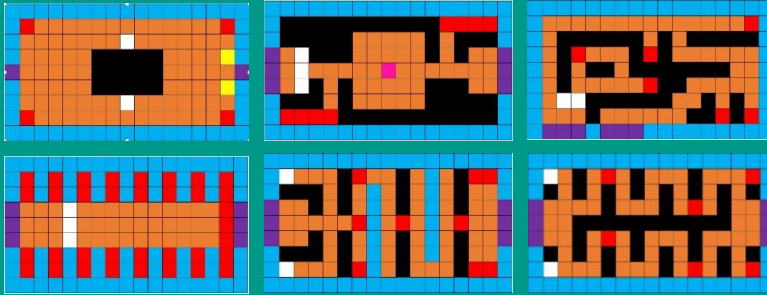


From Design to Unreal

When designing Nebula Knights' modular levels, many maps were exchanged between the team denoting different rooms colour coded to show specific tile properties. Measuring 9 x 17 tiles per room required placing over 150 tiles to see a single room in engine, and with over 50 room designs; building them, if only to visualise the design, became a daunting task. So I wrote a program to do it for us.



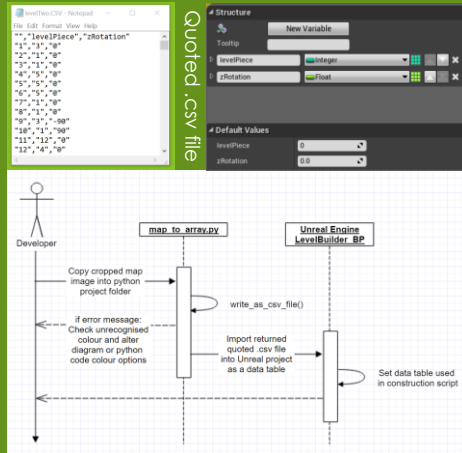
The python program (top right) uses the PyGame library to create a singleton of a room map image, and the csv library to return a .csv file of quoted integer values each corresponding to a tile colour defined as a modular level piece within a key. Spoofing an initial line denoting the column headers results in a file that Unreal Engine can import as a data table to be interpreted and used in blueprints. Unreal Engine can only import data in the form of quoted .csv files shown in the format below, so the python program is essentially an adapter for these incompatible data types (maps) and interpreter (Unreal Engine), so high coupling can be excused.

```

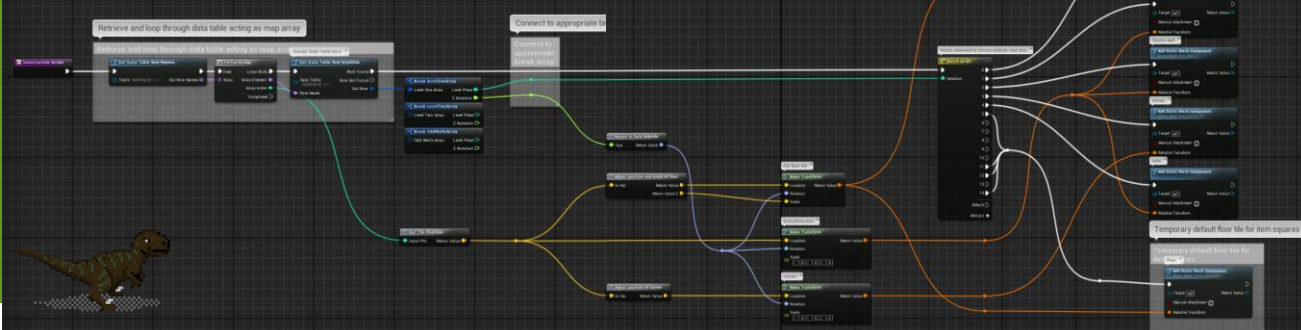
1  Colour code
2
3  (337, 125, 48) = Floor, orange 0.
4  ( 1, 176, 243) = Wall, black 1.
5  ( 8,  0,  8) = Stone, black 4.
6  (151, 40, 140) = Stone, purple 9.
7
8  (256, 5, 8) = Empty space, red 11.
9  (255, 255, 255) = Wall, white 12.
10 (220, 220,  1) = Dean element, yellow 14.
11
12
13 def colour_to_index(colour):
14     col = colour[0]
15     row = colour[1]
16     index = colour[2]
17
18     if col < 160:
19         if row < 160:
20             if index < 160:
21                 return 0
22             elif index < 160:
23                 return 1
24             elif index < 160:
25                 return 2
26             elif index < 160:
27                 return 3
28             elif index < 160:
29                 return 4
30             elif index < 160:
31                 return 5
32             elif index < 160:
33                 return 6
34             elif index < 160:
35                 return 7
36             elif index < 160:
37                 return 8
38             elif index < 160:
39                 return 9
40             elif index < 160:
41                 return 10
42             elif index < 160:
43                 return 11
44             elif index < 160:
45                 return 12
46             elif index < 160:
47                 return 13
48             elif index < 160:
49                 return 14
50             elif index < 160:
51                 return 15
52             elif index < 160:
53                 return 16
54             elif index < 160:
55                 return 17
56             elif index < 160:
57                 return 18
58             elif index < 160:
59                 return 19
60             elif index < 160:
61                 return 20
62             elif index < 160:
63                 return 21
64             elif index < 160:
65                 return 22
66             elif index < 160:
67                 return 23
68             elif index < 160:
69                 return 24
70             elif index < 160:
71                 return 25
72             elif index < 160:
73                 return 26
74             elif index < 160:
75                 return 27
76             elif index < 160:
77                 return 28
78             elif index < 160:
79                 return 29
80             elif index < 160:
81                 return 30
82             elif index < 160:
83                 return 31
84             elif index < 160:
85                 return 32
86             elif index < 160:
87                 return 33
88             elif index < 160:
89                 return 34
90             elif index < 160:
91                 return 35
92             elif index < 160:
93                 return 36
94             elif index < 160:
95                 return 37
96             elif index < 160:
97                 return 38
98             elif index < 160:
99                 return 39
100            elif index < 160:
101                return 40
102            elif index < 160:
103                return 41
104            elif index < 160:
105                return 42
106            elif index < 160:
107                return 43
108            elif index < 160:
109                return 44
110            elif index < 160:
111                return 45
112            elif index < 160:
113                return 46
114            elif index < 160:
115                return 47
116            elif index < 160:
117                return 48
118            elif index < 160:
119                return 49
120            elif index < 160:
121                return 50
122            elif index < 160:
123                return 51
124            elif index < 160:
125                return 52
126            elif index < 160:
127                return 53
128            elif index < 160:
129                return 54
130            elif index < 160:
131                return 55
132            elif index < 160:
133                return 56
134            elif index < 160:
135                return 57
136            elif index < 160:
137                return 58
138            elif index < 160:
139                return 59
140            elif index < 160:
141                return 60
142            elif index < 160:
143                return 61
144            elif index < 160:
145                return 62
146            elif index < 160:
147                return 63
148            elif index < 160:
149                return 64
150            elif index < 160:
151                return 65
152            elif index < 160:
153                return 66
154            elif index < 160:
155                return 67
156            elif index < 160:
157                return 68
158            elif index < 160:
159                return 69
160            elif index < 160:
161                return 70
162            elif index < 160:
163                return 71
164            elif index < 160:
165                return 72
166            elif index < 160:
167                return 73
168            elif index < 160:
169                return 74
170            elif index < 160:
171                return 75
172            elif index < 160:
173                return 76
174            elif index < 160:
175                return 77
176            elif index < 160:
177                return 78
178            elif index < 160:
179                return 79
180            elif index < 160:
181                return 80
182            elif index < 160:
183                return 81
184            elif index < 160:
185                return 82
186            elif index < 160:
187                return 83
188            elif index < 160:
189                return 84
190            elif index < 160:
191                return 85
192            elif index < 160:
193                return 86
194            elif index < 160:
195                return 87
196            elif index < 160:
197                return 88
198            elif index < 160:
199                return 89
200            elif index < 160:
201                return 90
202            elif index < 160:
203                return 91
204            elif index < 160:
205                return 92
206            elif index < 160:
207                return 93
208            elif index < 160:
209                return 94
210            elif index < 160:
211                return 95
212            elif index < 160:
213                return 96
214            elif index < 160:
215                return 97
216            elif index < 160:
217                return 98
218            elif index < 160:
219                return 99
220            elif index < 160:
221                return 100
222            elif index < 160:
223                return 101
224            elif index < 160:
225                return 102
226            elif index < 160:
227                return 103
228            elif index < 160:
229                return 104
230            elif index < 160:
231                return 105
232            elif index < 160:
233                return 106
234            elif index < 160:
235                return 107
236            elif index < 160:
237                return 108
238            elif index < 160:
239                return 109
240            elif index < 160:
241                return 110
242            elif index < 160:
243                return 111
244            elif index < 160:
245                return 112
246            elif index < 160:
247                return 113
248            elif index < 160:
249                return 114
250            elif index < 160:
251                return 115
252            elif index < 160:
253                return 116
254            elif index < 160:
255                return 117
256            elif index < 160:
257                return 118
258            elif index < 160:
259                return 119
260            elif index < 160:
261                return 120
262            elif index < 160:
263                return 121
264            elif index < 160:
265                return 122
266            elif index < 160:
267                return 123
268            elif index < 160:
269                return 124
270            elif index < 160:
271                return 125
272            elif index < 160:
273                return 126
274            elif index < 160:
275                return 127
276            elif index < 160:
277                return 128
278            elif index < 160:
279                return 129
280            elif index < 160:
281                return 130
282            elif index < 160:
283                return 131
284            elif index < 160:
285                return 132
286            elif index < 160:
287                return 133
288            elif index < 160:
289                return 134
290            elif index < 160:
291                return 135
292            elif index < 160:
293                return 136
294            elif index < 160:
295                return 137
296            elif index < 160:
297                return 138
298            elif index < 160:
299                return 139
300            elif index < 160:
301                return 140
302            elif index < 160:
303                return 141
304            elif index < 160:
305                return 142
306            elif index < 160:
307                return 143
308            elif index < 160:
309                return 144
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Colour tolerances checking script



Finally, the blueprint below acts as a factory, looping through whichever data table's rows it is supplied; using the index to calculate the position of the tile, the modular level piece, and any transform needed. In this case, a rotation around the z-axis so that all walls will face inwards.

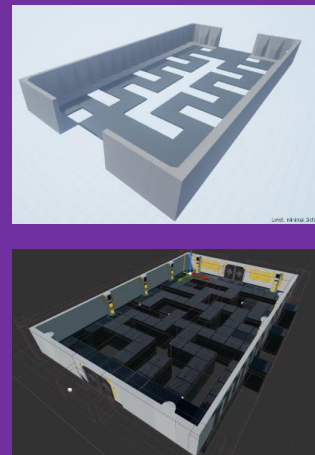


Python Map Reader

```

1  import sys, csv, pygame
2  from pygame.locals import *
3
4  pygame.init()
5
6  # ...
7
8  def colour_to_index(colour):
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



The result is a serviceable mock up of the room (top). Whilst it lacks the personal touch of a hand-crafted designers level (below), it can very quickly help to visualise any of the designer's level maps in engine. Even were they crudely drawn on paper, this program could identify the colours of each tile, and parse the data into a quoted .csv file, which the blueprint could build a room from.

Richard
Steele
Nebula Knights -
Boskiss Studios

