Bellmorn (any resemblance is purely coincidental)

Notes – Gordon Monro

These notes are from the point of view of the artist. In “A day in the life” and “News from Bellmorn” I have given various views from the imaginary people who inhabited the city or otherwise had to do with it.

A real city is shaped partly by the decisions of the many individuals and businesses in the city and partly by larger-scale decisions such as the zoning of districts or the construction of a public transport network. In Bellmorn, as far as possible the city is shaped entirely by decisions taken at an individual or very local level. I wanted to see how a complex thing such as a city would arise from purely local decisions.

The simulation of the imaginary city of Bellmorn is really concerned with only one thing: the mid-week morning commute to work, in a completely individualistic city where all decisions are made at the individual level. The simulation is carried out by a computer program I wrote; the program generates all the maps and 3D views of the city, including the frames for the time-lapse video.

An unrealistic simulation

In many ways the simulation is unrealistic. This is signalled at the outset by the hexagonal geometry of the imaginary city and the very simplistic representation of buildings. I chose hexagons partly because I have worked with them before when making abstract art, and partly because I wanted to make it clear that Bellmorn was not modelled on any real city.

As noted, the simulation of the city of Bellmorn is really concerned with only the morning commute to work. This focus on commuting means that the city has no schools, no hospitals, no parks. All it has is buildings, roads and commuters, with a small exception in that there is some freight traffic as well. The freight traffic connects industrial buildings to service workplaces; it has the effect of making some longer-distance connections between small communities near the start of the simulation.

I did have an intention to include a mechanism for creating parks, but that would require a local council or similar body doing something like offering a developer increased height limits if the developer provides a park. In a city driven by individual decisions there is no local council and no basis for such trade-offs to occur.

The real Melbourne is sprawling apparently without limit, but my program was not designed for that, and so the growth of Bellmorn takes place within a circumscribed hexagonal area.

Bellmorn is essentially an abstract work, and I was unable to set a consistent scale in the simulation, either in space or in time. I did attempt to have one person-entity in the simulation represent a fixed number of “real” people so as to give an estimated population for each stage of growth of the city, but that also proved unworkable.

The graphics and display

The geometry of the city is based on small hexagons (abbreviated to **hexes**). There are 2,791 hexes on the map, with six blocks per hex, so there are potentially 16,746 buildings. Fewer than this will actually be built, as motorways and motorway junctions block buildings out from substantial areas of the map.

The 2D graphics (leaving aside the use of hexagons) are more or less based on Apple Maps and Google Maps, and to some extent recent street directories. I did also look at historical maps, but they didn't prove useful.

The 3D images together with the text displays ("A day in the life" panels and "News from Bellmorn" press cuttings) are intended to give an ambiguous picture. The graphics look toy-like, and placing the whole city on a podium makes it look like a model of a city. But the text displays indicate that the city is alive: people are living here.

The podium also solves another problem: my program has a fixed area for the city. To show this city sitting on an infinite ground plane would emphasise the artificiality of this restriction on area, and the podium side-steps this problem. Having the city floating in space with a black background was inspired by an artwork I have seen that had a landscape similarly floating in space. The simplified buildings were inspired by toy houses used as Christmas decorations, and also an installation that I saw with model buildings a couple of metres tall. I was happy for the graphics to be somewhat toy-like. I needed a style that would work for both single-storey houses and large office towers, and that would cope with the odd-shaped blocks arising from the hexagonal geometry.

The prints show five stages in the development of Bellmorn:

- Early days
- Growing fast
- Boom town
- Major city
- Gridlock

The time-lapse video covers this entire progression. There is very little change towards the end of the video, as the city eventually stagnates.

Jobs and locations

New jobs and people constantly arrive, with budgets for workplaces (employers) and housing (workers). People also leave, so there is a constant turnover. Developers will buy and knock down an existing building if they can make a sufficient profit by building something else more in demand.

No distinction is made between buying and renting in the program; the budget can be considered as rent or as mortgage payments. If the business owner cannot find an affordable workplace location, the job fails; if the worker cannot find an affordable place to live, the job fails. If the job fails, no-one moves in to Bellmorn. This doesn't affect the rate of people leaving.

The budgets are only considered once, at the time when the job is created and the worker finds a home; the simulation does not attempt to track the ongoing economic fortunes of individuals or businesses.

Jobs

The simulation includes four broad categories of jobs:

- Knowledge workers (professionals or managers)
- Clerical/office workers (generally less skilled, and lower paid)
- Service workers (mostly retail and hospitality)
- Industrial workers (businesses includes factories, warehousing and logistics, transport)

In general knowledge workers are the best paid (so can afford more for housing) and service workers the lowest paid, with the other two categories in the middle, but there is very considerable random variation.

When a new job arrives, a random starting point is picked on the map, and the business owner looks for the best location they can afford within a certain travel time from the random starting point. At first, with few roads, only a short distance can be covered in the travel time and small settlements occur, but with better roads businesses are freer to move from their random starting points, and larger centres form.

Initially the jobs are weighted towards the lower-laid categories, but the proportions change with time. Before Boom Town stage the final proportions are reached, with a substantial fraction of knowledge workers.

Building types

The simulation includes four broad categories of buildings:

- Residential
- Office
- Service
- Industrial

The Office category provides accommodation for both knowledge worker jobs and clerical/office jobs.

Most buildings are of one type only (e.g. all residential or all service businesses). Multi-storey residential and office buildings may have one or two storeys at their base occupied by service businesses.

The height of a building is generally determined by the best profit for the developer (as discussed later); however, industrial buildings have a height limit of six storeys.

The simulation assigns at one most occupant (worker in a workplace, or resident in a house or flat) per storey of a building. A building storey can be empty, in which case it is available for a new occupant.

Location preferences

Both businesses and workers have preferences, which influence the prices of buildings, as discussed under “Prices” below. A fairly large amount of self-segregation occurs, with buildings of the same type tending to cluster together, despite the complete lack of zoning restrictions.

The “Day in the life” displays show jobs such as “warehouse hand” and “assistant food technologist”. These descriptions were supplied by the artist; the simulation program only deals with the four broad job categories above. However, comments about the expense of housing and the length of the worker’s commute come directly from information generated by the program. The program doesn’t supply any reason when people leave; reasons were supplied by the artist.

Roads

At the beginning the only roads are rough tracks, which are not shown on the map. As soon as there is any significant traffic, the track is upgraded to a minor road. The successive road types are:

- track — rough dirt track, very slow to travel on
- minor road — paved, but narrow and of low traffic capacity
- district road — somewhat higher traffic capacity; in real life would be equipped with traffic lights, turning lanes
- major road — multiple lanes, high traffic capacity
- motorway — the highest traffic capacity, has special rules

The road engineers respond entirely to demand, in the form of number of vehicles using a section of road. They adhere strictly to the "desire lines" principle, which in common usage refers to the situation where pedestrians who use a park or other open ground will create their own paths if the official ones are inconvenient or non-existent. The road engineers interpret the principle to mean that they do not try to predict where heavy traffic may occur, but will wait until it happens and then try to ameliorate it: the engineers happily upgrade any road that has enough traffic.

A road section is never downgraded, even if the traffic on it falls to nothing.

In the simulation, there are two reasons why a section of road is slow to traverse. One is that the number of vehicles using the road is high. The other reason is a high number of residents or workers nearby: the rationale here is that traffic through a busy area is likely to be slowed down by pedestrians, delivery vehicles, cars coming out of parking places and so on. The road engineers only take account of the number of vehicles using the road, so a section of road may be slow to traverse, but not a candidate for upgrading.

Motorways

(I thought of these as freeways, but in a free enterprise city they won't be free.)

If a section of major road carries enough traffic it becomes a candidate for upgrading to a motorway. A motorway cannot have abrupt bends, and should not have too many access points. And I ruled out having parallel motorways very close to one another. As a consequence of these rules, there can be very busy and congested major roads that cannot be upgraded to motorways. Further, the restriction on the number of access points means that existing roads can be closed when motorways are built, leading to "roads to nowhere". However, the program does check that closing roads doesn't isolate part of the map.

The only allowable motorway junctions are Y-shaped junctions, where three motorways meet. Such a junction takes up a large area (seven hexes). The rules allow for motorway junctions to form next to one another, and complex tangles can form.

Since motorways are constructed piecemeal in response to demand, illogical configurations can very easily form.

When a motorway section or motorway junction is built, all buildings on the affected hexes are demolished and the occupants are evicted.

Freight traffic

There is a certain amount of freight traffic: each service business is required to have an industrial business as a supplier. This is not intended as a realistic simulation of freight traffic, but it does have the effect of causing some longer-distance roads to occur early in the simulation.

Finding commuting routes

A commuter naturally tries to find the quickest route. This is done in the program (*nerd alert!)* by what is called the A* (pronounced "A-star") algorithm, used for pathfinding in computer games. From time to time, and in particular when a new motorway section is built, commuters re-assess their journeys to work. From one point of view, the whole artwork is a riff on the A* algorithm.

Prices and development

Setting prices for buildings turned out to be hard. Initially I wanted to have some sort of auction scheme, in keeping with the individualistic, free-enterprise theme. However, an auction system requires a large pool of buyers and sellers, and I could not come up with anything workable. In the real world, prices in Ballarat (for example) are influenced by prices in Melbourne and in other cities across Victoria and the rest of Australia. Additionally, an auction scheme implies that purchasers have an order of preference among the available properties, so I would need to program such an order. From these considerations I ended up calculating prices according to formulas, discussed in what follows. But I allowed for an element of supply and demand, introducing a price moderation process.

There is no general inflation: the ranges of the budgets do not change over time, and nor do the costs of constructing a building. As the city grows, there will be a tendency for prices to increase because of more access to services and so forth, but this is not due to inflation.

"Naïve" formulas for prices

The program contains a formula for the price of each of the four kinds of building. The base price for any building is 10 units per storey; this is modified by factors such as the amount of activity (workers and residents) in the neighbourhood, proximity to motorways, and so on. These factors can be positive (increase the price) or negative (decrease the price). I have called the formulas "naïve" because I made them up according to my intuition.

For non-industrial businesses, being near industrial buildings is a negative. Being near a main road is a negative for office businesses and a positive for service and industrial businesses. Being near a motorway is a negative for non-industrial businesses and a positive for industrial businesses.

For non-industrial businesses, a larger amount of activity (number of workers and residents) in a reasonable travel time is a positive. For any of the four types of job, having other workers of the same type nearby is a positive factor.

An office can accommodate either a knowledge worker or a clerical/office worker. Its price is a weighted average of the prices for those two job types, weighted according to the proportions of new jobs of those types.

People looking for houses or flats avoid industrial businesses and busy roads, though they like service businesses fairly close by. However, the chance to get a large block of land on the city fringe is also attractive (which means that the first house in an otherwise empty hex attracts a premium).

Workers care about commuting time rather than the length of the commute in kilometres; a worker will put up with an arbitrarily long commute to get somewhere to live. However, workers are reluctant to take on a longer commute than they have to, though they may accept a longer commute if they can get a better house, of course always limited by their budget. A worker who lives very near their workplace can walk to work, and can afford to pay a little more for housing, since they don't have the commuting costs.

Moderation of prices

A business owner looking for a workplace or a worker looking for somewhere to live is only concerned with one kind of building, but a developer compares the different types of building to see what sort will give the best return on investment. I didn't expect that the naïve formulas would work well when comparing different building types, and I also wanted prices to respond to supply and demand.

I therefore introduced a price moderation process. I set an arbitrary goal of 10% vacancies in each building type: if the vacancies are higher the price of that type of building comes down; if the

vacancies are lower the price goes up. This generally worked to encourage developers to build needed types of buildings. Constructing a workable price moderation process proved to be one of the most difficult parts of creating *Bellmorn*.

Development

It may be that the best location that a new arrival (employer or worker) finds is an empty block. In that case it is imagined that the worker (in case of a residence) or the employer (for a workplace) commissions a new building of the appropriate type. I think of this as “owner-building”. Generally, the buildings are one-story, but a service building will be two storeys, with the service premises below and a flat occupying the upper storey (as seen in many older shopping strips in real life).

All development apart from owner-building is conceived to be carried out by developers. From time to time an existing building will be selected for possible redevelopment. The developer performs a calculation for each type of building:

- The cost of buying the existing building is calculated. This uses the moderated price for the building.
- The cost of building a specific number of storeys is calculated. Taller buildings cost more per storey.
- For each possible height (number of storeys) the sale price of the building is calculated. Industrial buildings have a height limit of six storeys. Other types of building effectively have no limit beyond the economic constraint in the next point.
- The developer calculates the net profit for each building type and each number of storeys, and selects the type and height that yields the greatest **return on investment** (net profit divided by total costs). If the greatest return on investment available is under 5% the developer does not proceed.

If a new building is built, the old building is torn down and the occupants evicted. The new building is initially empty.

Buildings with 40 to 50 storeys do occur in the simulation. Taller buildings are theoretically possible, but I didn’t see them in practice.

Suburbs

Suburbs play a minor role in the simulation. A suburb has a name and in the real world some sense of community or identity. In the simulation, a suburb has a centre hex, and buildings near that centre hex have a slight price premium, thus making them more attractive.

At the start, there is one suburb in the centre of the map, Bellmorn Central, thus giving the centre of the map a slight advantage. The rest of the map is open for settlement, but is considered “unincorporated”. As the simulation proceeds, an area with sufficient activity that is not already part of a suburb may be declared to be a suburb. Suburbs can also grow into adjacent unincorporated areas.

The name of a new suburb is chosen partly at random from a long list I constructed. The names have personal meaning: places I have been, a few people I knew (now all dead), mathematicians, scientists and artists whose work I have encountered (at school or later), a few references to personal incidents.

Suburbs never shrink and never amalgamate.

The main purpose of suburbs is to allow the comments in “A day in the life” to refer to specific areas on the map.

Comments on the growth of Bellmorn

At the start of the simulation the centre of the map has a slight advantage, corresponding to the one initially existing suburb, Bellmorn Central. In some runs of the program the growth did start more or less in the centre and spread out, but in the run that I used to make the *Bellmorn* artwork, the growth at the start was distributed in small settlements, only one of which was at the centre.

There was considerably more self-segregation than I expected; the various preferences of bosses and workers have a powerful effect. However, some districts with a mixture of low-rise housing and low-rise industry persist.

Also, I expected to see more “mixed” buildings, with one or two service storeys at the base of an office or residential tower.

And the great concentration of high-rise office buildings was also a little surprising to me, though there is nothing to stop it: in this simulation, workers will commute as far as needed to get a house, and bosses don’t consider the travel times of their workers. However, high-rise buildings do cost more per storey to build, and I was a little surprised that buildings of 40 or 50 storeys proved profitable to construct.

A problem that I thought might occur was that, since development is very much local, isolated communities would develop, which would only connect to one another when they expanded enough to touch each other. The freight traffic mechanism mentioned above had the effect of creating some longer-distance roads early on: near the start a service business may find that its nearest available industrial supplier is a considerable distance away; the freight route found will immediately be upgraded to a minor road. This worked well, in that most areas of settlement were connected to each other early on. Some isolated settlements remained, considered to be connected by rough dirt tracks (which are not shown on the maps).

Once paved connecting roads existed, they facilitated ribbon development, and then the major centres started to join up. The final stage was filling in all the remaining vacant land.

As noted above, getting a functioning pricing regime was difficult. The one I settled on worked well as far as the Major City stage, but seems to have run into trouble after that, as by the Gridlock stage there was a shortage of housing, but the developers weren’t responding by replacing non-residential buildings with apartment blocks. A difficulty with investigating this was that a run of the program to Gridlock stage took nearly a week, and I ran out of weeks.

Possible extensions

In the “News from Bellmorn” commentary, the decay of the city is predicted. Causing this to happen would need some new mechanism, perhaps varying the rate of people wanting to come to the city according to average commuting times and the rate of growth of the city.

I considered having a new category of “cultural workers”, badly paid but making their neighbourhoods trendy, until rising costs of accommodation push them out — a familiar story.

It would require a major redesign of the program to accommodate Melbourne-type indefinite sprawl.

I haven’t proceeded with such extensions, for three reasons. Firstly, as it is the program took nearly a week to run to Gridlock stage, and changes such as the possible decay of the city would take numerous runs to get working properly, if my experience with prices is any guide. Secondly, the program is already close to the limit of the complexity that I can handle. Thirdly, it becomes increasingly hard to see how to convey the results in the framework of an exhibition or the like.

Nerd alert!

What follows is a listing of the settings file for the program *Bellmorn3*, which ran the simulation of the imaginary city of Bellmorn. The settings file contains most of the main settings for the program. When the program is started, it reads in this file.

The notation `//` indicates the start of a comment. The program ignores the `//` and everything that follows it to the end of the line. The comments are for humans (me), not for the computer.

Since I wrote this for my own use, the abbreviations and terminology are obscure.

“Jopac” stands, more or less, for “job package”. It consists of the job type (four broad categories only: knowledge worker, clerical/office worker, service worker, industrial worker), the location of the worker’s residence, the location of the workplace, and, for a service workplace, the location of the industrial business that supplies the workplace. The worker has a budget for accommodation; the business also has a budget for a workplace. The budgets vary with job type and have a probabilistic element.

The program tries to find both a workplace and a dwelling for the worker. If this fails, probably due to nothing within the budgets being available, the job is lost. If a dwelling and a workplace are found, the program finds the quickest route from the dwelling to the workplace (for super-nerds: using the A* algorithm), and calculates the commute time.

From time to time, people “leave town”, leaving vacant workplaces and dwellings available for new arrivals.

The pricing of workplaces and dwellings depends on various factors including proximity to industrial buildings (generally a negative), activity in the neighbourhood, and so on. There is also an element of supply and demand, in that if there is a high proportion of vacancies of one building type, a discount factor is applied to that type. Getting a reasonable pricing mechanism turned out to be the most difficult part of writing the program: I tended to get a massive over-supply of one type, or ridiculously high or low prices.

In the narratives in the exhibition I have assigned people jobs such as warehouse hand. The program did not generate these; it assigned only the four broad categories mentioned above: I assigned jobs within those categories “by hand”. The comments about which suburbs people live and work in, the numbers of storeys of buildings, budgets and commute times do derive from the output of the program.

(The settings file starts on the next page.)

```
// <><> File settings.bmn3.json <><> //
{
  // Arena size
  "jURingCnt": 30,
  "jVRingCnt": 30,

  // For placing jopacs
  "jLowerDropTimeLimit": 60.0, // later, try 60

  // Premiums - for a new suburb, and for tests
  "jPremiumRadius": 2, // try 2
  "jPremiumAmount": 2.0, // try 2.0

  // ----- ----- -----[1] Road parameters ----- ----- -----

  "jRoadParamsRaw_Track": [12.0, 1.0e+20, 1.0e+20, 1.0e+20, 1.0e+20, 1.0e-
5],
  "jRoadParamsRaw_Minor": [6.0, 2.0, 0.45, 10.0, 50.0, 4.0],
  "jRoadParamsRaw_District": [3.6, 10.0, 0.225, 20.0, 100.0, 20.0],
  "jRoadParamsRaw_Major": [2.0, 25.0, 0.175, 40.0, 150.0, 50.0],
  "jRoadParamsRaw_Freeway": [1.2, 150.0, 0.08, 1.0e+20, 1.0e+20, 1.0e+20],

  "jRoadParams_FreightWeighting": 3.5,
  "jRoadParams_ProbAccessStraight": 0.3,
  "jRoadParams_ProbAccessBent": 0.65,

  // ----- ----- ----- Sweeps ----- ----- -----

  "jActivityTravelTimeLimit": 45.0,

  // ----- ----- Prices and utilities of building units ----- -----

  "jDwell_ServiceWorkersIn19Fact": 0.1003,
  "jDwell_ActivityInTravelTimeFact": 0.00475,
  "jService_ServiceIn19Fact": 0.0626,
  "jService_ActivityInTravelTimeFact": 0.00402,
  "jIndustActivityFact": 0.0273,

  "jKnow_KnowWorkersFact": 0.00676,
  "jKnow_ActivityInTravelTimeFact": 0.00475,
  "jClerk_OfficeInHexFact": 0.0676,
  "jClerk_ActivityIn19Fact": 0.0205,

  "jPristineFact": 1.25,
  "jLongCommutePenalty": 0.35,

  "jWalkToWork_Radius": 2,
  "jWalkToWork_Prob": 1.0, //? This no longer applies, now hard-wired.
  "jWalkToWork_BudgetMult": 1.075, // can afford a better house if walking
to work

  // ----- ----- ----- Worker proportions mechanism ----- ----- -----
  // The order is know, clerk, service, indust

  // By experiment, 0.375 is about the lowest proportion of service
  // buildings at the start that gives a reasonable road network.
  "jWorkerProportions_Initial": [0.01, 0.09, 0.375, 0.525],
  "jWorkerProportions_Final": [0.2, 0.5, 0.15, 0.15],
  "jWorkerProportions_FadeIn": 20000,
```

[1]

```
// ----- ----- ----- Price mod mechanism ----- ----- -----

"jPriceMod_TargetVacancyProportion": 0.10,
"jPriceMod_IndivLowerFact": 0.1,
"jPriceMod_IndivUpperFact": 1.25,
"jPriceMod_GlobalLowerFact": 0.8,
"jPriceMod_GlobalUpperFact": 1.2,

// ----- ----- ----- Budget parameters ----- ----- -----

"jBudgetParamsRaw_KnowBu": [60, 0.5, 30.0, 100.0],
"jBudgetParamsRaw_KnowDwell": [50, 0.5, 25.0, 100.0],
"jBudgetParamsRaw_ClerkBu": [50, 0.5, 15.0, 100.0],
"jBudgetParamsRaw_ClerkDwell": [40, 0.5, 12.0, 100.0],
"jBudgetParamsRaw_ServiceBu": [50, 0.75, 12.0, 100.0],
"jBudgetParamsRaw_ServiceDwell": [40, 0.75, 12.0, 100.0],
"jBudgetParamsRaw_IndustBu": [50, 0.5, 12.0, 100.0],
"jBudgetParamsRaw_IndustDwell": [40, 0.5, 12.0, 100.0],

// ----- ----- ----- Suburbs ----- ----- -----

"jSuburbThreshold": 40.0,
"jSuburbIncrement" : 6.0,
"jSuburbSecIncFact": 2.5,


// ----- ----- ----- Suburb names ----- ----- -----

// Name order partly randomised in the program. More names than needed.

"jSuburbNames_Raw": [ "Bellmorn Central",

  // British places 1
  "Stoke Bishop", "Clifton Downs", "Blaise Castle", "Long Ashton",
  "Inverness", "Ben Wyvis", "Foulis Castle", "Cricklade",
  "Glen Affric", "Aberdeen", "Macandrew", "Mendip",

  // British surnames 1
  "Shepherdson", "Rowbottom", "Fredericks", "Kernighan",
  "Ritchie", "Carslaw", "Chadwick", "Woolley",
  "Madsen", "Manning", "Eyland", "Cleveland",
  "Spowers", "Balson", "Crowley", "Faraday",
  "Maxwell",

  // Aussie 1
  "Peters Hut", "Buttongrass", "Hobwee", "Old Billy",

  // First names, etc. 1
  "Davitina", "Domino", "Jemima",

  // European places 1
  "Neufeld", "Dilsberg", "Kochel", "Riga",
  "Trier", "Bremen", "Dagstuhl", "Oberwolfach",
  "Heidenhammer", "Kiruna", "Helsinki",

  // British real 2
  "Quantock", "Exmoor", "Beverley", "Millington",
  "Bainton", "Stamford Bridge", "Harrison", // (last one is a surname)

  // European surnames 1
  "Tarski", "Kripke", "Cauchy", "Dedekind",
```

```
    "Cavalieri", "Hilbert", "Peano", "Ackermann",
    "Xenakis", "Lavoisier", "Noether", "Markov",

    // Aussie 2
    "Slippery Run", "Broken Bridge", "Mossy Rock",

    // First names, etc. 2
    "Estelle", "Wilma", "Aaron",

    // British surnames 2
    "Maclane", "Rutherford", "Knuth",
    "Gleeson", "Boyd", "Tucker", "Francis",
    "Boden", "Priestley", "Hooke", "Cayley",
    "Coxeter", "Hester", "Donne", "Proctor",
    "Davy", "Galton",

    // European places 2
    "Sankt Michael",
    "Avignon", "Bienne", "Lucca", "Bologna",
    "Patras", "Prato", "Middelburg", "Naarden",
    "Pelion", "Prague",

    // European surnames 2
     "Vasilieff", "Brancusi", "Bergner", "Berzelius",
     "Mersenne", "Kupka", "Mendeleyev", "Frege",
     "Danti", "Aldrovandi", "Kippenberger", "Molnar",

     // British surnames 3
     "Boyle", "Flamsteed",
     "Jacobs", "Russell", "Vaughan", "Britten",
     "Walton", "Warren", "Douglas", "Rowell",
     "Goddard", "Morris", "Nake", "Thompson",
     "Reynolds", "Martin", "Conway",

    // Developer name
    "Windermere Lakes",
    ],

  // ----- ----- ----- Scheduling and checkpointing ----- ----- -----
  // -----  Schritt Clock settings -----

  // Values for 9 wheels
  // The order is:
  //    upgrade fwy, price mod, activity in trav time, set19, recalc trav
times,
  //    suburb, pop and vacancy report, consist check, (image gen),
checkpoint.

  "jStartTeeth": [10,   1,    1,     1,    1, 1, 10, 1, 20, 200],
  "jFinalTeeth": [10,  10,   20,    10,   20, 1, 10, 1, 20, 200],
  "jFadeInTimes": [1, 1000, 1000, 1000, 1000, 1, 1, 1,  1, 1],

  "jMaxNumCheckpoints": 1000,

  "jEonEarlyVal": 200,  // suggest 200
  "jEonLaterVal": 1000, // suggest 1000
  "jEonSwitchVal": 2000, // suggest 2000

  "jEpochMin": 200,
  "jEpochPopFrac": 0.025,
```

```
  // ----- ----- ----- 2D drawing, large ----- ----- -----

  // ----- Sizes -----

  "jL2D_ImHtPx": 9000,
  "jL2D_LeftMarginPx": 400.0,
  "jL2D_TopMarginPx": 400.0,
  "jL2D_TristepsPerSeg": 48.0,

  // ----- Settings for elements -----

  // Image as a whole

  "jL2D_Diagnost_ArenaClear_Raw": [0.0, 0.0, 1.0, 1.0],
  "jL2D_Display_ArenaClear_Raw": [0.25, 0.15, 1.0, 1.0], // was [0.25,
0.15, 1.0, 1.0]

  // Hex background

  "jL2D_BkgStd_EmptyHex_Raw": [0.0, 0.0, 1.0, 1.0],
  "jL2D_BkgStd_Builtup_Raw": [0.0, 0.0, 0.970, 1.0],
  "jL2D_BkgStd_FreewayAccess_Raw": [0.5, 0.2, 0.85, 1.0],
  "jL2D_BkgStd_FreewayNoAccess_Raw": [0.6, 0.2, 0.85, 1.0],

  // same as jDisplay_ArenaClear_Hsva
  "jL2D_BkgDisplay_EmptyHex_Raw": [0.25, 0.15, 1.0, 1.0], // was [0.25,
0.15, 1.0, 1.0]

  "jL2D_BkgDisplay_Builtup_Raw": [0.0, 0.0, 0.950, 1.0],

  // same as jL2D_BkgStd_FreewayNoAccess_Raw
  "jL2D_BkgDisplay_CannotBuild_Raw": [0.0, 0.0, 0.950, 1.0], // was [0.7,
0.05, 0.9, 1.0]

  // Blocks

  "jL2D_BlocksMaxInfo_Dwell_Raw": [0.4, 1.0, 1.0, 1.0],
  "jL2D_BlocksMaxInfo_Office_Raw": [0.6, 1.0, 1.0, 1.0],
  "jL2D_BlocksMaxInfo_Service_Raw": [0.8, 1.0, 1.0, 1.0],
  "jL2D_BlocksMaxInfo_Indust_Raw": [0.1, 1.0, 1.0, 1.0],

  "jL2D_BlocksDisplay_Body_Raw": [0.0, 0.0, 0.8, 1.0],
  "jL2D_BlocksDisplay_Outline_Raw": [0.0, 0.0, 0.0, 1.0],

  "jL2D_CappedHt": 50, // used for diagnostic drawing

  // Wash

  "jL2DWash_Office_Raw": [0.6, 1.0, 1.0, 0.2],
  "jL2DWash_Service_Raw": [0.8, 1.0, 1.0, 0.2],
  "jL2DWash_Indust_Raw": [0.1, 1.0, 1.0, 0.2],

  // Roads

  "jL2D_RoadWidths": [1, 1, 2, 3, 6, 1], // track, ...., closed

  "jL2D_EmptyRoad_Raw": [0.0, 0.0, 0.85, 1.0], // used for road with no
traffic

  "jL2D_MaxInfo_Congest_RawArr": [
    [0.389, 1.0, 0.9, 1.0], // grade 0 (green) = 140/360
    [0.125, 1.0, 1.0, 1.0],  // grade 1 (yellow-orange) = 45/360
```

```
    [0.000, 1.0, 1.0, 1.0]], // grade 2 (red)

  "jL2D_MaxInfo_Traffic_RawArr": [
    [0.389, 1.0, 0.9, 1.0], // grade 0 (green) = 140/360
    [0.111, 1.0, 1.0, 1.0],  // grade 1 (orangish-yellow) = 40/360
    [0.000, 1.0, 1.0, 1.0]], // grade 2 (red)

  "jL2D_Display_Congest_RawArr": [
    [0.389, 0.75, 0.75, 1.0], // grade 0 (green) = 140/360
    [0.111, 1.0, 1.0, 1.0],  // grade 1 (orangish-yellow) = 40/360
    [0.000, 1.0, 1.0, 1.0]], // grade 2 (red)

  "jL2D_Display_Greyscale_RawArr": [
    [0.0, 0.0, 0.9, 1.0], // track (though not drawn)
    [0.0, 0.0, 0.75, 1.0], // minor  - was 0.6
    [0.0, 0.0, 0.75, 1.0], // district - was 0.5
    [0.0, 0.0, 0.2, 1.0], // major
    [0.0, 0.0, 0.1, 1.0], // freeway
    [0.489, 1.0, 1.0, 1.0]], // closed (for tests, not production)


  "jL2D_MaxInfoCongest_BreakPts": [2.5, 5.0],

  "jL2D_ClosedRoad_Raw": [0.489, 1.0, 1.0, 1.0],
  "jL2D_CannotFreewayRoad_Raw": [0.9, 0.5, 1.0, 1.0],
  "jL2D_ProtectedRoad_Raw": [0.8, 0.9, 0.6, 1.0],


  // ----- ----- ----- 2D drawing, small ----- ----- -----

  // ----- Sizes -----

  "jS2D_ImHtPx": 3544, // 30 cm in px, rounded up
  "jS2D_LeftMarginPx": 400.0,
  "jS2D_TopMarginPx": 400.0,
  "jS2D_TristepsPerSeg": 48.0,

  // ----- Settings for elements -----

  // Image as a whole

  "jS2D_ArenaClear_Raw": [0.25, 0.15, 1.0, 1.0], // same as for large
display

  // Hex background

  "jS2D_Builtup_Raw": [0.6, 0.05, 0.950, 1.0],
  "jS2D_CannotBuild_Raw": [0.0, 0.0, 1.0, 1.0],

  // Roads

  "jS2D_RoadWidths": [1, 1, 2, 3, 6, 1], // track, ...., closed

  "jS2D_MajorRoad_Raw": [0.0, 0.0, 0.85, 1.0],
  "jS2D_Freeway_Raw": [0.0, 0.0, 0.75, 1.0],

  // Washes for suburbs - socio-ec

  "jS2D_GradedSocio_RawArr": [
  [0.0, 1.0, 1.0, 0.2],  // grade 0 - low socio-ec status
  [0.0625, 1.0, 1.0, 0.2],
  [0.125, 1.0, 1.0, 0.2],
```

```
  [0.1875, 1.0, 1.0, 0.2],
  [0.25, 1.0, 1.0, 0.2],
  [0.3125, 1.0, 1.0, 0.2],
  [0.375, 1.0, 1.0, 0.2],
  [0.4375, 1.0, 1.0, 0.2]], // grade 7

  // Washes for suburbs - commute time

  "jS2D_GradedCommute_RawArr": [
    [0.333, 1.0, 1.0, 0.2],  // grade 0 - low commute time
    [0.404, 1.0, 1.0, 0.2],
    [0.476, 1.0, 1.0, 0.2],
    [0.547, 1.0, 1.0, 0.2],
    [0.619, 1.0, 1.0, 0.2],
    [0.690, 1.0, 1.0, 0.2],
    [0.762, 1.0, 1.0, 0.2],
    [0.833, 1.0, 1.0, 0.2]], // grade 7 - very long commute time


  "jS2D_SuburbNoData_Raw": [0.0, 0.0, 0.0, 0.15],

  "jS2D_SuburbEdgeHex_Raw": [0.4, 1.0, 0.5, 0.1],

  "jS2D_SocioLowQ": 3.9, // a bit low - maybe 3.9?
  "jS2D_SocioHighQ": 5.4,  // was 5.4

  "jS2D_CommuteTimeLowQ": 30.0,
  "jS2D_CommuteTimeHighQ": 320.0,

  // Suburb boundaries

  "jS2D_SuburbBdry_Raw": [0.4, 1.0, 0.5, 1.0],

  // Route to work

  "jS2D_SlowestCommute_Raw": [0.95, 0.5, 1.0, 0.35], // slowest commute
  "jS2D_MaxSlowRoutesDrawn": 100,


  // ----- ----- ----- 3D drawing ----- ----- -----

  "jDummy_3D_Raw": [0.0, 1.0, 1.0, 1.0],

  "jRoadWidths3D": [1.0, 2.0, 3.0, 6.0],

  "jSky_3D_Raw": [0.0, 0.0, 0.0, 1.0], // was [0.533, 0.29, 0.91, 1.0],
pale sky blue
  "jPristineHex_3D_Raw": [0.283, 0.15, 1.0, 1.0], // subdued olive green
  "jNormalHex_3D_Raw": [0.086, 0.10, 0.9, 1.0],
  "jFreewayHex_3D_Raw": [0.6, 0.2, 0.85, 1.0],
  "jMinorRoad_3D_Raw": [0.2, 0.1, 0.75, 1.0],
  "jDistrictRoad_3D_Raw": [0.0, 0.0, 0.0, 1.0],
  "jMajorRoad_3D_Raw": [0.0, 0.0, 0.0, 1.0],
  "jFreewayRoad_3D_Raw": [0.0, 0.0, 0.0, 1.0],

  "jDwelling_3D_Raw": [0.4, 0.8, 1.0, 1.0],
  "jOffice_3D_Raw": [0.56, 0.8, 1.0, 1.0],
  "jService_3D_Raw": [0.8, 0.8, 1.0, 1.0],
  "jIndust_3D_Raw": [0.1, 1.0, 1.0, 1.0],
  "jRoof_3D_Raw": [0.5, 0.0, 0.85, 1.0],
```

```
  "jGroundPlaneNomRings": 30, // for a hexagon, same as number of rings;
otherwise large
  "jStoreyHeight": 0.125, // 6.0/48.0, i.e. 6 tristeps  (was 8 for taller
storeys)
  "jRelRoofHeight": 0.57735, // makes the front roof tri equilateral, with
above storey height (was 0.43301 for taller storeys)
  "jPedHeight": 5.0,

  "jCameraPosn": [0.0, -100, 44],
  "jPanAngle": 0.0,  // pirads
  "jTiltAngle": -0.15,  // pirads
  "jFov": 0.135, // pirads

  // ----- ----- ----- Random numbers ----- ----- -----

  "jRandom_MainSeed": 2000,
  "jRandom_ExtraInc": 42,
  "jUseTimeForRandom": false,
}

// <><> End file settings.bmn3.json <><> //
// <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><> //
```