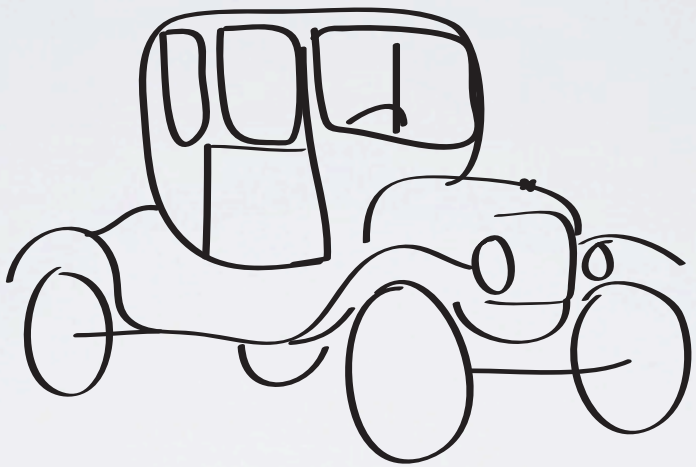
Project Overview

By Patrick Hebron

Design Through History



When a person sits down to design a solution to a problem - whether it's geometry, mechanics or pure logic - they are in many ways tied to the history of how others have solved the component problems of the challenge at hand.

Questioning one of these component assumptions could lead to new possibilities for the system as a whole.

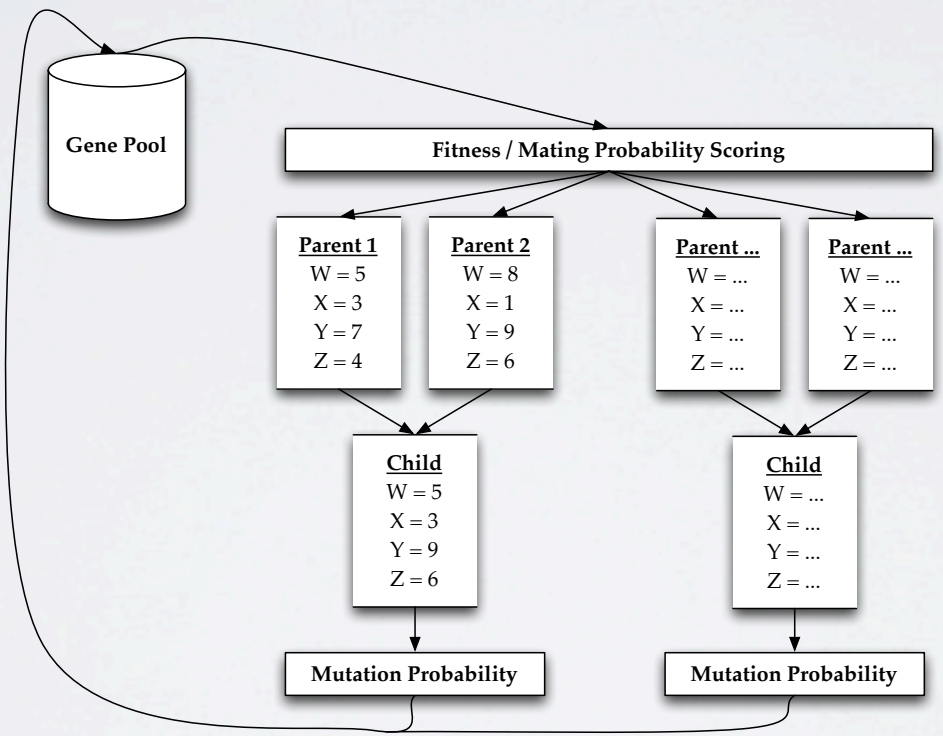
But doing so may require the reworking of numerous embedded assumptions.

To warrant this effort, the value of the newly-opened possibility should be substantial.

Yet, we cannot always recognize this value without undergoing a lengthy process of experimentation and contextualization.

Genetic Algorithms & "Ahistorical" Design

Genetic algorithms pose an alternative to this historical path. The approach they take to reach a viable solution may have nothing to do with the canonical assumptions a person would likely draw upon. The algorithm tests a variety of options across the spectrum of possible approaches and is only concerned with the extent to which each individual meets certain criteria.



With enough computing power, it seems we could invent centuries of technology overnight...

Genetic Algorithms Are Not Magic Bullets

There is often a big difference between a great design and one that people want to live with. Even the greatest of designers have sometimes had difficulty getting others to adopt their ideas.

Those zany augmented-reality glasses walking around the Media Lab in the 90s quickly became ubiquitous, but in our pockets rather than on our faces.

It is unlikely that genetic algorithms would find the path from headset to iPhone without guidance.

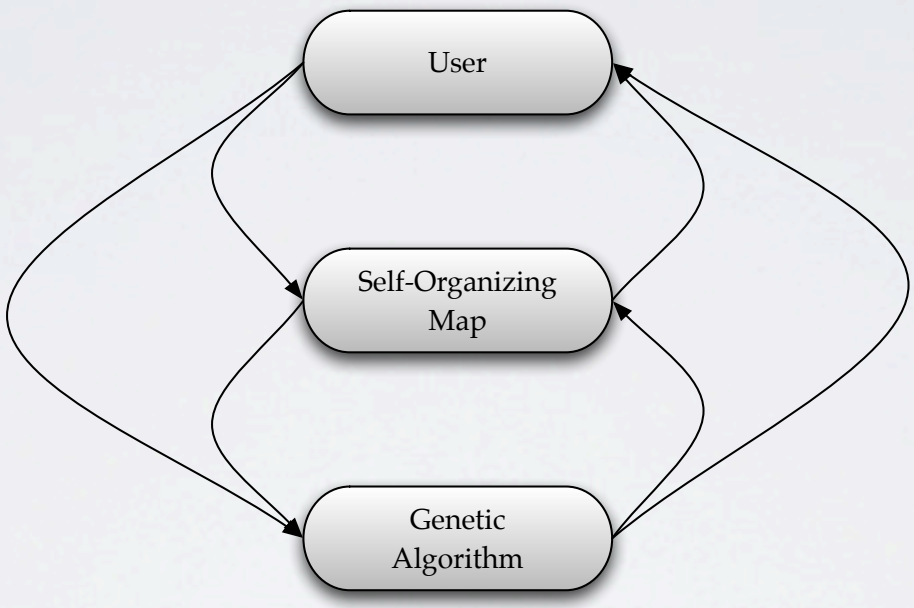
In general, people are needed to establish the significant questions and goals within a field of study.

If genetic algorithms are to become truly useful to design, it will only be through a well-integrated partnership with human users.

The goal of my thesis work on Arrow is to develop a CAD tool that uses genetic algorithms and self-organizing maps to facilitate an intuitive process of design iteration and version-control.

This process should be driven as much as possible by user-interaction concepts for which there already is widespread literacy.

Arrow's core premise is that every facet of the design process should be informed by every other. Everything should be folded back in - the genetic algorithm's efforts should help the user to clarify his or her ideas and the user's efforts should help the computer to clarify its goals and understanding of the project. The idea is to generate discourse between the user and the machine.



Arrow serves as an organizational and version-tracking assistant to the user while also folding the data gathered by this process into the gene pool and selection behaviors of the genetic algorithm. A self-organizing map helps to mediate this exchange of information.

Modeling Tools

Before anything else, Arrow is a CAD program and will provide a standard set of modeling tools.

These should be easy to use, but capable of building complex forms.

The modeling aspects of Arrow will borrow conceptually from programs like SketchUp and AutoCAD.

The key difference from those applications is that the Arrow Modeler will encourage (more or less require) the user to save as many iterations of the design as reasonably possible.

Though Arrow will not enforce a particular iterative style such as the “10-3-1” model used at Apple, the user will be encouraged to experiment freely in the early phases of the design process so as to generate a great deal of variation in the initial set of design iterations.

The Browser

The Browser panel is the central hub of Arrow's version control system and is designed to tap into some of the interaction concepts found in programs like iPhoto.

The Browser's purpose is to allow users to:

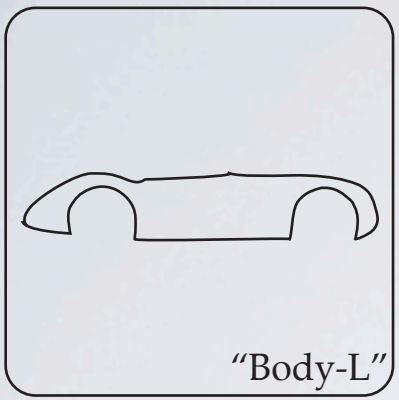
• Compare various iterations or a particular component across each iteration.

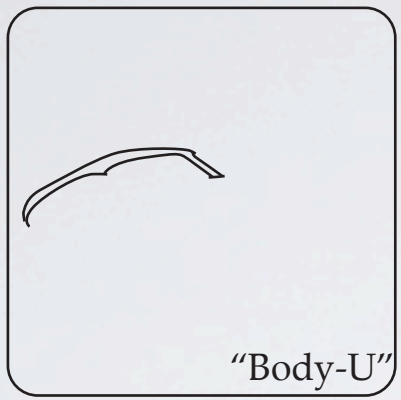
• Rank iterations and components. This is meant to help the user organize his or her thoughts while also giving the genetic algorithm another mode of insight into the user's design goals.

• Generate new iterations through the mixing and matching of components. If the pieces don't match up, the self-organizing map and genetic algorithm can assist in determining the necessary adaptations.

Project Iterations:

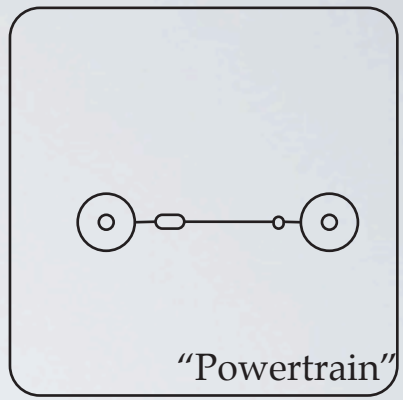
"Initial concept" 2/6/2011 @ 2:10pm USER GENERATED


"Body-L"

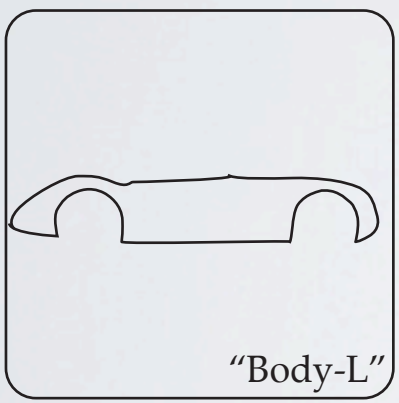

"Body-U"

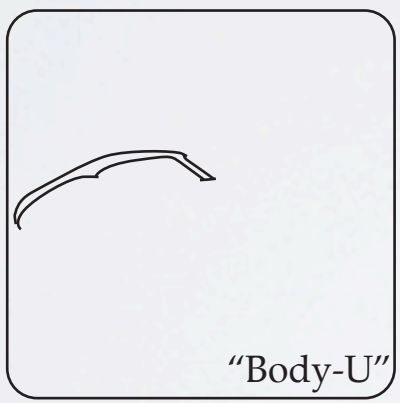

"Door"

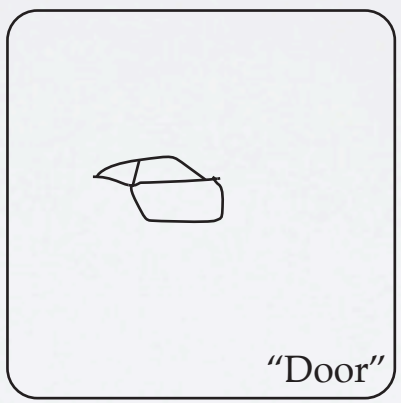

"Mirror-RV"


"Powertrain"

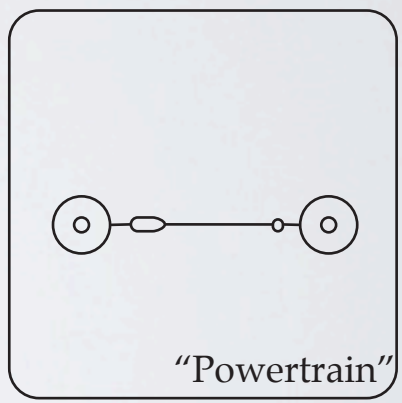
"Longer Powertrain" 2/8/2011 @ 5:23pm USER GENERATED


"Body-L"

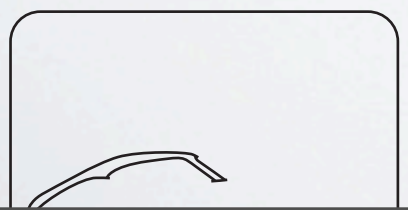

"Body-U"

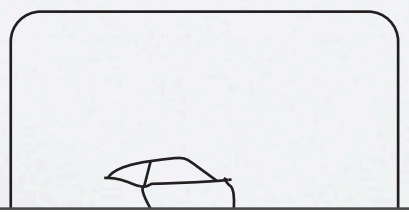

"Door"

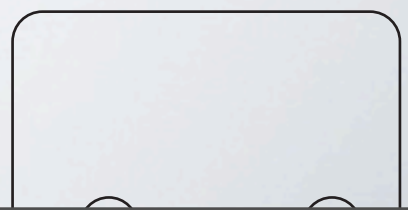

"Mirror-RV"


"Powertrain"

"Initial concept - Fork #1" 2/7/2011 @ 7:43am COMPUTER GENERATED







The Mapview

The Mapview centers around a self-organizing map, which generates a topologically coherent representation of how the features of each iteration relate to those of each other.

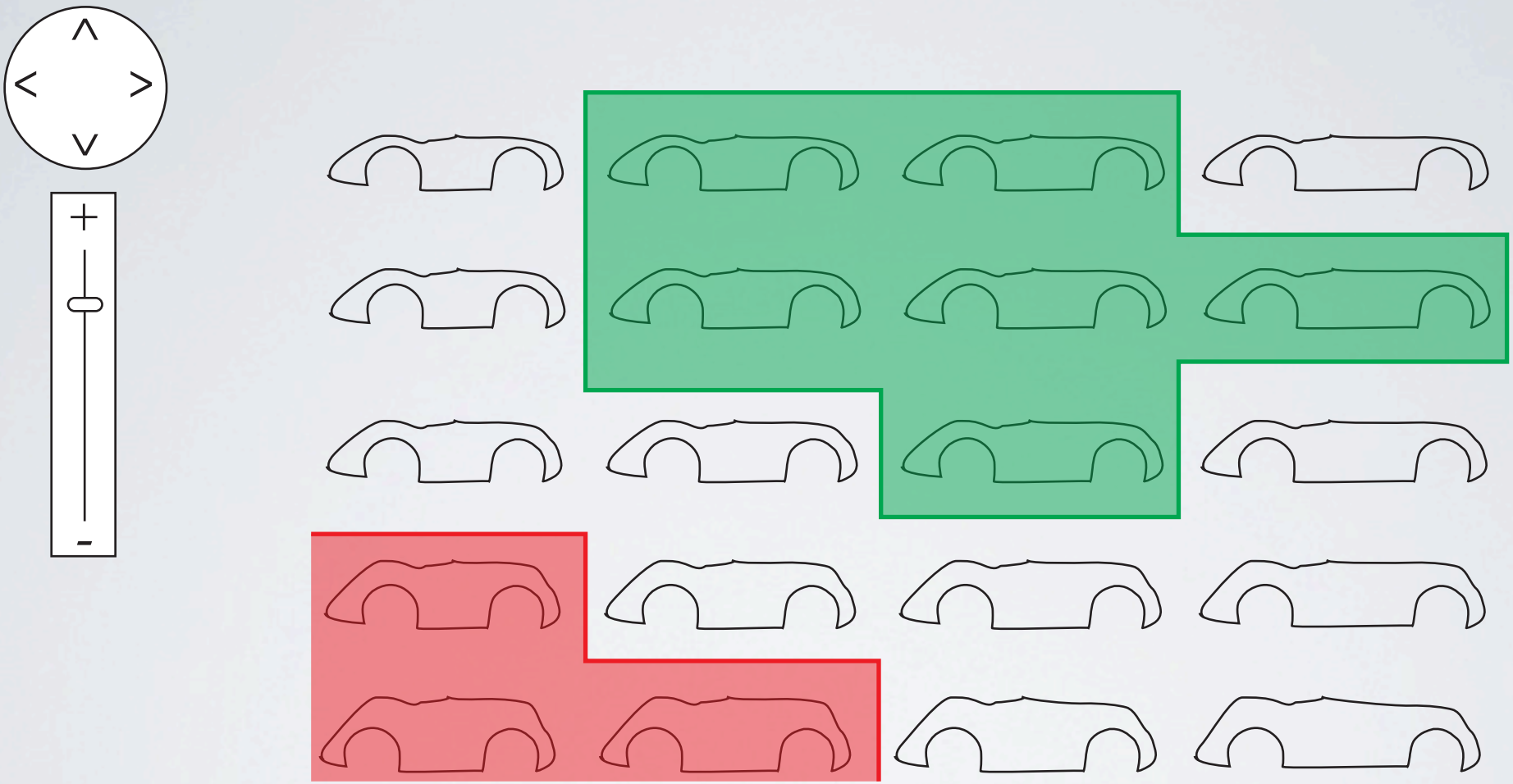
A Google Maps style of interaction allows users to explore the design space and identify regions of the map that warrant further consideration.

The Mapview does not enable the blazing of truly new terrain, but allows the user to explore the space between the iterations already created and add iterations from these regions to the project's pool.

As new iterations are generated by the user or genetic algorithm, they are automatically inserted into the map (and flagged for approval).

The Mapview joins the Browser in serving as an organizational aid, but also acts as a tool for the creation of new iterations through the user's marking of desirable regions of the map.

Form Map: “Body-L”



Genetic Algorithm Mark-ups

Unlike the self-organizing map, the genetic algorithm's purpose is to generate new iterations which take design approaches that were not necessarily considered by the user.

A visual mark-up language, applied within the Arrow Modeler, allows the user to inform the algorithm of which features are most important to the design.

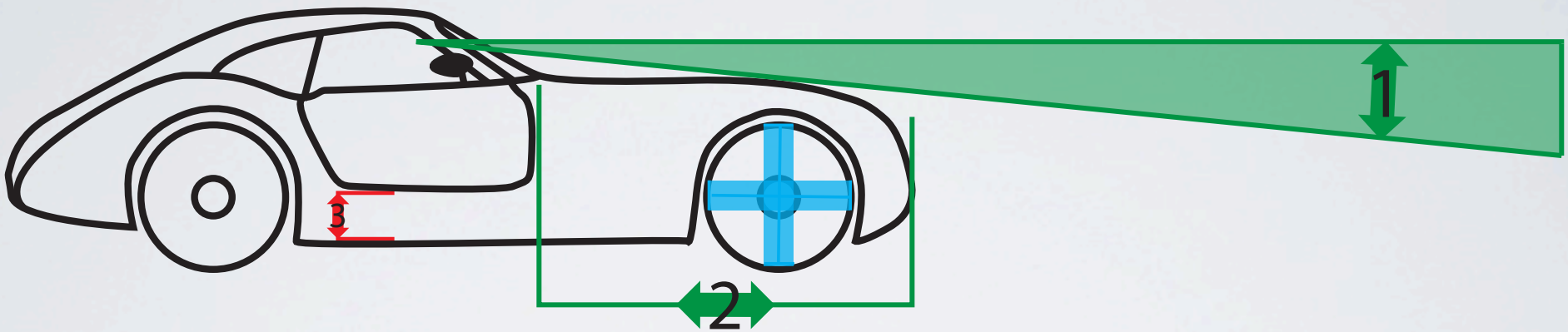
Fitness criteria are stipulated by these mark-ups and their priority ordering. Maximize, Minimize and Static mark-ups tell the algorithm which features the user would like to emphasize, de-emphasize or keep untouched.

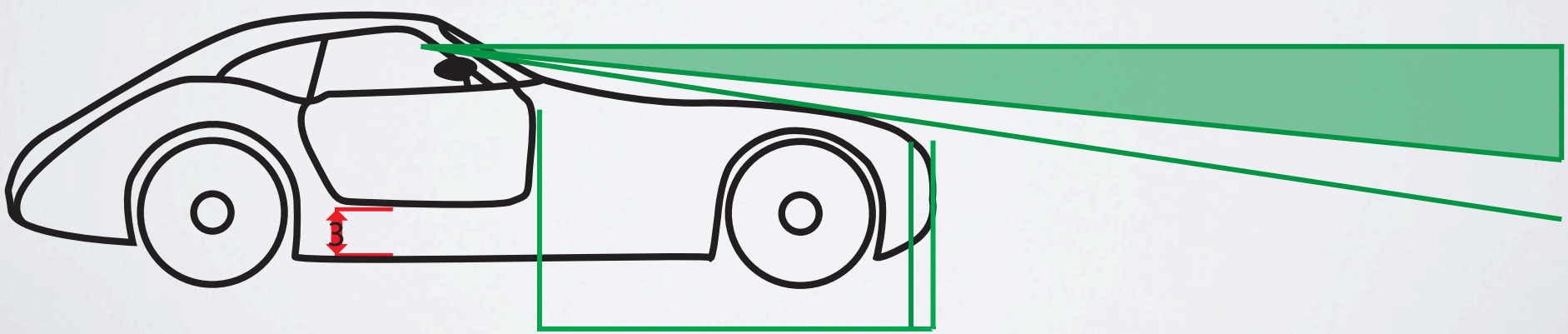
Maximize and Minimize mark-ups share a priority list. Static mark-ups must be satisfied or the individual will be discarded.

The project iteration pool (or a subset thereof) forms the genetic algorithm's initial population. Over numerous generations, the algorithm employs a fitness selection process in order to evolve iterations that meet the user-specified requirements.

Genetic Algorithm Mark-ups:

STATIC | MAXIMIZE | MINIMIZE





Other Ideas... For After Thesis

Aside from the refinement of the features already mentioned, I have identified three areas that I believe are essential to Arrow's conceptual development, but for reasons of time will not be addressed in my thesis work.
These are:

Physics - Including a physics engine in Arrow would open up a number of possibilities for the user and would also enable the genetic algorithm to be of much greater service to the refinement and refactoring of designs that will ultimately operate in the real world.

Cloud Processing - In its finished state, Arrow's client application should run on a variety of desktop and tablet operating systems and communicate with the cloud, where project files would be hosted and the machine-learning and other time-intensive procedures would be computed.

User Network - To better facilitate the exchange of ideas, Arrow should provide an online user network that enables the sharing of design files between members and provides integration with rapid-prototyping and 3D printing services.