TransporT
EmpirE

Basic study for project
planing and cooperation

Mattias Svederberg
M.Sc Electronic-, Civil- and RF-Engineering
Norrköping, Sweden
2010-08-29 (29 of august 2010)

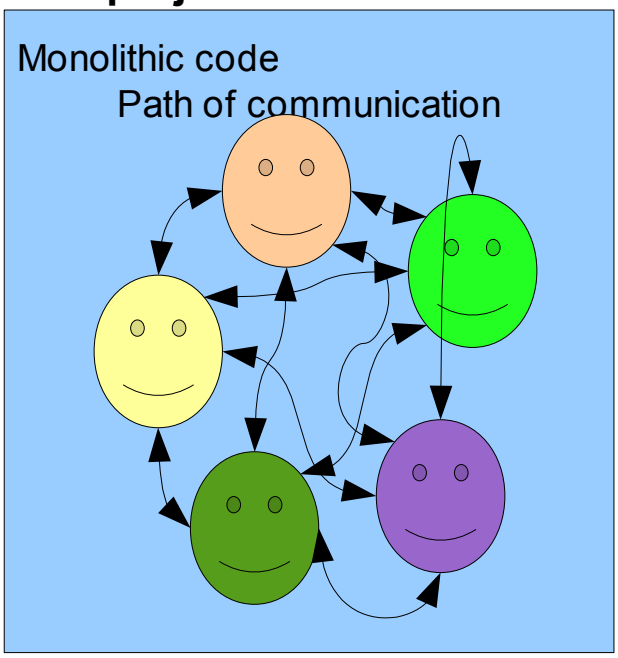
Register

Reason for study

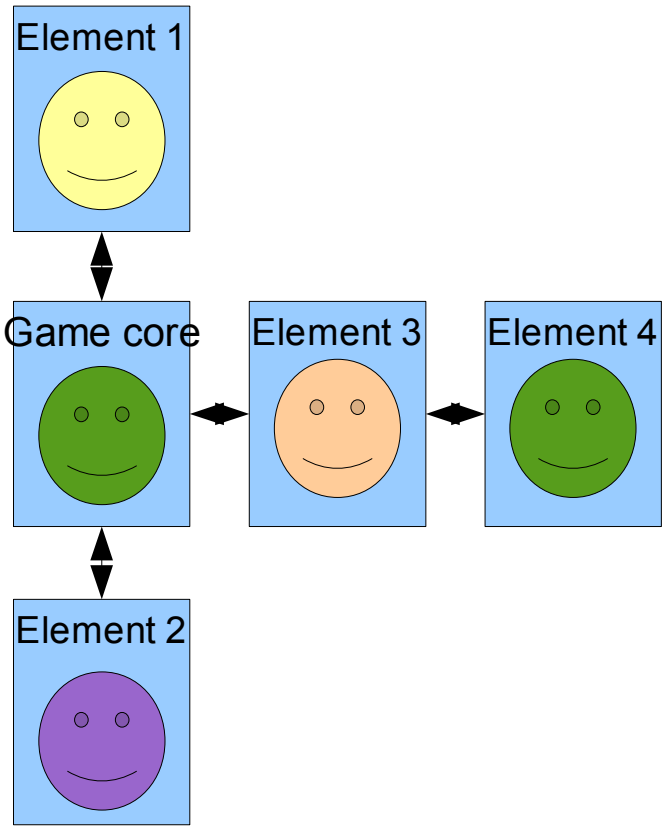
There is several reasons for this study. The main reason is to make cooperation between programmers easier. But several reasons is important in the global prospectives.

- Making cooperation easy between programmers and GFX people
- Avoiding conflicts
- Making the project more democratic avoiding project love
- Minimize micromanagement
- Making as much code as possible recyclable
- Making team members how quit replaceable
- Isolating problem spots, making problems easer to solve.
- Minimize the time needed for programmers to cooperate, making more time for coding
- Reducing the risk of total failure
- Make it possible to recycle code for a possible TransporT EmpirE 2
- Possible to test run and reduce bugs in code before the whole game is finnised
- Specialising coders making more use of less skilled coders

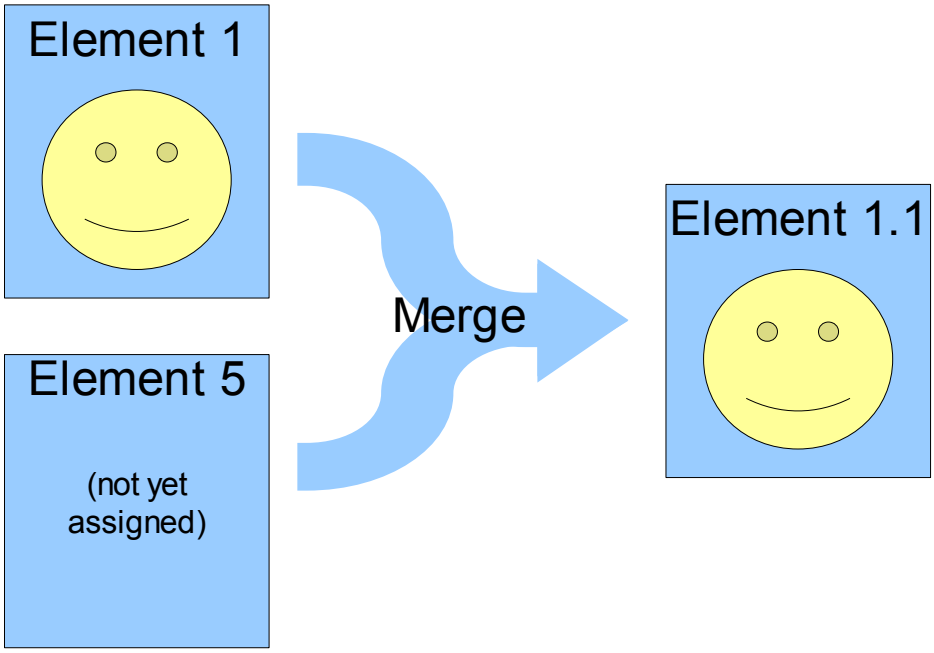
How project isolation works



Less communication equals less communications errors, and less time on communication and more time on actual work. Every teem member only have to communicate with members hows directly affected.

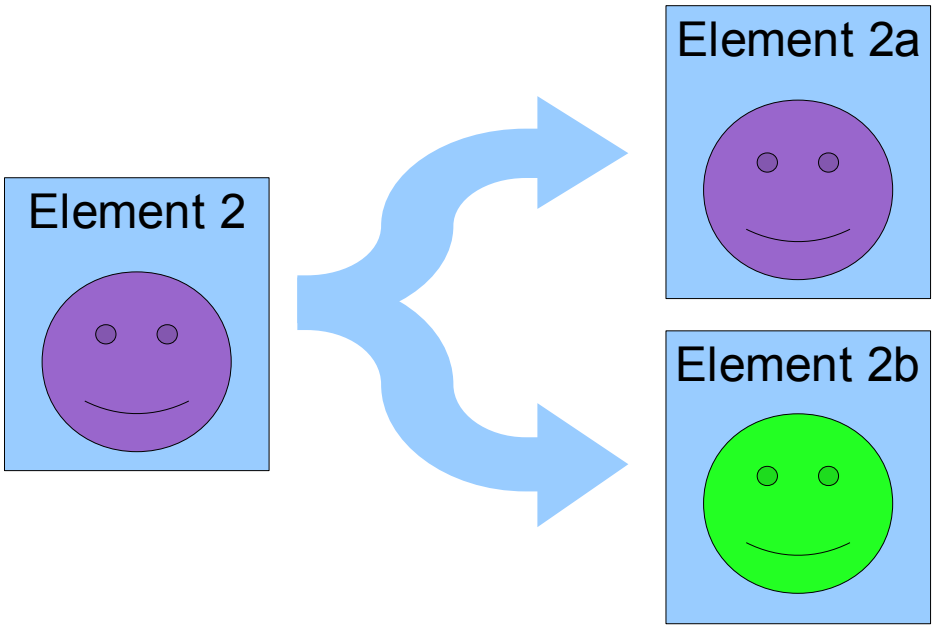


Merge

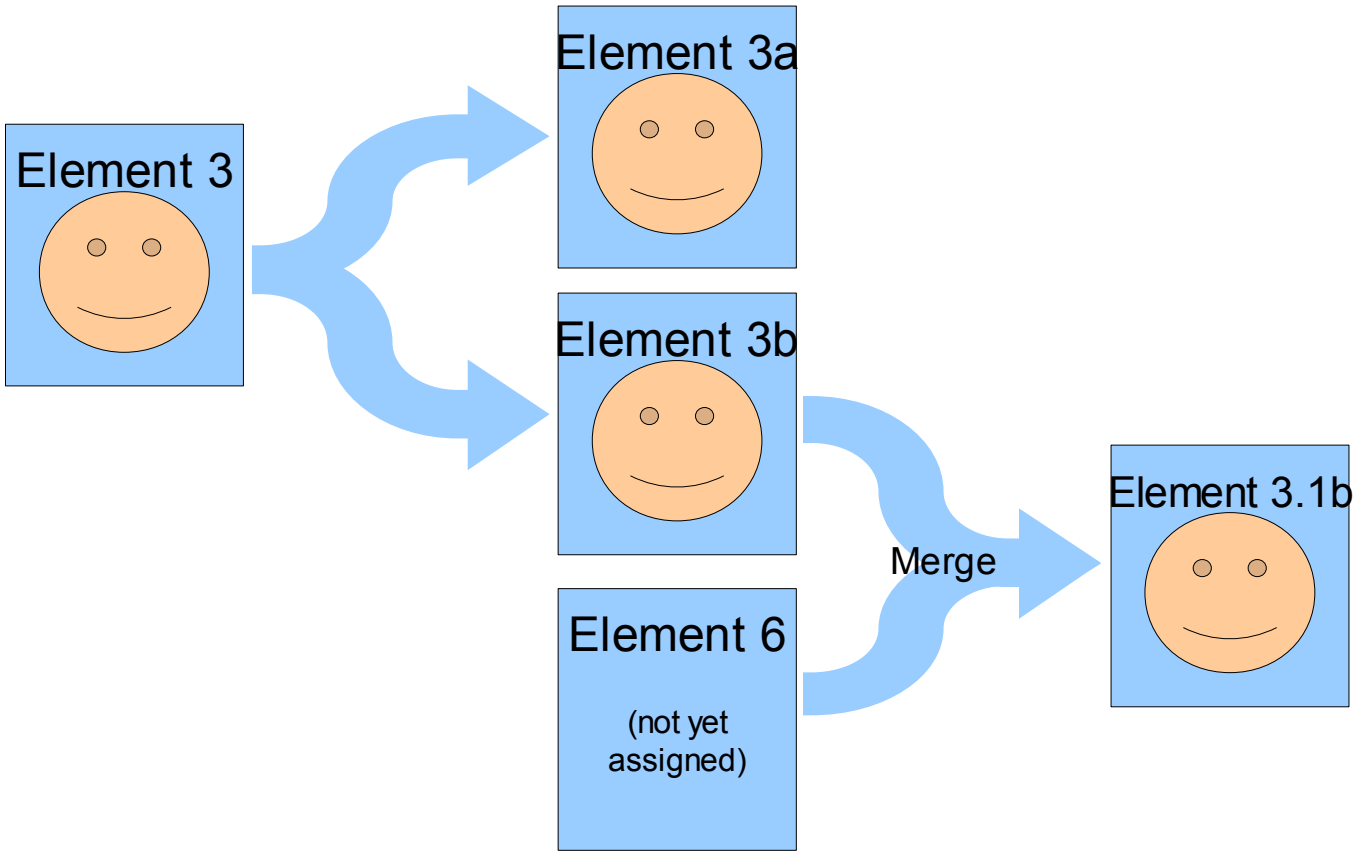


Element 5 is very similar to element 1, but no one noticed until element 1 is almost finished. But because element 5 is not yet started it can simply be merged with element 1 updating element 1 to interoperate new features making it replace element 5. Element 1 is renamed 1.1 and element 5 is scraped with minimal loss of manpower.

Split



The purple person is have way working throw element 2 when its obvious that the element 2 is a to hard workload for the purple person and he is over his head in work. But he already done a lot of great work with half of the element 2. With a split of element 2 into element 2a and 2b the purple person can finish up the part he already started while the green person can get started with the harder element 2b part simultanius.



Merge/spilt

It´s important that the blocks always is alive. Keep spiting an merging keeping them alive if the programmers falling behind. This can get other effects to. The brown person just realized that part of the code he used in element 3 is almost exactly what's needed to make element 6. So he split out the part of the code and name it element 3b, then he change it and merge it in to element 6 and call it element 3.1b.

Recycling code is a effective way to minimize workload, but to make it possible all programmers must know at all time what is needed to be done. To make this information available a project map have to be updated every time something change. A lot of administration, of cause, but it saves a lot of work to, and more important, it transfers workload to highly demanded coders to less demanded administrative personnel.

Main elements

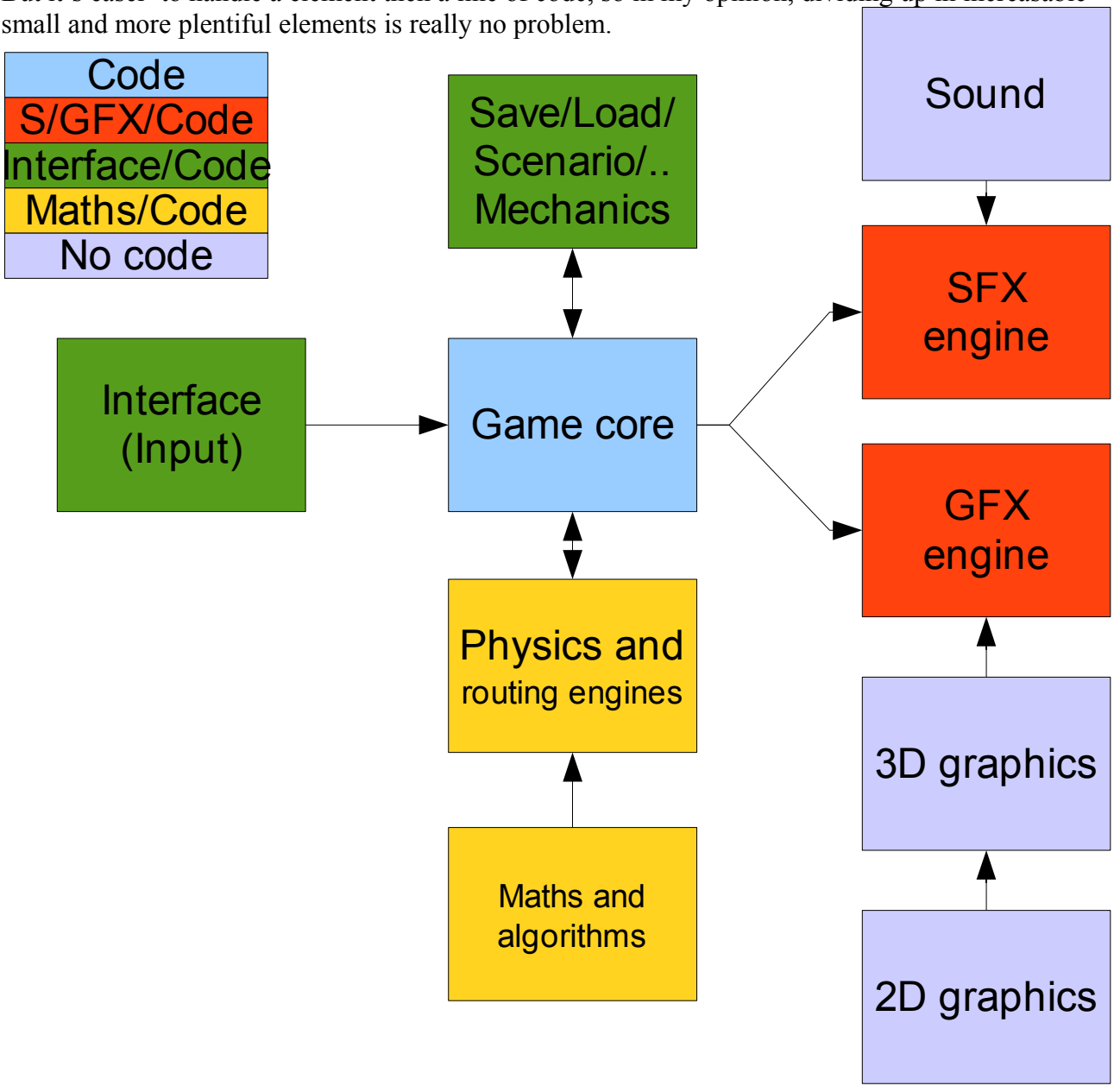
A complete game will need a lot of elements, over a 100, possibly over 1000. To make a map of that amount of elements is possible but not practical. To make it more practical I divide it up in main elements, elements an sub elements.

Main elements example: Game core, Engines, Interface

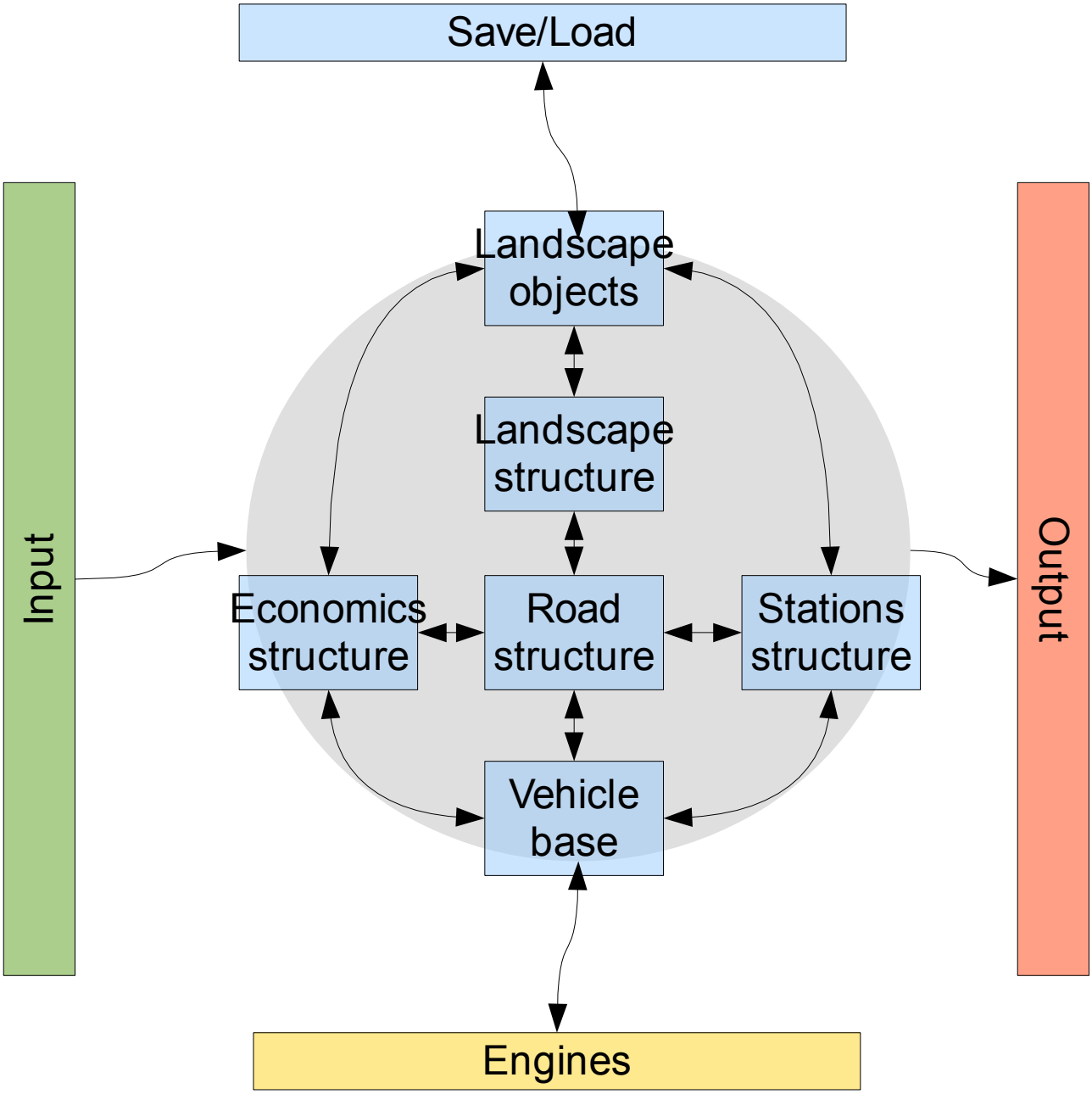
Elements example: physics engine, landscape generator, economics

Sub elements example: gods generation, revenue handle, rail dynamics

Of cause at some point maybe a sub-sub element is needed, but that's something fore the future to decide. Remember nothing is set in stone, this is only example and something to start working with. But it´s easer to handle a element then a line of code, so in my opinion, dividing up in increasable small and more plentiful elements is really no problem.

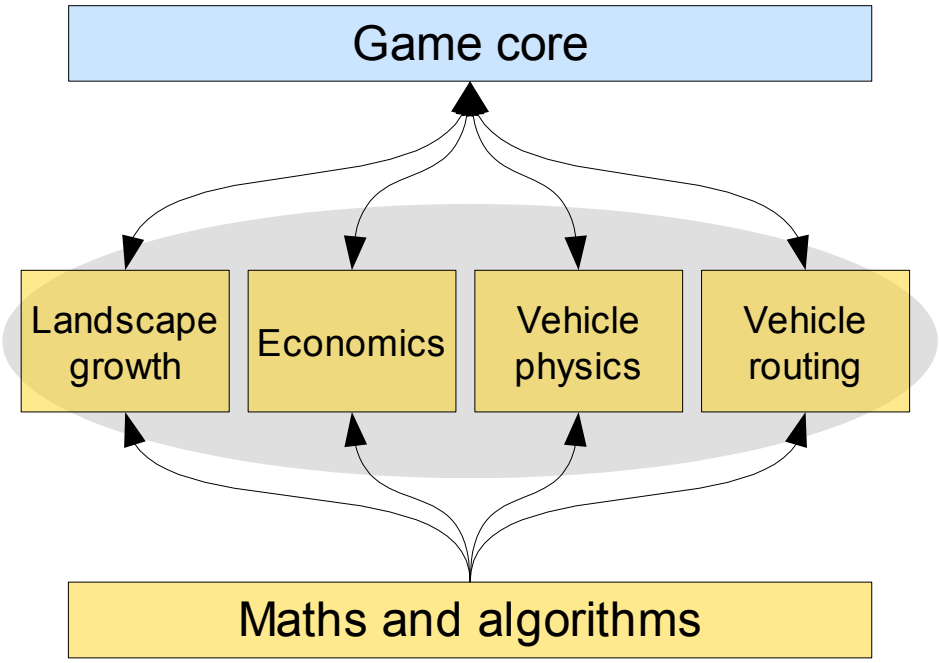


Normal elements – game core



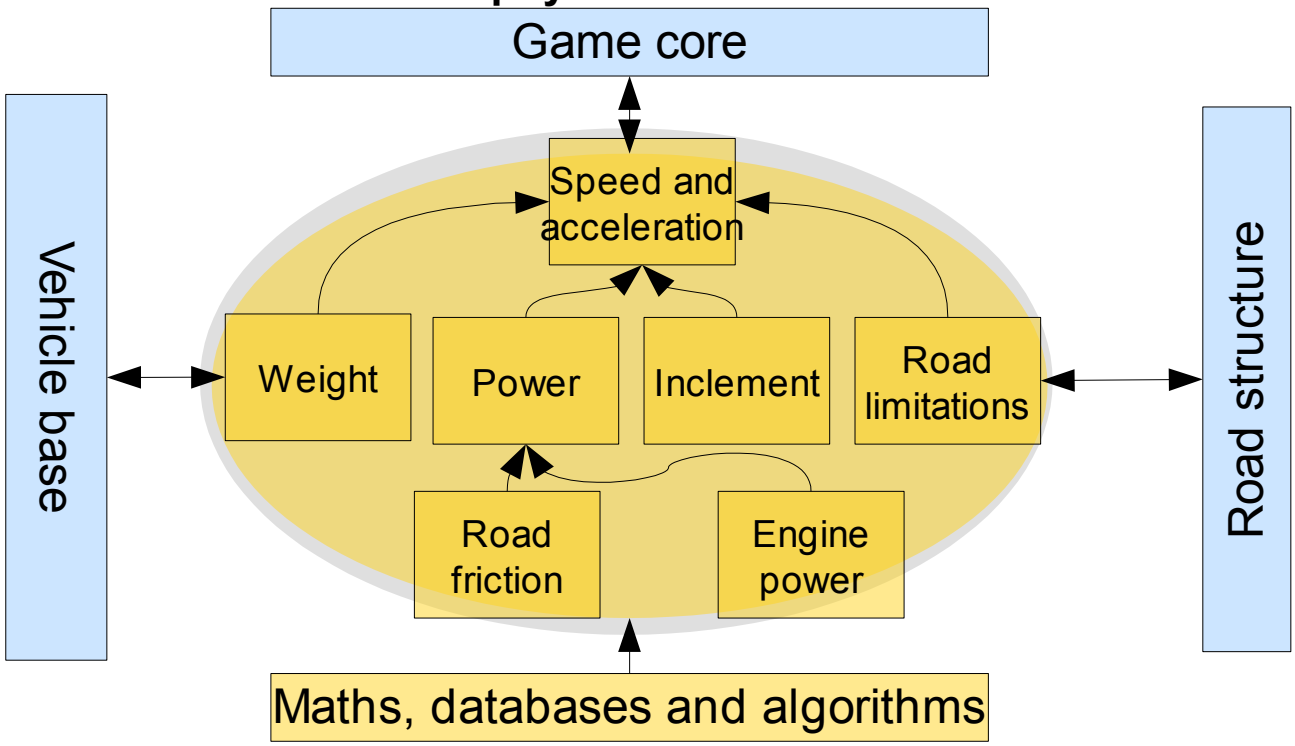
Example of what can be contained inside the game core. This is only mechanics indexing what object is were I relation to what.

Normal elements – engines



That's the only to main elements I intend to break a part, mostly because I don´t know a lot about the other ones, and that´s actually the whole point. Some one else can create them just as easily and make the it complete. But I can go deeper inside a normal element and see what can be inside. And of cause, this is not the end for the physics and game core main elements, they can, and probably will be expanded.

Sub elements – Vehicle physics



Here in the sub element several features is combined. Because of the general interfaces “road limitations” can ask road structures directly for information. It compile the structure (i.e. quality, inclination, curvature) to determinant a maximum allowed speed. The same is true about the weight program.

This way all the sub elements is quite simple little independent programs. If there is a problem, it can be located and solved whit ease.

Conclusion

This is not a standard “How to make a game”. It´s a framework or suggestion how to start planing. What´s important is that the plan is dynamic and constantly changing. All the changes have to be interoperated in the master plan to make all project attendees aware of the current status.

Also important is to break down difficult task to many easy, making its possible for less skilled people to contribute with out loading down skilled members with questions.