under this system a tile can be relabelled if and only if the new label is written with greater certainty than the previous label. The system guarantees that each tile will be represented by the label with the highest certainty at the conclusion of scanning.

2.5 I/O

To facilitate operations in the context of the Project Corridor VR simulation initial and closing file I/O rules were defined. File input reads in parameters and key tiles from a source file path passed in as part of the initial fact database, while file output writes out labels for each tile to an output file path at the conclusion of labelling operations.

3 Results

3.1 FuzzyCLIPS Engine Results

The procedural labelling engine is implemented in the FuzzyCLIPS language according to design specifications. Sample file input to the engine is as follows, where each of the x and z domains are defined as [0,4) and two opposite corners are populated with yellow and blue tiles. Each key tile line is written in the order `x(column) z(row) color CF`.

```
0 4 0 4
0 0 yellow 1.0
3 2 red 1.0
3 3 blue 0.7
```

During scanning the program overlays membership functions for each row and column as shown previously in Figure 1. The following output file is written at the conclusion of operations:

```
3,0,red
3,1,red
0,1,yellow
0,2,yellow
0,3,yellow
1,3,blue
2,3,blue
2,2,red
1,2,red
2,1,red
1,0,yellow
2,0,yellow
1,1,yellow
3,3,blue
3,2,red
0,0,yellow
```

3.2 Unity3D Project Integration

Helper scripts were written to interface the Unity3D C# project with the text-based map files. Figure 2 shows the above text data rendered out into tile placements in Unity3D. Some key details to note:

1. The red and blue key tiles are adjacent so the blue region is more limited than the yellow region.
2. The blue and yellow key tiles are placed at opposite corners, but the yellow tile is placed with an initial confidence greater than the blue tile so the yellow label propagates more strongly to the rightmost corner.
3. The yellow and red key tiles are placed with equal confidence, but the red tile is closer to the leftmost corner than the yellow tile so the red label propagates more strongly to the leftmost corner.

3.3 Relabelling in Unity3D

Each of the "red", "yellow", and "blue" labels are defined with Unity scripts as regions that can be remapped onto different types of tile variants. Several 10×10 text maps are generated with the FuzzyCLIPS engine and saved in the VR project. At runtime different tile types are mapped onto the regions so that the "boss room" can be designed to better integrate with the project. Figures 3 and 4 show the result of remapping tiles, while other examples of maps generated by the engine are included in the Appendix (Section 5.1). Each map adheres to specifications outlined in Section 2.1 and exhibits sufficient coherency to be useful in the Project Corridor initiative.

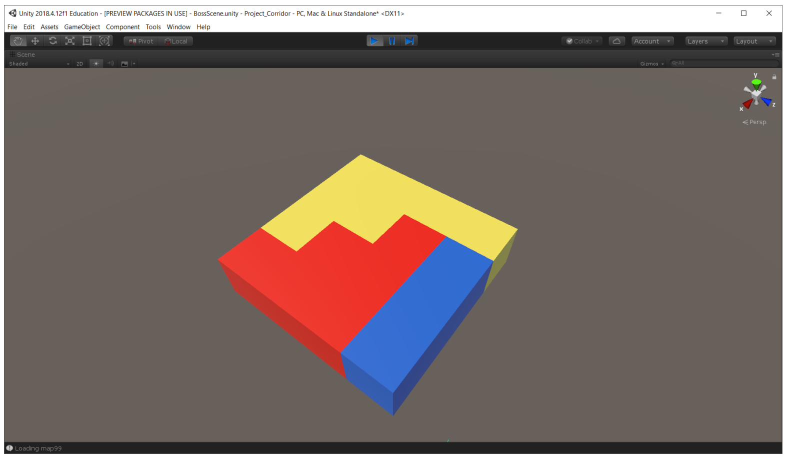

Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Persp
Loading map99

Figure 2: Example Color Map

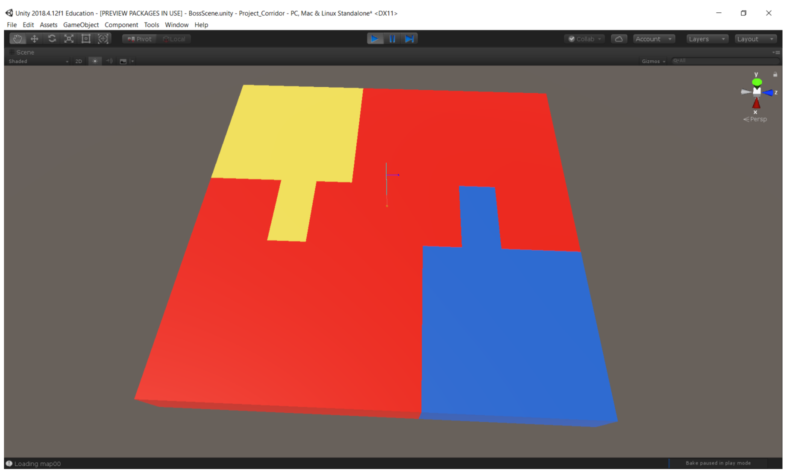

Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Persp
Loading map00

Figure 3: 10 × 10 Color Map

4 Summary of Effort

The deliverable outlined in the project proposal is successfully yielded. A FuzzyCLIPS engine is developed that appropriately uses fuzzy membership sets and certainty factors to populate and return a crisp $m \times n$ labelled grid representing a fully traversable map.

Each of the four goals presented in the project proposal is met and outlined in the Discussion section. During work on the project the two most difficult goals to meet were designing local uncertainty propagation rules for fuzzy logic and designing rules to convert fuzzy tile data to crisp tile labels. Two significant barriers to implementing these rules were overcome as follows:

1. While FuzzyCLIPS typically executes fuzzy sets along a single dimension the grid labelling problem required that two dimensions be considered in the membership

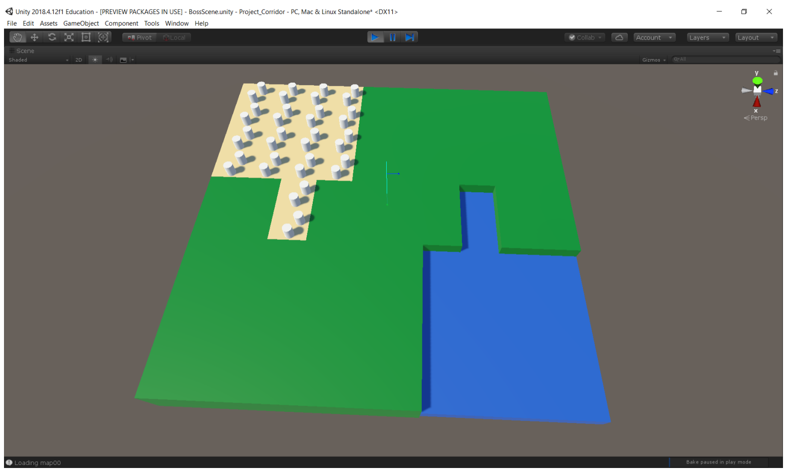

Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Scene
Shaded
Persp
Loading map00

Figure 4: 10 × 10 Color Map, Retiled

function in order to map tiles onto one-dimensional labels. The problem was overcome with iterative scanning through the grid, which produces sufficient but nonoptimal results requiring each tile to be considered under at least one row and one column iteration. An ideal solution would extend FuzzyCLIPS such that membership in a fuzzy set may be defined as a member of multiple variables, eliminating the need for traversal over rows and columns and simplifying the methods needed to carry out radial updates to tiles.

2. FuzzyCLIPS typically expects membership functions to be defined statically in advance of runtime, but problem specifications required that the user be able to influence membership functions at runtime in the form of inputting key tiles. A solution was implemented to dynamically clear and update membership functions for each of the three labels through the use of fuzzy unions; however, this precluded the use of known identifiers and resulted in greater complexity of implementation than static identifiers typically require. Future work could extend FuzzyCLIPS to more easily allow dynamic definition and modification of fuzzy membership functions in the form of rules definitions at runtime.

Future work could also allow other expansions and optimizations:

- Several hyperparameters were introduced during development including sensitivity factors for key tile PI functions and for radial labelling. These hyperparameters could be accepted as user input to tune the performance and map outputs of the FuzzyCLIPS engine.
- While three colors were used in the project, the number of colors could be extended to any arbitrary number of labels.

The project deliverable will be actively utilized to generate maps and assorted tile grids for the Project Corridor VR development project.

References

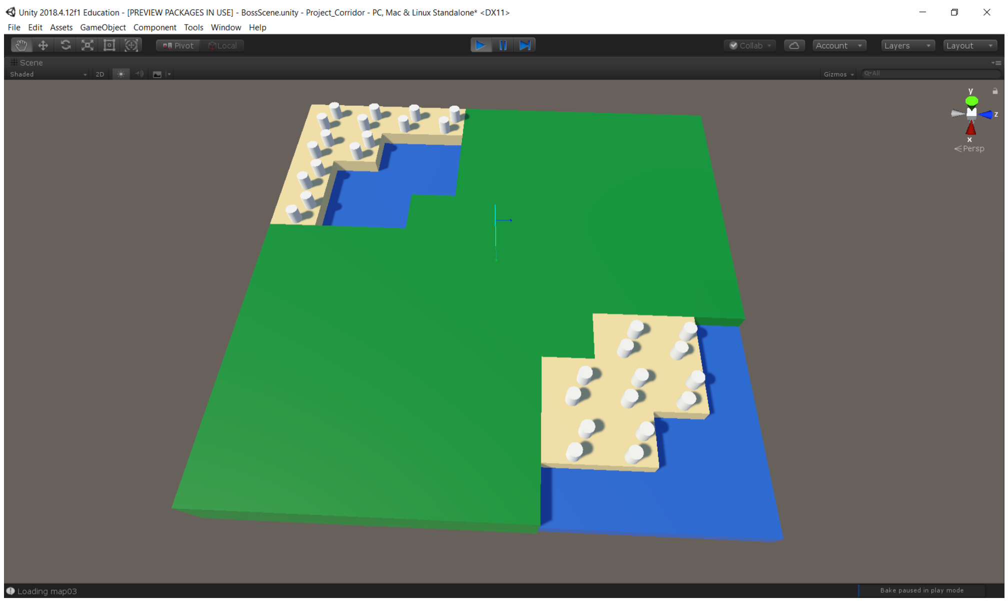
[1] I. Dykeman, “Procedural worlds from simple tiles.” `https://ijdykeman.github.io/ml/2017/10/12/wang-tile-procedural-generation.html`, October 2017. Online; accessed 30 October 2019.

[2] G. D. Riley, “Fuzzyclips 6.31 distribution.” `https://github.com/garydriley/FuzzyCLIPS631/tree/master/source`. Code repository; accessed 22 November 2019.

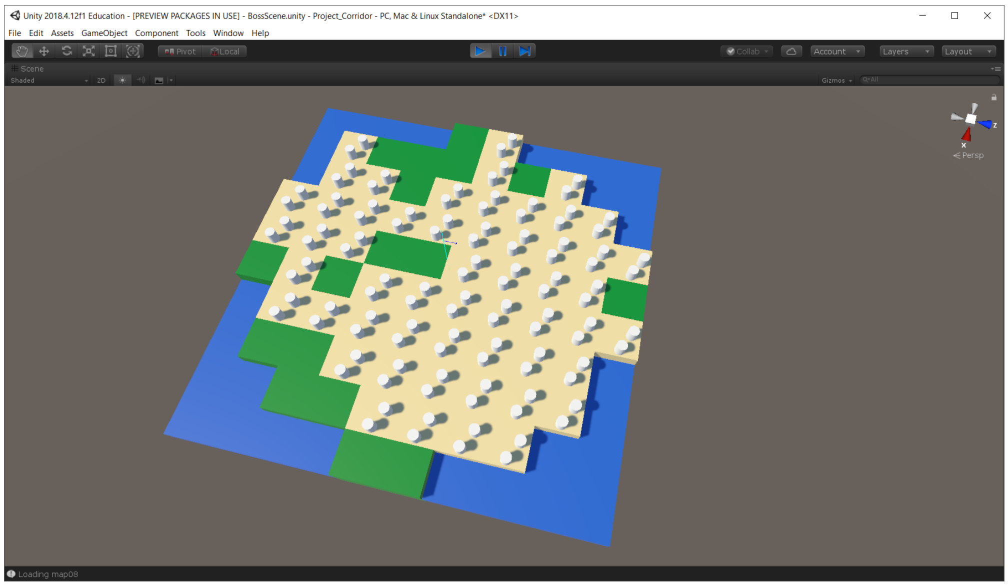
[3] B. Orchard, “Fuzzyclips version 6.10d user’s guide,” October 2004.

5 Appendix

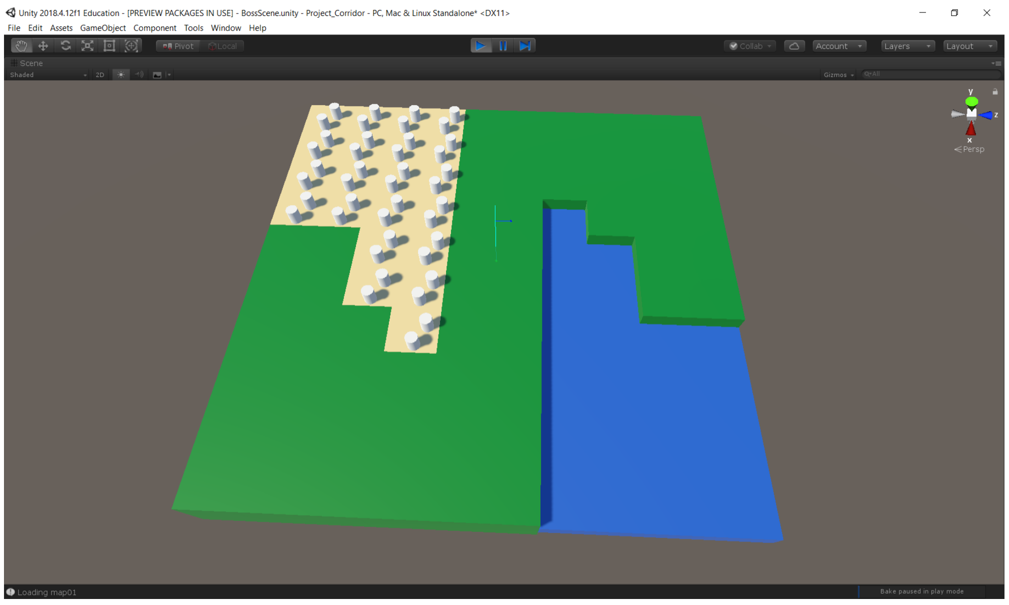
5.1 Maps from Procedural Labelling

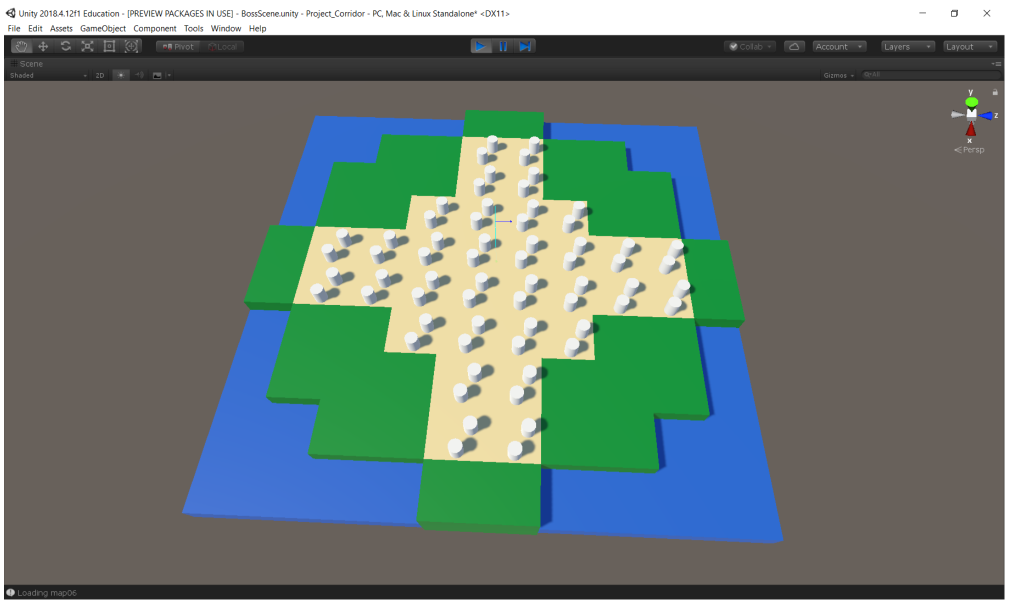

Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Pivot
Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
All
Persp
Bake paused in play mode

Loading map03


Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Pivot
Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
All
Persp

Loading map08


Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Pivot
Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
Persp
Loading map01
Bake paused in play mode


Unity 2018.4.12f1 Education - [PREVIEW PACKAGES IN USE] - BossScene.unity - Project_Corridor - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tools Window Help
Pivot
Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
Persp
Loading map06