Top down vs Isometric

The game that will be created will need a view point that can effectively show off the city that the player is building. Either top down or isometric will be effective for this purpose as the game will be 2d.
While Isometric may be slightly nicer to look at in some aspects, I personally think that top down is more fitting for the game I am aiming to create. The focus for this game is on depth of simulation more so than graphics, and while graphics will still be worked on with as much care as the rest of the game, the focus will still be squarely on simulating a complex city. Using top down will mean better access to assets, and better built in tools for creating the grid style map of the game. This allows more time to focus on the previously mentioned depth of simulation, meaning the city can be made as real as possible.
As a stretch task, Isometric may be looked at later in development, but the focus will be squarely on the simulation depth first

Depth of simulation

The depth of the simulation will be the main focus of the game, as mentioned in the previous section. The main aim for this game is to effectively simulate a city in a way that previous city sims have not done, which is to give added focus to building a system to check for environmental aspects.
In the game, every item will have different statistics on how they affect their environment. If a building is powered by renewable energy, it will be releasing less pollution than one powered by fossil fuels, players who build more renewable energy in their cities will see better environmental aspects developing in their city than if they built non renewables based power stations. Not building a proper waste management system and letting waste run into the water will negatively affect the water and over time the water quality of the city will be adversely affected. Small choices like that all over the city will affect the environment.
In addition to that, more traditional city sim aspects will be used. Population count will be compared against available housing, potentially seeing a decline in the city if housing is lacking , infrastructure systems will be put in place and need to be designed around, without proper transport between housing and key locations in the city, like roads and public transport, the city will see a decline, with the out of reach districts getting visited and used much less. These are just a few examples of systems that will be put in place. But The game will go deep into simulating a city as closely as possible, aiming to make the player have to think carefully about their choices and adding the responsibility of keeping their city environmentally friendly

Save/loading

Due to the continuous nature of playing the game, where players will gradually build up their city over multiple play sessions, the ability to save/reload a city is an absolute necessity.
To facilitate this feature, a database will be used. As the game is singleplayer, the game will use a local database meaning players wont require internet connection to play the game

which also has the benefit of removing the need for a hosted server to run the database for players to access.
The way this will be created will be by using an SQLlite database, which can be created essentially embedded into the main project and won't need to be hosted separately. This also has other benefits like being easy to include files wise, with everything contained in a single db file and the ability for it to work on other platforms if a port ever occurs.

In terms of how it will be stored, this will be designed more thoroughly near the start of development, but it will involve some sort of system where buildings locations, road locations, citizen information will all be saved, so that upon game load, building can be placed correctly, citizens can be placed with the correct sprite and location, and any other parts of the game will appear on load. Game stats like how much money the player has to use for the city and how much pollution the city has will also be saved here, with multiple tables being used.
The game will have an autosave feature that can be toggled on and off to save the game at frequent intervals

Performance

One of the greatest challenges the development of a simulation game will face is the issue of performance. With countless entities existing like citizens, buildings, vehicles/transportation all at the same time with all of them having their own values and updates to be calculated, the performance impacts can add up massively. This is especially notable when you consider almsot everything within that needs to be displayed graphically meaning every second hundreds of things may need to be updated multiple times. To ensure performance doesnt fall off, a few key methods will be used to ensure reasonable performance.
The first method that will be used, will be to do with graphics. Some sort of obeserver pattern system will be used, entities within the game, like citizens or vehicles will be flagged when an update occurs to do with them that would require an update(e.g a vehicle moving) . using this method, the game will no longer update eveyething every single frame, but only those things that require updates.
Another method that is used in essentially all games but will be used here will be to only render what is on screen, objects off screen will still have their statuses updated as frequently as everything else but will not be rendered unless on screen. While this method is very common,it will go a long way to keeping performance managable.
The rate at which the status of entities get updated will also be reduced to assist with performance. For example, instead of updating the position of a citizen every single frame, it may only update a few times a second. This would greatly help performance, if the game was hypothetically 30 fps and the entities were updated every 10 frames rather than every frame, that is about a 90% reduction in comparisons for these entities. Adding on to this, didfferant parts of the app may be updated on different frames, for example, the citizens may be updated at the 10th frame of a second while the buildings may get updated on the 15th. This decoupling means that the calculations are more spread out, and instead of one frame being a sudden notable drop in performance, there will be a more consistent set of frames with less notable fluctuations. In addition, the game could have a system that dynamically reduces update rate if poor performance is detected, this is more of a hypothetical idea, as

while it would be a solution to performance, it may affect other aspects of the game, so experimenting will be required for this.

Controls

The game will start off with just mouse and keyboard controls. But will aim to include controller support at a later time to support a wider amount of playstyles.
In terms of keyboard/mouse controls, the position of the camera over the city can be moved either by using WASD controls or through clicking and dragging with the mouse. This dual control is effective as it means players can choose to use just the mouse for most of the game or can use more 2 handed controls if they prefer. When the player clicks on a civilian or building they can see information/options to do with that building. In terms of city building, there will be menus that can be brought up through either clicking the UI or through keyboard inputs, from here the player can essentially drag and drop the buildings to where they want them to be, allowing for intuitive city designing.

If controller support was to be implemented, the left stick would take the place of moving the camera, as dragging with a stick would be less intuitive. Instead the right stick will be used like a mouse for selecting things, and the buttons/ direction pad will be used to select menu/UI elements
These controls are not final and will develop with the game, based on both progression of development and user feedback